UNIVERSITÄT BAYREUTH

Building Scientific Tools: From Heat Transport to Particulate Systems

Dissertation

Zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
Dr. rer. nat.

An der
Universität Bayreuth
Fakultät für Biologie, Chemie und Geowissenschaften

Vorgelegt von
Bernd Friedrich Alfred Kopera
geboren in Weiden i. d. Opf.

December 17, 2019

Die vorliegende Arbeit wurde in der Zeit von September 2016 bis Dezember 2019 in Bayreuth am Lehrstuhl Physikalische Chemie I unter Betreuung von Herrn Prof. Dr. Markus Retsch angefertigt.

Vollständiger Abdruck der von der Fakultät für Biologie, Chemie und Geowissenschaften der Universität Bayreuth genehmigten Dissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.).

Dissertation eingereicht am: 17.12.2019

Zulassung durch die Prüfungskommission: 08.01.2020

Wissenschaftliches Kolloquium: 15.05.2020

Amtierender Dekan: Prof. Dr. Matthias Breuning

Prüfungsausschuss:

Prof. Dr. Markus Retsch (Gutachter)
Prof. Dr. Jürgen Senker (Gutachter)
Prof. Dr. Josef Breu (Vorsitz)
Prof. Dr. Mukundan Thelakkat

I would like to die on Mars,
just not on impact.
- *Elon Musk*

Für Resi

Contents

1 Abstract **xix**

2 List of Publications **xxv**

3 List of Contributions **xxvii**

4 Acknowledgments **xxix**

5 Introduction **1**

5.1 Motivation 1
5.2 Particulate Systems 2
5.3 Heat Transport 6

6 Methods **9**

6.1 The Programming Language Python 3 9
6.2 Analyzing Stretched Hexagonal Lattices 10
6.3 Computing Custom Colloidal Glasses 24
6.4 Xenon Flash Analysis 26
6.5 The Photoacoustic Technique 27
6.6 Lock-in Thermography 28

7 Thesis Overview **43**

7.1 Graphical Abstract 43
7.2 Synopsis 44
7.3 My Contributions to Joint Publications 59

8 Outlook **63**

9 References **65**

10 Access to Any Two-Dimensional Bravais Lattices **77**

10.1 Abstract 78
10.2 Introduction 78
10.3 Experimental Section 80
10.4 Results and Discussion 81
10.5 Conclusion 89

10.6 References 91
10.7 Supporting Information 94

11 Computing the 3D Radial Distribution Function 99
11.1 Abstract 100
11.2 Introduction 100
11.3 Theory 102
11.4 Results 107
11.5 Conclusion 111
11.6 References 112
11.7 Supporting Information 114

12 Thermal Transport in Binary Colloidal Glasses 165
12.1 Abstract 166
12.2 Introduction 166
12.3 Experimental Section 168
12.4 Results and Discussion 170
12.5 Conclusion 178
12.6 Appendix: Supporting Information 179
12.7 References 183

13 Low Thermal Conductivity through Dense Particle Packings 187
13.1 Abstract 188
13.2 Introduction 188
13.3 Experimental Section 199
13.4 References 200
13.5 Supporting Information 203

14 Tunable Thermoelastic Anisotropy in Hybrid Bragg Stacks 217
14.1 Abstract 218
14.2 Introduction 218
14.3 Results and Discussion 220
14.4 Conclusion 231
14.5 Experimental Section 232
14.6 References 233
14.7 Supporting Information 239

15 Breathable and Flexible Polymer Membranes 261
15.1 Abstract 262
15.2 Introduction 262
15.3 Results and Discussion 264
15.4 Conclusion 274
15.5 Experimental Section 274

15.6 References 277
15.7 Supporting Information 281

16 Measuring Semitransparent Thin Films by Lock-In Thermography 289
16.1 Abstract 290
16.2 Introduction 290
16.3 Modeling and Experimental Section 291
16.4 Results and Discussion 297
16.5 Conclusion 303
16.6 References 305
16.7 Supporting Information 307

17 Reducing DC Heating in Lock-in Thermography 323
17.1 Abstract 324
17.2 Introduction 324
17.3 Methods 327
17.4 Results and Discussion 332
17.5 Conclusion 338
17.6 References 339
17.7 Supporting Information 343

A Source Code 361

List of Figures

5.1 Examples for 2D and 3D colloidal assemblies 2
5.2 Parameters for heat transport in colloids 5

6.1 Stretch and hexagonal lattice vector definition 10
6.2 Stretch vector decomposition 11
6.3 Phase line for reduced primitive lattice vectors of equal length 14
6.4 Phase diagram for orthogonal lattices 16
6.5 Uniaxial stretching of a hexagonal lattice 17
6.6 Primitive hexagonal lattice vectors 18
6.7 Vector projection for lattice reduction 18
6.8 2D histogram of inter-particle distances 20
6.9 Particle detection in the SEM image of a stretched monodomain . . . 21
6.10 Different coordination regions around a large central particle 22
6.11 Creation of complex 3D particle assemblies 24
6.12 Complex 3D particle assembly 25
6.13 XFA setup and raw data 26
6.14 Photoacoustic measurement cell 27
6.15 Surface temperature vs. time during lock-in thermography 28
6.16 Combinations of sample and heating dimension in LIT 29
6.17 Possible cominations of signals and data analysis in LIT 31
6.18 Black body spectrum 32
6.19 Components of our lock-in thermography setup 33
6.20 Pixel size calibration standard 34
6.21 Pixel size calibration for lock-in thermography 34
6.22 Temperature amplitude and phase for a point source 35
6.23 Linearized temperature amplitude and phase 36
6.24 Temperature amplitude and phase images 37
6.25 Vector representation of the line laser 38
6.26 Linearized temperature amplitude and phase 39
6.27 Direction dependent thermal diffusivity of Kapton 40
6.28 Class hierarchy of my lock-in thermography analysis software 41

7.1 Graphical abstract of my PhD thesis 43
7.2 Z-type transfer of a hexagonal monolayer 44
7.3 Uniaxial stretching of a hexagonal lattice 45

7.4 SEM images of non-close packed domains 46
7.5 Radial distribution function comparison 47
7.6 Radial bins extending beyond the sample boundaries 48
7.7 Results for the analytic $g(r)$ algorithm 49
7.8 Heat transport in colloidal glasses 50
7.9 Histograms of next neighbors and touching neighbors 51
7.10 Structure of a stratified Hectorite/PVP composite 52
7.11 Properties of a stratified Hectorite/PVP composite 53
7.12 Strain induced anisotropy in elasitc nonwovens 54
7.13 Cracks in stretched silve nanowire layers 55
7.14 Heat losses in a semi-transparent film excited by a point laser 56
7.15 Validation data for semi-transparent lock-in thermography 57
7.16 Temperature amplitudes and DC components 58

10.1 Fabrication of non-close packed monolayers 81
10.2 Stretched particle arrays 83
10.3 Phase diagram for Bravais lattice structures 86
10.4 Uniform stretching of multicrystalline monolayers 87
10.5 SEM images of particle monolayers 94
10.6 Side view SEM images of non-immobilized and immobilized particles . 94
10.7 SEM images of particle monolayers 95
10.8 Photograph of an immobilized particle monolayer 95
10.9 Streamline profiles upon immersion of a solid substrate 96
10.10 SEM image of a stretched monodomain 96
10.11 2D histogram of the distance between particle pairs 97

11.1 Radial distribution functions for a Gaussian disturbed FCC lattice . . 101
11.2 Possible sample boundary shapes 103
11.3 Cross section through a box intersected by a spherical shell 104
11.4 Cartesian coordinate system with the central particle at the origin . . 104
11.5 Comparison of radial distribution functions 107
11.6 Maximal useful radial distance 109
11.7 Runtime comparison for the two $g(r)$ algorithms 110
11.8 Slightly distorted cubic grid of particles 115
11.9 Small sample volume extended by periodic boundary conditions 116
11.10 Colloidal assemble measured with confocal scanning microscopy 116
11.11 Effects of periodic boundary conditions 117
11.12 $g(r)$ of a simple cubic lattice 117
11.13 Finite sample volume with a smaller subvolume 118
11.14 Radial distribution function for our FCC test data set 118
11.15 Intersection volume between a spherical shell and a rectangular box . 119
11.16 Definition of the box boundaries 120

11.17 The volume of a spherical shell 120
11.18 Separation of a sphere into eight octants 121
11.19 The three boundary planes 122
11.20 Intersection volume between a box corner and an octant 123
11.21 Single boundary intersecting the sphere octant 123
11.22 Two boundaries intersecting the octant 124
11.23 Spherical cut created by slicing a sphere with two planes 125
11.24 Decomposition used to calculate the volume of a spherical cut 125
11.25 Volume of a spherical segment 128
11.26 Volume of a spherical cap 129
11.27 Partial sphere volume by integrating a surface 130

12.1 Effective thermal conductivities in colloidal glasses 168
12.2 Finite element modeling (FEM) of binary colloidal assemblies 170
12.3 TEM and SEM images of solid and hollow particles 172
12.4 Schematic setup of the vacuum filtration system 172
12.5 Measured thermal conductivity data of binary colloidal glasses 174
12.6 Scanning electron microscopy images of a colloidal assembly 175
12.7 SEM images of the binary colloidal glass mixing series 176
12.8 Density and heat capacity of binary colloidal glasses 177
12.9 Temperature dependent thermal conductivity of colloidal glasses . . . 178
12.10 Schematic of the xenon flash analysis setup 181
12.11 Radial distribution function of random close-packed particles 182

13.1 Structural characterization of binary colloidal assemblies 189
13.2 Thermal conductivity of the investigated assemblies 191
13.3 Molecular dynamics simulations of a binary particle mixture 195
13.4 Influence of the size ratio on the thermal conductivity 197
13.5 Ashby plot of density versus thermal conductivity 198
13.6 Mean specific heat capacity of polystyrene binary assemblies 204
13.7 Side views received from MD simulations 208
13.8 FFT comparison between the particle structures 208
13.9 Histograms of the next neighbors per particle 209
13.10 Evaluation of the number of contacting next neighbors 209
13.11 Comparison of contact area and stream line length 210
13.12 Dependency of normalized thermal conductivity on contact area . . . 211
13.13 Statistic evaluation of contacting neighbors 212
13.14 Comparison of the effective thermal conductivity 214
13.15 Effective thermal conductivity of the particle assembly 214

14.1 Ultra-anisotropic and extremely confined Hec/PVP Bragg stacks . . . 221
14.2 Thermal conductivities of Hec/PVP hybrid Bragg stacks 223
14.3 BLS measurements and strong mechanical anisotropy 227

14.4 Analysis of anisotropic thermomechanical coupling in Hec/PVP 230
14.5 Atomic force microscopy image of a delaminated Hec nanosheet 239
14.6 1D SAXS pattern of the concentrated gel sample Hec40/PVP60 240
14.7 TGA curves of four hybrid Bragg stacks 241
14.8 Space filling models of PVP and Hectorite 241
14.10 Lattice plane termination of a single Hec nanosheet 242
14.11 SEM image and photograph of the hybrid film 244
14.12 Temperature and composition dependencies of the specific heat 245
14.13 Scheme of the lock-in thermography set-up 245
14.14 Exemplary two-dimensional lock-in thermography phase images 246
14.16 Scheme of our photoacoustic instrument 248
14.18 BLS experiments in the reflection geometry 252
14.19 BLS spectra and peak intensity 253
14.20 Direction-dependent sound velocities of hybrid stack films 253

15.1 Schematic of preparing the TPU-AgNW/PCL-TPU membranes 265
15.2 Photographs and SEM images of electrospun TPU nonwovens 266
15.3 Stress-strain curves of sandwich membranes 269
15.4 Laser scanning microscopy images of TPU fibers 271
15.6 Photograph of the AgNW dispersion 281
15.7 Optical microscopy image of short PCL fibers 281
15.8 SEM image of TPU-PCL-TPU 281
15.9 Pore size distribution of the sandwich membranes 282
15.10 Thickness of sandwich membranes with different amounts of AgNWs. . 282
15.11 Sheet resistance stability of AgNW in air 282
15.12 Images of the color change of an aqueous basic solution 283
15.13 100-cycle stress-strain curves of TPU-2.0AgNW/PCL-TPU 283
15.14 Schematic diagram of hotspots appearance by stretching 284
15.15 Polar orientation maps for the TPU surface fibers 284
15.16 Thermal diffusivity as a function of direction 284
15.17 SEM images of AgNWs mesh without stretching 285
15.18 SEM images of AgNWs mesh under strain 286
15.19 Dedicated adjustment of the local electrical resistance 286
15.20 Sandwich membrane utilizing as a strain sensor 287

16.1 Semitransparent film heated by a modulated Gaussian laser beam . . 292
16.2 Plots of $\ln(|T\sqrt{w}|)$ and Ψ 296
16.3 Plot of A_{IR} versus A_{Vis} of all polymeric thin films 298
16.4 Exemplary measurement data of a PMMA film 300
16.5 Thermal diffusivity of PMMA and LDPE 300
16.6 Amplitude and phase of the surface temperature 309
16.7 Product between amplitude and phase derivatives 310

16.8 Photographs of LIT samples 310
16.9 Micrographs of LIT samples with PR 311
16.10 Schematic set-up used for lock-in thermography measurements 311
16.11 Influence of carbon coating on the optical properties 312
16.12 Optical properties of the thin, free-standing polymeric films 313
16.13 XRD pattern of thick PMMA samples without PR 316
16.14 XRD curves of amorphous PMMA and semi-crystalline LDPE 317
16.15 2D SAXS patterns of LDPE films 319
16.16 2D SAXS patterns of thick LDPE samples 320
16.17 Thermal diffusivity of PMMA and LDPE 321

17.1 Components of our lock-in thermography setup 327
17.2 Temperature signal from square waves 328
17.3 DC offset for lock-in thermography measurements 332
17.4 DC temperature reduction with square waves 333
17.5 DC temperature reduction with two sine waves 334
17.6 Temperature signal from multiple sine waves 336
17.7 Multi frequency lock-in thermography results 337
17.8 Two sine interferrence signals 343
17.9 Numerically optimized sum of sine waves 344
17.10 Temperature signal from two sine waves 345
17.11 Square wave centered around the origin 346

List of Tables

12.1 Polymer particle volume fraction of binary collidal glasses 173
12.2 Specific heat capacity, density, and thermal conductivity 183

13.1 MD simulation interaction parameters 206
13.2 Specific heat capacity, density, and thermal conductivity of PS 213

14.1 Overview of the structural and chemical characterization 243
14.2 In-plane thermal diffusivity values of Hec/PVP stack films 247
14.3 Summary of the photoacoustic measurements 250
14.4 Summary of elastic stiffness constants 256
14.5 Summary of engineering mechanical properties 257

15.1 Relevant parameters of flexible sandwich-like membranes 267

16.1 Summary of A_{Vis} and A_{IR} values of PMMA and LDPE Films 299

17.1 Optimized parameters for a multiplexed excitation 336
17.2 Comparison of measured thermal diffusivities 337
17.3 DC reduction with multiple sine waves 345

1 Abstract

Time is the most valuable resource we have. Most time in a physicochemical laboratory is spent measuring a variety of quantities: voltages, extinction coefficients, temperatures, heat capacities, thermal diffusivity, density, and particle size to name but a few. Another time consuming task is data analysis. Certain analysis tools, like the radial distribution function are impossible to compute by hand due to the sheer amount of data.

During my PhD, I developed automated measurement and data analysis tools and applied them to specific research questions. These tools allowed my colleagues and me to gain a better understanding of nanostructured particulate systems. Furthermore, we could deduce structure-property relationships that govern heat flow in particulate matter. Most of the tools I created come in the form of software. My programming languages of choice are Python 3 and C++. They allow both rapid and object oriented software development.

My research focused on two main topics: heat transport and particulate systems. Both topics are intertwined in the diverse projects I completed during my PhD. Each project had its own, unique challenges that I addressed with programming. I will use the following paragraphs to summarize important results from each project.

Monodisperse, spherical, particles can self-assemble into hexagonal monolayers at the air-water interface. We discovered a new way to create non-close packed colloidal monolayers with all possible 2D Bravais lattices (chapter 10). My particle analysis tools were instrumental in the discovery of this new method. I developed a theoretical framework that allows us to predict and understand the lattice transformation.

While developing automated particle analysis, I solved a 35 year old problem. No efficient way was known for computing the 3D radial distribution function, $g(r)$, in finite volumes from particle coordinates. The radial distribution function is a powerful tool to investigate the structure of particulate systems. My new, analytic algorithm will help other researchers analyze particle distributions in finite volumes more efficiently (chapter 11). Finite particulate systems often occur during the analysis of colloidal materials.

Colloidal systems are an ideal model system to study heat transport in particulate materials. Therefore, we used colloidal glasses to study the effect of two parameters: the size distribution and the particle material. We reproduce our experimental results with COMSOL simulations. These simulations require a well controlled 3D model of the colloidal structure. I developed software that transforms particle coordinates from molecular dynamic simulations into a 3D model for COMSOL. These simulations facilitated our discovery of percolation pathways in binary colloidal assemblies (chapter 12). We also found that colloidal glasses, created by mixing particles of different size, have a reduced thermal conductivity due to tortuous thermal pathways (chapter 13). These discoveries provide new design guidelines for particulate thermal insulation materials.

I extended our thermal characterization toolbox by implementing the lock-in thermography technique. Automating the data analysis with software was important to speed up the data analysis and to create reproducible results. Lock-in thermography is ideally suited to characterize freestanding thin films. We applied lock-in thermography to characterize two different materials: highly anisotropic Bragg stacks made from Hectorite and polyvinylpyrolidone (PVP) and sandwich membranes consisting of electrospun polyurethane and silver nanowires. We discovered a record high thermal anisotropy in the layered Hectorite / PVP samples. Furthermore, we were able to correlate, for the first time, strong thermal anisotropy with the Young's and sheer moduli (chapter 14). Lock-in thermography also allowed us to probe fiber alignment and island formation during stretching of the sandwich membranes. I obtained the 2D order parameter for the fibers through image analysis. The implementation of a line laser even allowed measuring anisotropic thermal transport within the plane of the non-woven material (chapter 15).

Since lock-in thermography proved to be an invaluable tool for our research, we extended lock-in thermography in two directions. We showed that lock-in thermography can be applied to thermally thin films that are translucent for both visible and infrared radiation. Thermal losses, either by radiation or convection, have a negligible effect on the final results (chapter 16). In addition, I improved the excitation signal used for lock-in thermography. I designed two optimized multi frequency excitation signals with minimal DC heating: a square wave with reduced duty cycle and a sum of sine waves. Reducing DC heating is important to reduce distortions created by temperature dependent thermal properties. Custom analysis software was essential for measuring the reduced DC heating and optimizing the excitation signals. These signals allow us to measure at multiple frequencies with optimal signal efficiency (chapter 17). We think that these multi frequency signals can be adapted to a wide range of frequency domain methods.

Zusammenfassung

Zeit ist die wertvollste Ressource, die wir haben. Die meiste Zeit in einem physikalisch-chemischen Labor wird damit verbracht, eine Vielzahl von Größen zu messen: Spannungen, Extinktionskoeffizienten, Temperaturen, Wärmekapazitäten, Temperaturleitfähigkeit, Dichte und Partikelgröße, um nur einige zu nennen. Eine weitere zeitaufwändige Aufgabe ist die Datenanalyse. Bestimmte Analysewerkzeuge, wie die radiale Verteilungsfunktion, sind aufgrund der schieren Datenmenge nicht manuell berechenbar.

Während meiner Promotion entwickelte ich automatisierte Mess- und Datenanalyse Werkzeuge und wandte sie auf spezifische Forschungsfragen an. Diese Werkzeuge ermöglichten es meinen Kollegen und mir ein besseres Verständnis für nanostrukturierte Partikelsysteme zu erlangen. Darüber hinaus konnten wir Struktur-Eigenschaftsbeziehungen ableiten, die den Wärmefluss in partikulären Systemen bestimmen. Die meisten meiner Werkzeuge basieren auf Software. Meine bevorzugten Programmiersprachen sind Python 3 und C++. Sie ermöglichen sowohl eine schnelle als auch eine objektorientierte Softwareentwicklung.

Meine Forschung konzentrierte sich auf zwei Hauptthemen: Wärmetransport und Partikelsysteme. Beide Themen sind in den verschiedenen Projekten, die ich während meiner Doktorarbeit abgeschlossen habe, miteinander verknüpft. Jedes Projekt hatte seine eigenen, einzigartigen Herausforderungen, die ich mittels selbstgeschriebener Programme gelöst habe. In den folgenden Abschnitten werde ich wichtige Ergebnisse aus jedem Projekt zusammenfassen.

Monodisperse, kugelförmige Partikel können sich an der Luft-Wasser-Grenzfläche von selbst zu hexagonalen Monolagen zusammenfügen. Wir haben einen neuen Weg gefunden, nicht dichtgepackte kolloidale Monolagen mit allen möglichen 2D Bravais-Gittern zu erzeugen (Kapitel 10). Durch meine Partikelanalysewerkzeuge hab ich maßgeblich zur Entwicklung dieser neuen Methode beigetragen. Ich habe einen theoretisches Modell entwickelt, das es uns ermöglicht, die Gitterumwandlung vorherzusagen und zu verstehen.

Bei der Entwicklung einer automatisierten Partikelanalyse habe ich ein 35 Jahre altes Problem gelöst. Kein effizienter Weg war bekannt, um die 3D radiale Verteilungsfunktion $g(r)$ in endlichen Volumina aus Partikelkoordinaten zu berechnen. Die radiale Verteilungsfunktion ist ein leistungsfähiges Werkzeug zur Untersuchung der inneren Struktur von partikulären Systemen. Mein neuer, analytischer Algorithmus wird anderen Forschern helfen, Partikelverteilungen in endlichen Volumina effizienter zu analysieren (Kapitel 11). Partikuläre Systeme in endlichen Volumina treten häufig bei der Analyse von kolloidalen Materialien auf.

Kolloidale Systeme sind ein ideales Modellsystem zur Untersuchung des Wärmetransports in partikulären Materialien. Wir verwendeten deshalb kolloidale Gläser um den Einfluss von zwei Parametern zu untersuchen: die Partikelgrößenverteilung und das Partikelmaterial. Wir reproduzieren unsere experimentellen Ergebnisse mit COMSOL Simulationen. Diese Simulationen erfordern ein gut kontrolliertes 3D-Modell der kolloidalen Struktur. Ich habe eine Software entwickelt, die Partikelkoordinaten aus molekulardynamischen Simulationen in ein 3D-Modell für COMSOL umwandelt. Diese Simulationen ermöglichten uns die Entdeckung von Perkolationspfaden in binären kolloidalen Materialien (Kapitel 12). Wir haben auch festgestellt, dass kolloidale Gläser, die durch das Mischen von Partikeln unterschiedlicher Größe entstehen, eine geringere Wärmeleitfähigkeit aufgrund von gewundenen Wärmewegen aufweisen (Kapitel 13). Diese Entdeckungen bieten neue Designrichtlinien für partikuläre Wärmedämmstoffe.

Ich habe unseren Werkzeugkasten zur thermischen Charakterisierung durch den Aufbau der Lock-in Thermografie erweitert. Die Automatisierung der Datenanalyse mittels Software war wichtig, um die Datenanalyse zu beschleunigen und reproduzierbare Ergebnisse zu erzielen. Die Lock-in Thermografie eignet sich hervorragend zur Charakterisierung von freistehenden dünnen Schichten. Wir haben die Lock-in-Thermografie eingesetzt, um zwei verschiedene Materialien zu charakterisieren: hochanisotrope Bragg-Stacks aus Hectorit und Polyvinylpyrolidon (PVP) und Sandwichmembranen aus elektrogesponnenen Polyurethan- und Silber-Nanofasern. Wir entdeckten eine rekordverdächtig hohe thermische Anisotropie in den geschichteten Hectrorit / PVP-Proben. Darüber hinaus konnten wir erstmals eine starke thermische Anisotropie mit den Elastizitäts und Schub Modul korrelieren (Kapitel 14). Die Lock-in Thermografie ermöglichte es uns auch, die Faserausrichtung und Inselbildung durch Verstreckung zu untersuchen. Ich habe den 2D-Ordnungsparameter für die Fasern durch Bildanalyse erhalten. Die Implementierung eines Linienlasers ermöglichte sogar die Messung des anisotropen Wärmetransports innerhalb der Ebene des Vliesstoffes (Kapitel 15).

Da sich die Lock-in Thermografie als wertvolles Werkzeug für unsere Forschung erwiesen hat, haben wir die Lock-in Thermografie in zwei Richtungen erweitert. Wir haben gezeigt, dass die Lock-in Thermografie auf thermisch dünne Schichten angewendet werden kann, die sowohl für sichtbares Licht als auch für Infrarotstrahlung durchlässig sind. Wärmeverluste, zum Beispiel durch Strahlung oder Konvektion, haben einen vernachlässigbaren Einfluss auf das Endergebnis (Kapitel 16). Darüber hinaus habe ich das Anregungssignal für die Lock-in Thermografie verbessert. Ich habe zwei optimierte multifrequenz Anregungssignale mit minimaler Gleichstromerwärmung entwickelt: eine Rechteckwelle mit reduzierter Einschaltdauer und eine Summe von Sinuswellen. Die Reduzierung der Gleichstromheizung ist wichtig, um Messfehler zu reduzieren, die durch temperaturabhängige thermische Eigenschaften entstehen. Selbst entwickelte Analysesoftware war unerlässlich, um die reduzierte Gleichstromerwärmung zu messen und die Anregungssignale zu optimieren. Diese Signale ermöglichen es uns, bei mehreren Frequenzen mit optimaler Signalwirkung zu messen (Kapitel 17). Wir sind der Meinung, dass diese Mehrfrequenzsignale an eine breite Palette von Methoden angepasst werden können die in der Frequenzdomäne arbeiten.

2 List of Publications

My thesis is based on the following publications in peer-reviewed journals:

1. Qiang Gao, **Bernd A. F. Kopera**, Jian Zhu, Xiaojian Liao, Chao Gao, Markus Retsch, Seema Agarwal, Andreas Greiner, „Breathable and flexible polymer membranes with mechanoresponsive electric resistance“, *Advanced Functional Materials*, **accepted**, doi: 10.1002/adfm.201907555, **2019**; (see chapter 15)

2. Zuyuan Wang, Konrad Rolle, Theresa Schilling, Patrick Hummel, Alexandra Philipp, **Bernd A. F. Kopera**, Anna Lechner, Markus Retsch, Josef Breu, George Fytas, „Tunable thermoelastic anisotropy in hybrid Bragg stacks with extreme polymer confinement“, *Angewandte Chemie Int. Ed.*, **accepted**, doi: 10.1002/ange.201911546, **2019**; (see chapter 14)

3. Alexandra Philipp, Nelson W. Pech-May, **Bernd A. F. Kopera**, Anna Lechner, Sabine Rosenfeldt, Markus Retsch, „Direct measurement of the in-plane thermal diffusivity of semitransparent polymers by lock-in thermography: an extension of the slopes method“, *Analytical Chemistry*, vol. 91, no. 13, pp. 8476-8483, **2019**; (see chapter 16)

4. Miriam E. J. Hummel, Christian Stelling, **Bernd A. F. Kopera**, Fabian A. Nutz, Matthias Karg, Markus Retsch, and Stephan Förster, „Ordered Particle Arrays via a Langmuir Transfer Process: Access to Any Two-Dimensional Bravais Lattice“, *Langmuir*, vol. 35, no. 4, pp. 973-979, **2019**; (see chapter 10)

5. **Bernd A. F. Kopera** and Markus Retsch, “Computing the 3D Radial Distribution Function from Particle Positions: An Advanced Analytic Approach“, *Analytical Chemistry*, vol. 90, no. 23, pp. 13909-13914, **2018**; (see chapter 11)

6. Fabian A. Nutz, Alexandra Philipp, **Bernd A. F. Kopera**, Martin Dulle, Markus Retsch, „Low Thermal Conductivity through Dense Particle Packings with Optimum Disorder“, *Advanced Materials*, vol. 30, p. 1704910, **2018**; (see chapter 13)

7. Pia Ruckdeschel, Alexandra Philipp, **Bernd A. F. Kopera**, Flora Bitterlich, Martin Dulle, Nelson W. Pech-May, and Markus Retsch, „Thermal transport in binary colloidal glasses: Composition dependence and percolation assessment“, *Physical Review E*, vol. 97, p. 022612, **2018**; (see chapter 12)

My thesis also includes the following manuscript:

8. **Bernd A. F. Kopera** and Markus Retsch, „Reducing DC heating in lock-in thermography: the advantage of thermal multiplexing“, *in preparation*; (see chapter 17)

My results from the following, peer-reviewed publications are not included:

9. Mokit Chau and **Bernd A. F. Kopera**, V. R. Machado, S. M. Tehrani, M. A. Winnik, E. Kumacheva, and M. Retsch, „Reversible transition between isotropic and anisotropic thermal transport in elastic polyurethane foams“, *Materials Horizons*, vol. 4, no. 10, pp. 236-241, **2017**

10. Mokit Chau, Kevin J. De France, **Bernd Kopera**, Vanessa R. Machado, Sabine Rosenfeldt, Laura Reyes, Katelyn J. W. Chan, Stephan Förster, Emily D. Cranston, Todd Hoare, and Eugenia Kumacheva, „Composite Hydrogels with Tunable Anisotropic Morphologies and Mechanical Properties“, *Chemistry of Materials*, vol. 28, no. 10, pp. 3406-3415, **2016**

11. Ana Querejeta-Fernández, **Bernd Kopera**, Karen S. Prado, Anna Klinkova, Myriam Methot, Grégory Chauve, Jean Bouchard, Amr S. Helmy, and Eugenia Kumacheva, „Circular Dichroism of Chiral Nematic Films of Cellulose Nanocrystals Loaded with Plasmonic Nanoparticles“, *ACS Nano*, vol. 9, no. 10, pp. 10377-10385, **2015**

My results from the following manuscript are not included:

12. Anna M. Lechner, Tanja Feller, Lukas Heindl, Sabine Rosenfeldt, Markus Drechsler, **Bernd A. F. Kopera**, Markus Retsch, „Scalable synthesis of smooth PS@TiO_2 core-shell and TiO_2 hollow spheres in the (sub) micron size range: understanding synthesis and calcination parameters“, *submitted*, **2019**

3 List of Contributions

During my PhD, I made the following contributions to scientific conferences:

1. Oral presentation: *Computing the 3D Radial Distribution Function from Particle Positions: An Advanced Analytic Approach*, **2019**, DPG Spring Meeting of the Condensed Matter Section, Regensburg (DE)

2. Poster presentation: *Computing the 3D Radial Distribution Function from Particle Coordinates*, **2018**, Particle-Based Materials Symposium, Erlangen (DE), **Prize for the best poster presentation**

3. Poster presentation: *Solution-based Fabrication of Ordered Particle Arrays for All Two-Dimensional Bravais Lattices*, **2017**, MRS Conferrence, Boston (USA)

4. Poster presentation: *Reversible Transition between Anisotropic and Isotropic Thermal Transport in Polyurethane Foams*, **2017**, MRS Conferrence, Boston (USA)

5. Poster presentation: *Reversible Transition between Anisotropic and Isotropic Thermal Transport in Polyurethane Foams*, **2017**, Bayreuther Polymer Symposium, Bayreuth (DE)

6. Poster presentation: *Reversible Transition between Anisotropic and Isotropic Thermal Transport in Polyurethane Foams*, **2017**, Macromolecular Colloqium, Freiburg (DE), **Prize for the best poster presentation**

7. Poster presentation: *Reversible Transition between Anisotropic and Isotropic Thermal Transport in Polyurethane Foams*, **2016**, Biennial Meeting of the GDCh-Division Macromolecular Chemistry, Halle a. d. Saale (DE)

4 Acknowledgments

Before I present the main part of my thesis, I would like to thank all the people that supported me on my journey here.

First and foremost, I would like to thank my *Doktorvater* Prof. Dr. Markus Retsch for his unwavering support and unprecedented leadership. He granted me the academic freedom to investigate new and unexpected discoveries whenever they happened. It was my pleasure and a privilege to work with him. I also owe him a lot of gratitude for facilitating my three visits to the Georgia Institute of Technology.

I would also like to thank Prof. Shannon Yee Ph.D. my supervisor during my visits to the Georgia Institute of Technology. He openly shared the expertise of his group on heat transport characterization with me. In particular, I have to thank Sampath Kommandur and Misha Rodin for teaching me the 3ω and FDTR techniques in a hands on manner. I could experience an international work environment at the forefront of scientific research. Thank you for the wonderful time in Atlanta!

These visits would have been impossible without support from the German Academic Exchange Service (DAAD). Thank you for the opportunity to conduct research abroad.

I would like to thank all my colleques in the Retsch group. They created a supporting work environment and I enjoyed working with all of them on many challenging projects. I will keep the annual conference visits to Freiburg, our numerous group dinners, and challenging discussions in good memory.

I would like to thank the machine shop staff. My thesis would have been impossible without their masterful creation of components for my measurement setups.

Last but not least, my parents, who allowed me to experience the joy of chemistry from an early age and supported my academic career whenever possible.

Bragg reflections in colloidal superballs (Fig. 5.1a) were first explained in detail by Vogel *et al.*[14],[20] They created these superballs by evaporating emulsified droplets of colloidal dispersions. The frustrated colloidal self-assembly in a spherical confinement creates different crystalline domains. Each domain scatters light, wavelength dependent in different directions. Overall, the spherical symmetry creates uniform Bragg scattering in all directions. Vibrant and homogeneous structural colors are the result of the particle arrangement.

Barako *et al.*[15] showed that electrons in inverse metallic opals (Fig. 5.1b) move *ballistically* over small length scales. They used heat as a probe for electron movement. Heat is mainly carried by two types of particles in solid materials: electrons and phonons. Both particles move diffusively on a macroscopic scale due to scattering. The small features in the inverse opal, created by the highly ordered particle template, are of similar size as the electron mean free path. The ballistic scattering of electrons at the surfaces creates heat transport that differs from purely diffusive models.

Particle nested inverse opals (Fig. 5.1c) contain individual particles surrounded by a bicontinous layer. They are synthesized by infusing core-shell colloids with a third phase. A subsequent removal of the particle shells creates a bicountinous nest for the core particles. Particle nested inverse opals are a promising structural motive for battery electrodes.[21] The core particles could store ions and expand into the space around them. The continuous solid part in the bicountinous layer can act as the electrode for charge collection. The continuous pores create diffusion pathways for the ions in the electrolyte. These beneficial properties depend on the structure of the initial core-shell particles.

Hierarchical filter membranes with defined pore sizes (Fig. 5.1d) are accessible from colloidal monolayers by float-casting. These microsieves[17] have superior flux, size selectivity, and clogging resistance due to their uniform and thin pores. The pore size can be as small as 200 nm. Applications range from microsieve emulsification, to beverage filtration, and ultra fine aerosol generation for drug delivery. The uniformity of the membrane directly reflects the initial particle arrangement and size distribution.

Binary mixtures of colloidal particles (Fig. 5.1e) can self-assemble into lattices with quasi-crystalline order. Quasicrystals have long range rotational symmetry without translation symmetry. Photonic crystals with quasi-crystalline order exhibit band gaps with higher symmetry than classic crystals. The high symmetry band gaps minimize the optical losses in distributed feedback lasers built with quasi-crystalline materials.[22]

Orthogonally functionalized gold nanomeshes (Fig. 5.1f) are a promising approach to control the adsorption of colloidal particles at interfaces.[19] Electrostatic forces guide the charged colloids preferentially into the holes of the nanomesh. Individual particles can be selectively and reversible adsorbed into separate holes. The adsorption can be well controlled by the pH and the ionic strength of the dispersion. Analyzing the positions of the adsorbed colloids gives us a better understanding of the process.

Colloidal monolayers of equal particles usually self-assemble into an hexagonal monolayer. However, creating different lattice structures is beneficial for their application as nanolithography masks,[23] waveguides,[24] and plasmonic sensors.[25] These examples show that the function of a colloidal particulate system is strongly correlated with its structure. There are two key challenges during the structural analysis of particulate systems:

1. digitizing the particle positions, and
2. analyzing the particle coordinates.

A variety of techniques exist to determine particle coordinates at different length scales. They usually rely on 2D imaging and then extracting particle positions through image analysis. Particles are either placed at local intensity maxima[26] or at the geometric center of their boundary (e.g. the center of the boundary circle for spherical particles). Both methods rely on manual user input to optimize certain parameters like intensity thresholds or particle size and distance ranges. The resulting particle positions should always be checked by a human for correctness.

Common imaging techniques for 2D particulate materials are scanning electron microscopy,[19],[27]–[29] transmission electron microscopy,[18],[30], scanning tunneling microscopy,[31] atomic force microscopy,[32]–[35] and optical microscopy.[36],[37]

Three dimensional particle structures are imaged with tomography or confocal microscopy.[38],[39] Probe beams for tomography range from neutron radiation[40] to electron beams[41] and focused x-rays.[42] Both techniques have in common that they produce an image stack. Particle positions in the individual images are detected and then linked in 3D through the image stack to yield the 3D particle distribution.

The standard tool to analyze particle coordinates is the radial distribution function, $g(r)$. We can determine the crystal structure, average inter particle distances, domain sizes, and interaction potentials between particles from $g(r)$. Efficiently computing $g(r)$ in small, finite, volumes was an unsolved problem for over 35 years.[1] I used these tools to analyze particle assemblies made from colloids.

[1] see Larsen and Shaw[43] on page 4264, left column, fourth paragraph.

In addition to the particle arrangement, Figure 5.2 demonstrates that a lot of other parameters additionally influence the effective properties of a colloidal superstructure. These are size and shape of the individual particles (Fig. 5.2a), the adhesion between adjacent spheres and their contact area (Fig. 5.2b). The chemical composition of a colloidal particle can vary from polymeric to dielectric or even metallic (Fig. 5.2e), which can be further integrated into a heterogenous composition of colloidal assemblies (Fig. 5.2d). Finally, also the gas surrounding the polymer ensemble can have a strong influence (Fig. 5.2f) on the thermal properties.

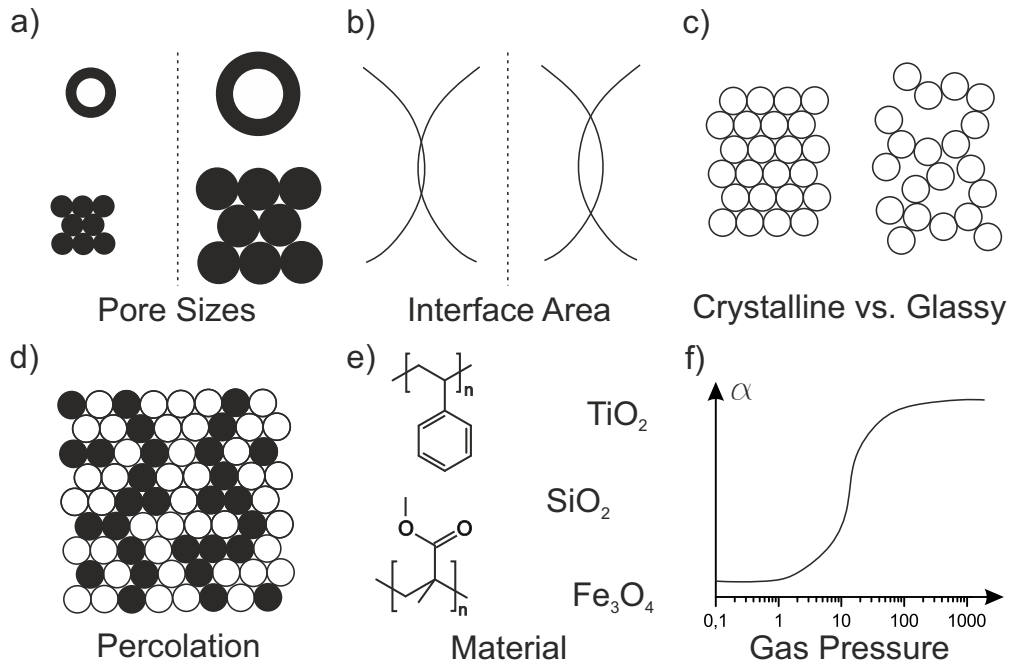


Fig. 5.2.: Different parameters that can be controlled in colloidal assemblies: a) internal and external pore sizes, b) interface area between particles, c) degree of order, d) percolation in binary systems, e) different organic and inorganic materials, f) gas pressure in pores changes the gas thermal diffusivity.

All of the above listed parameters render colloidal materials a very interesting and rich system to investigate heat transport in particulate systems. The colloidal assembly features translate into a variety of thermal transport phenomena, specifically: The pore size and type (internal vs. external) can be small enough to create the Knudsen effect.[44] The overlap between particles has a strong influence on heat transport.[45]–[47] The particle order changes both the density and the number of thermal pathways between particles.[48] Binary colloidal mixtures contain percolation pathways above a certain particle ratio.[49] Different colloidal materials have different intrinsic thermal conductivites.[50] Consequently, heat transport and heat transport characterization in nanostructured materials evolved as an important research topic during my PhD thesis.

5.3 Heat Transport

Thermal energy is present everywhere.[51],[52] We can neither confine it, nor fully transform it into another, more useful, type of energy.[53] Heat is thermal energy that moves from one object to another. The driving force is a temperature difference between the objects. Heat is transported by three different mechanisms: radiation, convection, and conduction.

My work focuses on heat flow in solid materials where conduction is the main transport mechanism. Two quantities are necessary to describe heat flux in solids: thermal conductivity and thermal diffusivity. Intuitively, the thermal conductivity determines how much heat flows between two objects in thermal equilibrium. The thermal diffusivity describes the temperature distribution in thermal equilibrium and how fast it is reached. Both quantities are necessary to describe heat transfer.

Guiding the flow of heat is essential for many practical applications. We can group these applications into two main categories:

- thermal management (efficiently removing heat) and,
- thermal insulation (preventing heat flux).

Thermal management deals with waste heat. The most prominent example are integrated circuits. Modern computer processors have a thermal design power of more than 100 W. This waste heat has to be removed through the packaging. Common surface areas are $52 \times 45\,\mathrm{mm}^2$. This is equivalent to a specific heating power of roughly $42\,\mathrm{kW/m^2}$, similar to the cooktop on an oven.

Most of the thermal resistance between the electronics and the heat sink originate from interfaces. Microscopic surface roughness at these interfaces prevents efficient heat transfer. The technical solution to this challenge are thermal interface materials. These soft materials, with a high thermal conductivity, fill the gaps and facilitate heat flux. Paste like metal alloys are commonly used today. They outperform dielectric thermal pastes based on silicones and ceramics like alumina.[54] However their electrical conductivity is a serious disadvantage. Spilling these metal pastes onto the surrounding chip can destroy the device. Recent approaches focus therefore on dielectric pastes. Suh and coworkers[55],[56] presented a soft thermal interface material with a thermal conductivity of up to $100\,\mathrm{W/mK}$, similar to solid metals. The high thermal conductivity is created by a high volume fraction of silver nanoflowers. They could demonstrate a significant reduction in the processor temperature of a smart phone.

Rechargeable batteries have seen tremendous improvement over the last decades.[57] The applications range from smart phones to power tools and electric vehicles. Managing the temperature of a battery pack is important to avoid thermal runaway[58] and increase battery lifetime.

The kinetics of chemical reactions inside a battery are also temperature dependent. Cars in a cold climate should be able to operate even when the battery is cold. Heating the battery with its own electric energy is inefficient. Active battery management is complicated and consumes energy.[59] A better approach is to retain the waste heat, created by the cell reactions during start up. This requires an adaptive thermal insulation with a reversible and temperature dependent thermal conductivity. Dames and coworkers[60] designed a passive interface thermal regulator that fulfills these requirements. Their regulator is based on a shape memory alloy. Thermal conduction pathways are created above a certain temperature. Excess heat is then efficiently removed once the battery has reached its operating temperature.

Thermal insulation is also important for energy efficient homes. My shared student apartment, built in the 1920's, needs 232 $^{\mathrm{kWh}}/_{\mathrm{m}^2\cdot\,\mathrm{year}}$ for heating in the form of natural gas.[2] Heating accounts for 70 % of the energy consumption in homes.[61] Newly built homes in Germany are required to have good thermal insulation.[62] Proper thermal insulation reduces the heating requirements of a home to 15 $^{\mathrm{kWh}}/_{\mathrm{m}^2\cdot\,\mathrm{year}}$.[63]

Common insulation materials are mineral wool and expanded polystyrene foam. Both have thermal conductivities in the range from 36 to 32 $^{\mathrm{mW}}/_{\mathrm{mK}}$ close to the value of still air (26.2 $^{\mathrm{mW}}/_{\mathrm{mK}}$).[64] There are two ways to reduce the thermal conductivity further. One can remove the air and create vacuum insulation panels with a very low thermal conductivity of $< 5\,^{\mathrm{mW}}/_{\mathrm{mK}}$. However, these panels loose their superior thermal insulation properties when they are punctured.[65]

Another way is to reduce the pore size of foams while keeping the foam density constant. This reduces the intrinsic thermal conductivity of the gas in the pores via the Knudsen effect.[44] Small pores limit the mean free path of the gas particles. This creates a ballistic transport regime where the mean free path is determined by the pore size and not the gas properties. The reduction in the mean free path, relative to the free gas reduces the thermal conductivity of the gas phase. Air has a mean free path of around 65 nm.[66] Creating foams with small enough pores that show the Knudsen effect remains challenging.[67],[68]

[2] According to my energy bill in 2018.

Thermal insulation using clothes is an important part of our everyday lives. More humans die from cold weather than hot weather.[69] Clothing enables us to live in places that are otherwise too cold to survive.[3] Nature optimized insulating fur over millions of years. A good example are polar bears (*Ursus maritimus*).[70] Their white fur consists of hollow fibers. The trapped, non convecting air inside the fibers creates good thermal insulation. Bionic hollow polymer fibers adapt this principle.[71] They are widely used to create clothing with good thermal insulation. A prominent example are lightweight sleeping bags.[72] The encapsulated air retains the thermal insulation properties even when wet.

Improving the insulation of clothing requires knowledge about heat transport in fabrics and individual fibers. Lock-in thermography (see section 6.6) is intrinsically suited to measure heat transport in thin 2D materials and fibers. We use lock-in thermography to characterize thin films and nonwovens.

Lock-in thermography with a line laser was first described by Wolf *et al.*[73] They used a static line laser and their IR camera pixels were aligned with the heat flux. Simavilla *et al.*[74] extended lock-in thermography with a point source to anisotropic polymer films. Pradere *et al.*[75] used lock-in thermography to measure micron sized carbon and ceramic fibers. They were able to measure the thermal diffusivity of their fibers up to 2700 K. Mendioroz *et al.*[76] analyzed different experimental factors that influence the accuracy of lock-in thermography measurements. They focused on a simple analysis technique called *the slope method*.

Lock-in thermography has proven vital in the correct measurement of individual fibers. Huang *et al.*[77] investigated heat transport in spider silk fibers with a self built DC heating technique. They claimed that spider silk has an extraordinarily high thermal diffusivity around $10^{-4}\,\mathrm{m^2/s}$. Later work from Fuente *et al.*[78] with lock-in thermography revealed that spider silk only has an average thermal diffusivity of $2 \cdot 10^{-7}\,\mathrm{m^2/s}$, as expected for organic materials. The previously measured high thermal diffusivity was most likely a measurement artifact caused by heat losses. Heat losses strongly influence lock-in thermography measurements too,[79] especially for thin fibers.[78] Accurate measurements of fibers require vacuum.[80]

In summary, lock-in thermography is a mature tool to analyze heat transport in thin films and fibers. Artifacts from heat losses are well understood and can be avoided by measuring in vacuum.

[3] Like Bayreuth during winter.

Methods

6

Method development was a central part of my PhD. I used programming and mathematics to create new scientific tools. These tools are indispensable for our research. I will use the following sections to describe the tools I developed and used.

The first part describes tools to analyze and predict stretched colloidal monolayers. I present a simple approach to describe stretching of lattices. My method differs from the transfer matrix method used in our publication.[81] However, the results are identical. My new approach is advantageous, because it is simpler. The second part contains information about particle assembly generation and analysis. I will show how we generated complex 3D particle assemblies for heat transport simulations. Understanding the result from these simulations required an in depth look at next neighbors in a particulate system. Details on the computation of $g(r)$ are not included in this section. An explanation of both my new algorithm and the radial distribution function is in the SI of our publication (please see chapter 11). The last part is dedicated to heat transport characterization techniques. I first present a commercially available technique: Xenon flash analysis. I then present two custom built setups: the photoacoustic technique and lock-in thermography. We use these methods in the publications presented in chapters 12 through 17.

6.1 The Programming Language Python 3

The programming language Python[82] is a powerful, interpreted, scripting language that allows fast and efficient creation of programs. There are many active programmers that maintain a healthy ecosystem of add-ons for all sorts of endeavors. The matplotlib library,[83] NumPy,[84] and SciPy[85] are important extensions for scientists to name but a few. I use matplotlib in this work to visualize all data generated from the calculations and the post-processing routines. Python is intrinsically slower in execution than compiled languages like C++. This is not an issue for me. The faster development usually compensates the longer runtime. Overall, Python is a good choice for scientific programming because it is simple, concise, and the source code is easy to read.

6.2 Analyzing Stretched Hexagonal Lattices

I now derive the theoretical framework to describe uniaxial stretching of arbitrary, 2D lattices. I then use this framework to stretch hexagonal lattices. We can also predict the necessary stretch factor and direction to create a specific lattice from a hexagonal lattice. This formulation is different from the one in our publication (see Hummel *et al.*[81] or chapter 10) but yields identical results. I use vectors instead of matrices to describe stretching. Using only vectors and dot products is superior to a matrix based formulation. Dot products of vectors are commutative (changing the order preserves the results) and distributive (multiplying a sum is the same as summing the multiples).[86] These properties greatly simplify the manipulation of equations.

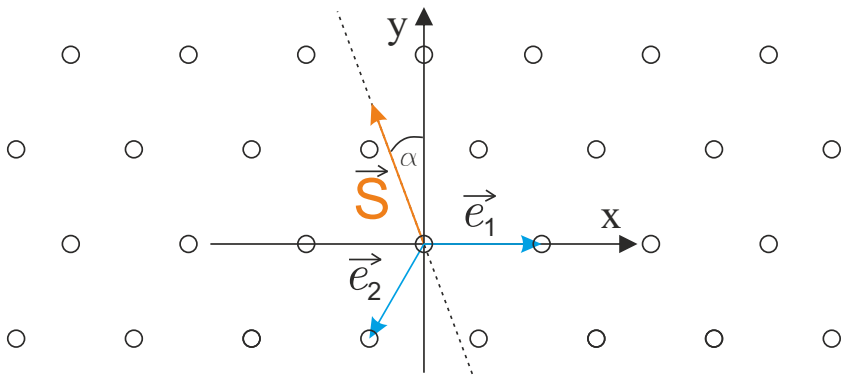


Fig. 6.1.: Coordinate system used in the following sections. The stretch vector, $\vec{S}$, (orange) encloses the angle, α, with the y-axis and is collinear with the stretch axis (black dashed line). The reduced primitive translation vectors, $\vec{e_1}$ and $\vec{e_2}$, define the original hexagonal lattice (black circles).

Figure 6.1 shows the orientation of the original hexagonal lattice (black circles) and the coordinate system I use. One pair of primitive translations vectors, $\vec{e}_1$, $\vec{e}_2$, that define the hexagonal lattice are

$$\vec{e_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \text{ and } \vec{e_2} = -\frac{1}{2} \cdot \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix}. \tag{6.1}$$

The stretch vector, $\vec{S}$, is collinear with the stretch axis and given by

$$\vec{S} = \begin{pmatrix} -S \cdot \sin(\alpha) \\ S \cdot \cos(\alpha) \end{pmatrix} = S \cdot \begin{pmatrix} -\sin(\alpha) \\ \cos(\alpha) \end{pmatrix} = S \cdot \vec{R}(\alpha). \tag{6.2}$$

S is the stretch factor and $\vec{R}(\alpha)$ the stretch direction. We choose α to start at the y-axis because the monolayer transfer happens along the vertical direction.

6.2.1 Stretching with a Stretch Vector

In this section, I introduce a description of uniaxial stretching in 2D using only the stretch vector, $\vec{S}$. I decompose the original particle position vector into two vectors. One vector is stretched and the other one stays the same.

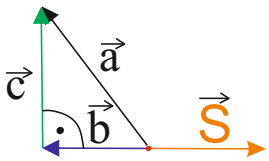


Fig. 6.2.: The stretch vector $\vec{S}$ (orange) and an arbitrary vector $\vec{a}$. We decompose the vector $\vec{a}$ into the vectors $\vec{b}$ and $\vec{c}$. The vector $\vec{b}$ is collinear to $\vec{S}$. The vector $\vec{c}$ is perpendicular to $\vec{S}$.

We decompose an arbitrary vector $\vec{a}$ into two orthogonal vectors, $\vec{b}$ and $\vec{c}$, according to Figure 6.2. We choose the two vectors such that

$$\vec{a} = \vec{b} + \vec{c}, \text{ and therefore, } \vec{c} = \vec{a} - \vec{b}. \tag{6.3}$$

The vector $\vec{b}$ is the projection of $\vec{a}$ onto $\vec{S}$ given by

$$\vec{b} = \left(\vec{a} \circ \vec{R}\right) \cdot \vec{R}. \tag{6.4}$$

I use the „∘“ operator to denote dot products. The advantage of this decomposition is that $\vec{c}$ is not stretched by $\vec{S}$. Only $\vec{b}$ is stretched. Stretching $\vec{b}$ is simple because $\vec{S}$ and $\vec{b}$ are collinear. We simply multiply $\vec{b}$ with the stretch factor S. Overall, the stretched vector of $\vec{a}$, called $\vec{a}'$, is given by

$$\vec{a}' = S \cdot \vec{b} + \vec{c}. \tag{6.5}$$

Next, we insert the equations for $\vec{b}$ (Eq. (6.4)) and $\vec{c}$ (Eq. (6.3)) and get

$$\vec{a}' = S \cdot \left(\vec{a} \circ \vec{R}\right) \cdot \vec{R} + \vec{a} - \vec{b}, \tag{6.6}$$

$$\vec{a}' = S \cdot \left(\vec{a} \circ \vec{R}\right) \cdot \vec{R} + \vec{a} - \left(\vec{a} \circ \vec{R}\right) \cdot \vec{R}. \tag{6.7}$$

Rearranging and factoring yields

$$\boxed{\vec{a}' = \vec{a} + (S-1) \cdot \left(\vec{a} \circ \vec{R}\right) \cdot \vec{R}}. \tag{6.8}$$

This is the final equation we use when we stretch a vector using only the stretch vector $\vec{S} = S \cdot \vec{R}$.

6.2.2 The Length of a Stretched Vector

I now derive the equation for the squared length of a stretched vector. We will use the squared length on the next page to identify square, hexagonal, and rectangular lattices.

We start with a well known equation from vector analysis.[86] The dot product of a vector with itself equals the square of its length:

$$\vec{a}' \circ \vec{a}' = |\vec{a}'|^2. \tag{6.9}$$

A square always means multiplication of numbers, not vectors. We insert the equation for a stretched vector (Eq. (6.8)), yielding

$$|\vec{a}'|^2 = \left[\vec{a} + (S-1) \cdot \left(\vec{a} \circ \vec{R}\right) \cdot \vec{R}\right] \circ \left[\vec{a} + (S-1) \cdot \left(\vec{a} \circ \vec{R}\right) \cdot \vec{R}\right]. \tag{6.10}$$

Next, we multiply the square brackets and combine similar terms into

$$|\vec{a}'|^2 = \vec{a} \circ \vec{a} + 2 \cdot (S-1) \cdot \left(\vec{a} \circ \vec{R}\right)^2 + (S-1)^2 \cdot \left(\vec{a} \circ \vec{R}\right)^2 \cdot \left(\vec{R} \circ \vec{R}\right). \tag{6.11}$$

Note, that $\vec{R}$ has unit length

$$\vec{R} \circ \vec{R} = |\vec{R}|^2 = 1^2 = 1, \tag{6.12}$$

which simplifies equation (6.11) to

$$|\vec{a}'|^2 = \vec{a} \circ \vec{a} + 2 \cdot (S-1) \cdot \left(\vec{a} \circ \vec{R}\right)^2 + (S-1)^2 \cdot \left(\vec{a} \circ \vec{R}\right)^2. \tag{6.13}$$

Again, we combine terms with equal factors

$$|\vec{a}'|^2 = \vec{a} \circ \vec{a} + \left[2 \cdot (S-1) + (S-1)^2\right] \cdot \left(\vec{a} \circ \vec{R}\right)^2, \tag{6.14}$$

and expand the terms in the square brackets into

$$|\vec{a}'|^2 = \vec{a} \circ \vec{a} + \left[2 \cdot S - 2 + S^2 - 2S + 1\right] \cdot \left(\vec{a} \circ \vec{R}\right)^2. \tag{6.15}$$

Simplifying the square brackets gives us the final equation for the squared length of a stretched vector

$$\boxed{|\vec{a}'|^2 = \vec{a} \circ \vec{a} + \left(S^2 - 1\right) \cdot \left(\vec{a} \circ \vec{R}\right)^2}. \tag{6.16}$$

We directly see that for $S = 1$, meaning we do not stretch, we get

$$|\vec{a}'|^2 = \vec{a} \circ \vec{a} = |\vec{a}|^2, \tag{6.17}$$

as we would expect.

6.2.3 The Phase Line for Lattices with Equal Length

In this section, I derive the equation for the phase line containing square, hexagonal, and rectangular lattices. These lattices have in common that their reduced primitive translation vectors have equal length. We need an equation that connects the stretching direction, α, with the stretch factor, S. We start by setting the lengths of the stretched primitive translation vectors equal

$$|\vec{e_1}'| \stackrel{!}{=} |\vec{e_2}'|. \tag{6.18}$$

Instead of using the absolute values, we proceed with their squares

$$|\vec{e_1}'|^2 = |\vec{e_2}'|^2. \tag{6.19}$$

Next, we use the corresponding equations (6.16) for the squared length

$$|\vec{e_1}'|^2 = \vec{e_1} \circ \vec{e_1} + (S^2 - 1) \cdot (\vec{e_1} \circ \vec{R})^2, \tag{6.20}$$

$$|\vec{e_2}'|^2 = \vec{e_2} \circ \vec{e_2} + (S^2 - 1) \cdot (\vec{e_2} \circ \vec{R})^2 \tag{6.21}$$

and insert them into equation (6.19) to get

$$\vec{e_1} \circ \vec{e_1} + (S^2 - 1) \cdot (\vec{e_1} \circ \vec{R})^2 = \vec{e_2} \circ \vec{e_2} + (S^2 - 1) \cdot (\vec{e_2} \circ \vec{R})^2. \tag{6.22}$$

We separate the terms with S on the right side

$$\vec{e_1} \circ \vec{e_1} - \vec{e_2} \circ \vec{e_2} = (S^2 - 1) \cdot (\vec{e_2} \circ \vec{R})^2 - (S^2 - 1) \cdot (\vec{e_1} \circ \vec{R})^2, \tag{6.23}$$

and factor out the term $(S^2 - 1)$ yielding

$$\vec{e_1} \circ \vec{e_1} - \vec{e_2} \circ \vec{e_2} = (S^2 - 1) \cdot \left[(\vec{e_2} \circ \vec{R})^2 - (\vec{e_1} \circ \vec{R})^2 \right]. \tag{6.24}$$

Again, we separate the term with S on one side

$$S^2 - 1 = \frac{\vec{e_1} \circ \vec{e_1} - \vec{e_2} \circ \vec{e_2}}{(\vec{e_2} \circ \vec{R})^2 - (\vec{e_1} \circ \vec{R})^2}, \tag{6.25}$$

and solve for S, assuming $S > 0$, to get

$$\boxed{S(\alpha) = \sqrt{1 + \frac{\vec{e_1} \circ \vec{e_1} - \vec{e_2} \circ \vec{e_2}}{\left[\vec{e_2} \circ \vec{R}(\alpha)\right]^2 - \left[\vec{e_1} \circ \vec{R}(\alpha)\right]^2}}}. \tag{6.26}$$

The equation for $S(\alpha)$ depends on the specific primitive lattice vectors, $\vec{e_1}$ and $\vec{e_2}$. All phase lines are accessible by inserting different primitive lattice vectors for a hexagonal lattice.

As an example, we compute the phase line from the primitive translation vectors given in section 6.2. These vectors are defined by

$$\vec{e_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \text{ and } \vec{e_2} = -\frac{1}{2} \cdot \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix}. \tag{6.27}$$

We insert $\vec{e_1}$, $\vec{e_2}$, and $\vec{R}$ into equation (6.26) and get

$$S(\alpha) = \sqrt{1 + \frac{\begin{pmatrix} 1 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \left(-\frac{1}{2}\right) \cdot \left(-\frac{1}{2}\right) \cdot \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix} \circ \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix}}{\left[-\frac{1}{2} \cdot \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix} \circ \begin{pmatrix} -\sin(\alpha) \\ \cos(\alpha) \end{pmatrix}\right]^2 - \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \circ \begin{pmatrix} -\sin(\alpha) \\ \cos(\alpha) \end{pmatrix}\right]^2}}. \tag{6.28}$$

We then multiply the dot products, yielding

$$S(\alpha) = \sqrt{1 + \frac{1 - \frac{1}{4} \cdot (1+3)}{\left(-\frac{1}{2} \cdot \left[\sqrt{3} \cdot \cos(\alpha) - \sin(\alpha)\right]\right)^2 - [\sin(\alpha)]^2}}. \tag{6.29}$$

The numerator, $1 - \frac{1}{4} \cdot (1+3)$, simplifies to 0, leaving us with

$$\boxed{S(\alpha) = \sqrt{1} = 1}. \tag{6.30}$$

Figure 6.3 shows the resulting phase line $S(\alpha)$ embedded into the continuous phase diagram for the length ratio between reduced primitive lattice vectors.

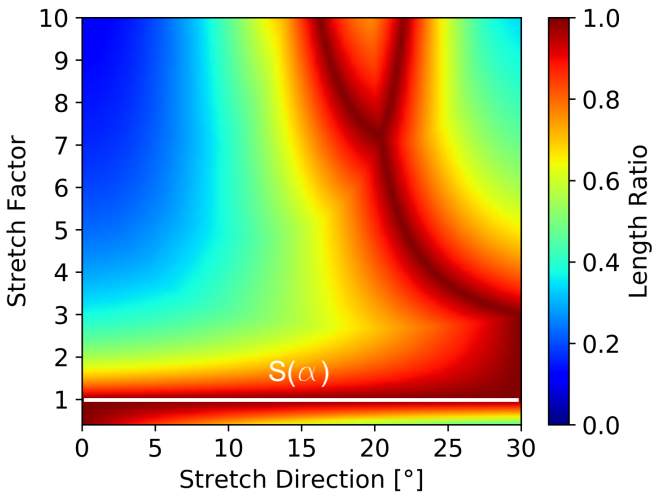


Fig. 6.3.: Phase diagram for the length ratio between stretched primitive lattice vectors. The original lattice was close packed hexagonal. The white line represents the phase line for equal length given by equation (6.30). Other phase lines for equal length are accessible by using other primitive translation vectors for a hexagonal lattice.

6.2.4 The Phase Lines with Orthogonal Lattices

A second set of phase lines contain square and rectangular lattices. These lattices have in common that their reduced primitive translation vectors are orthogonal. We look for an equation between the stretching direction, α, and the stretch factor, S. We want the stretched primitive translation vectors, $\vec{e_1}'$ and $\vec{e_2}'$, to be orthogonal. Therefore, their dot product has to be zero

$$\vec{e_1}' \circ \vec{e_2}' \stackrel{!}{=} |\vec{e_1}'| \cdot |\vec{e_2}'| \cdot \cos(90°) = |\vec{e_1}'| \cdot |\vec{e_2}'| \cdot 0 = 0. \tag{6.31}$$

Next, we insert the equations (6.8) for the stretched basis vectors

$$\left[\vec{e_1} + (S-1) \cdot \left(\vec{e_1} \circ \vec{R}\right) \cdot \vec{R}\right] \circ \left[\vec{e_2} + (S-1) \cdot \left(\vec{e_2} \circ \vec{R}\right) \cdot \vec{R}\right] = 0 \tag{6.32}$$

and expand the dot product in the middle into

$$\vec{e_1} \circ \vec{e_2} + 2 \cdot (S-1) \left(\vec{e_1} \circ \vec{R}\right) \left(\vec{e_2} \circ \vec{R}\right) + (S-1)^2 \left(\vec{e_1} \circ \vec{R}\right) \left(\vec{e_2} \circ \vec{R}\right) \left(\vec{R} \circ \vec{R}\right) = 0. \tag{6.33}$$

Again, we use the fact that $\vec{R}$ has unit length which simplifies equation (6.33) to

$$\vec{e_1} \circ \vec{e_2} + 2 \cdot (S-1) \cdot \left(\vec{e_1} \circ \vec{R}\right) \cdot \left(\vec{e_2} \circ \vec{R}\right) + (S-1)^2 \cdot \left(\vec{e_1} \circ \vec{R}\right) \cdot \left(\vec{e_2} \circ \vec{R}\right) = 0. \tag{6.34}$$

We combine terms with equal factors

$$\vec{e_1} \circ \vec{e_2} + \left[2 \cdot (S-1) + (S-1)^2\right] \cdot \left(\vec{e_1} \circ \vec{R}\right) \cdot \left(\vec{e_2} \circ \vec{R}\right) = 0, \tag{6.35}$$

and simplify the square bracket into

$$\vec{e_1} \circ \vec{e_2} + \left[2 \cdot S - 2 + S^2 - 2 \cdot S + 1\right] \cdot \left(\vec{e_1} \circ \vec{R}\right) \cdot \left(\vec{e_2} \circ \vec{R}\right) = 0, \tag{6.36}$$

$$\vec{e_1} \circ \vec{e_2} + \left[S^2 - 1\right] \cdot \left(\vec{e_1} \circ \vec{R}\right) \cdot \left(\vec{e_2} \circ \vec{R}\right) = 0. \tag{6.37}$$

Finally, we solve for S by separating all terms containing S on the left side

$$S^2 - 1 = \frac{-\vec{e_1} \circ \vec{e_2}}{\left(\vec{e_1} \circ \vec{R}\right) \cdot \left(\vec{e_2} \circ \vec{R}\right)}. \tag{6.38}$$

The stretch factor S has to be positive, which leaves us with

$$\boxed{S(\alpha) = \sqrt{1 - \frac{\vec{e_1} \circ \vec{e_2}}{\left[\vec{e_1} \circ \vec{R(\alpha)}\right] \cdot \left[\vec{e_2} \circ \vec{R(\alpha)}\right]}}}. \tag{6.39}$$

The equation for $S(\alpha)$ depends on the specific basis vectors $\vec{e_1}$ and $\vec{e_2}$. All orthogonal phase lines are accessible by using a different primitive basis.

As an example, we compute the phase line from the primitive basis given in section 6.2. The primitive translation vectors are

$$\vec{e_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \text{ and } \vec{e_2} = -\frac{1}{2} \cdot \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix}. \tag{6.40}$$

We insert $\vec{e_1}$, $\vec{e_2}$, and $\vec{R}$ into equation (6.39) and get

$$S(\alpha) = \sqrt{1 - \frac{-\frac{1}{2} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix}}{\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \circ \begin{pmatrix} -\sin(\alpha) \\ \cos(\alpha) \end{pmatrix}\right] \cdot \left[-\frac{1}{2} \cdot \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix} \circ \begin{pmatrix} -\sin(\alpha) \\ \cos(\alpha) \end{pmatrix}\right]}}. \tag{6.41}$$

Solving the dot products yields

$$S(\alpha) = \sqrt{1 + \frac{-\frac{1}{2}}{\sin(\alpha) \cdot \left(-\frac{1}{2} \cdot \left[\sqrt{3}\cos(\alpha) - \sin(\alpha)\right]\right)}}. \tag{6.42}$$

Finally, we cancel the factor $-1/2$ and obtain

$$\boxed{S(\alpha) = \sqrt{1 + \frac{1}{\sin(\alpha) \cdot \left[\sqrt{3}\cos(\alpha) - \sin(\alpha)\right]}}}. \tag{6.43}$$

Figure 6.4 shows the resulting phase line $S(\alpha)$ embedded into the continuous phase diagram for the angle between reduced primitive translation vectors.

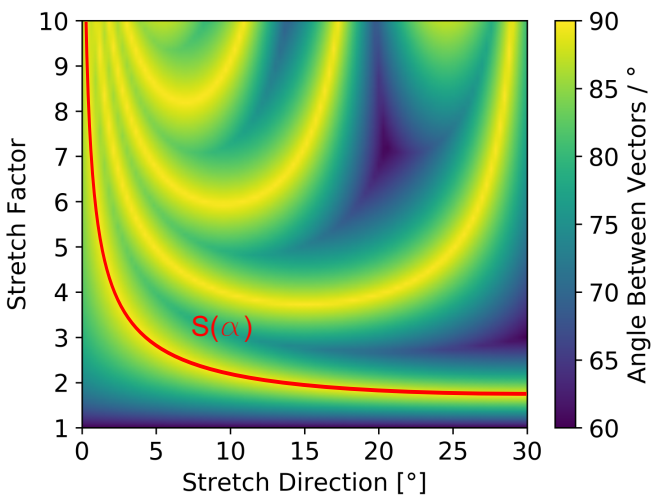


Fig. 6.4.: Phase diagram for the angle between reduced primitive translation vectors. The original lattice was hexagonal. The red line indicates the phase line for orthogonal lattices described by equation (6.43).

6.2.5 Reduced Primitive Basis from a Stretched Basis

I now present an algorithm that computes the *reduced* primitive basis of a stretched 2D lattice. This algorithm was first proposed by Lagrange and Gauss.[87],[88] I use this algorithm to compute the phase diagrams of a stretched hexagonal lattice. The challenge is, that stretching the reduced primitive basis does not necessarily yield the reduced primitive basis of the stretched lattice. We need the reduced primitive basis to determine the Bravais type of the stretched lattice. The angle and length ratio between the reduced primitive translation vectors determine the Bravais type.

A reduced basis consists of primitive translation vectors with minimal length. The angle between the reduced basis vectors in 2D is always between 60 and 120 degree. Due to symmetry in 2D, we can limit the range to either [60 to 90] or [90 to 120] degree. We choose the range [60 to 90] degree in our publication.[81]

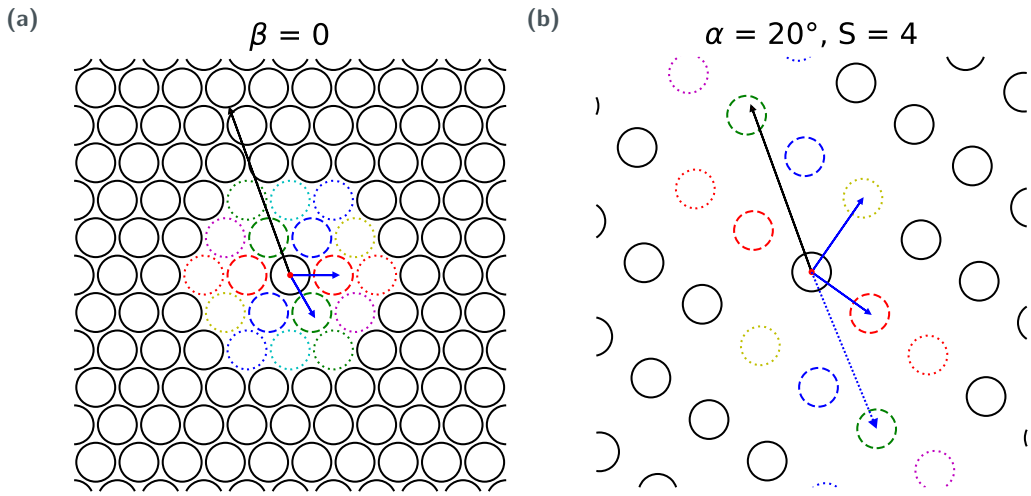


Fig. 6.5.: a) Close packed hexagonal lattice with color coded particles in the first and second coordinations sphere around the center. b) Resulting non-close packed lattice when the lattice in a) is stretched in the direction and amount defined by the black vector. Note that the primitive lattice vectors point to *different* particles. The dotted blue vector, created by stretching one of the primitive lattice vectors in a) is no longer a reduced primitive lattice vector.

Figure 6.5 illustrates the change of the reduced primitive basis. The unstretched primitive basis vectors (Figure 6.5a) point to a red and green dashed particle. In contrast, the reduced primitive basis vectors of the stretched lattice (Figure 6.5b) point to a red dashed and yellow dotted particle. The dotted blue vector is no longer a reduced primitive translation vector.

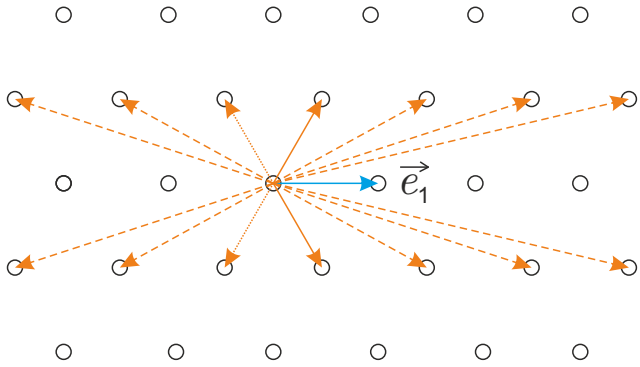


Fig. 6.6.: Primitive lattice vectors for a hexagonal lattice. Every combination of the blue and an orange vector is a primitive base. However, only the solid and dotted orange vectors form a *reduced* primitive base with the blue vector. In this work, we further limit the angle between reduced primitive grid vectors to [60°; 90°].

To derive the lattice vector reduction algorithm we look at special properties of lattices. A lattice has infinitely many primitive lattice vectors that form a primitive base as shown in Figure 6.6. Subtracting an integer multiple of a primitive lattice vector from its peer yields a new primitive lattice vector

$$\vec{e_3} = \vec{e_1} + k \cdot \vec{e_2}, \text{ for all } k \text{ in } \mathbb{Z}. \tag{6.44}$$

The idea is to repeatedly generate shorter lattice vectors by subtracting an integer multiple of one from another. The apparently optimal reduction factor, μ, creates a new lattice vector with minimal length.

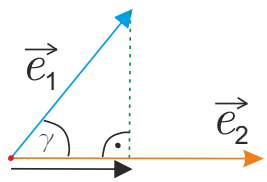


Fig. 6.7.: Projection of vector $\vec{e_1}$ onto the vector $\vec{e_2}$. The length ratio between the orange and blue vector is the optimal reduction factor, μ.

We obtain the optimal factor by projecting one lattice vector ($\vec{e_1}$) onto the other ($\vec{e_2}$) as shown in Figure 6.7. The dot product between the two vectors divided by the length of the base vector gives us the length of the projected vector:

$$\frac{\vec{e_1} \circ \vec{e_2}}{|\vec{e_2}|} = \frac{|\vec{e_1}| \cdot |\vec{e_2}| \cdot \cos(\gamma)}{|\vec{e_2}|} = |\vec{e_1}| \cdot \cos(\gamma)\,. \tag{6.45}$$

γ, denotes the angle between $\vec{e_1}$ and $\vec{e_2}$.

We then determine how often the projected vector fits into the base vector. The precise value is

$$\mu = \frac{|\vec{e_1}| \cdot \cos(\gamma)}{|\vec{e_2}|} = \frac{|\vec{e_1}| \cdot |\vec{e_2}| \cdot \cos(\gamma)}{|\vec{e_2}|^2} = \frac{\vec{e_1} \circ \vec{e_2}}{|\vec{e_1}|^2}. \tag{6.46}$$

However, we can only use integer factors for the subtraction. The resulting vector is otherwise no longer a lattice vector. Therefore, we round μ to the nearest integer

$$\mu' = \text{round}\left(\frac{\vec{e_1} \circ \vec{e_2}}{|\vec{e_1}|^2}\right). \tag{6.47}$$

The new, shorter, primitive lattice vector, $\vec{e_3}$, is then given by

$$\vec{e_3} = \vec{e_2} - \mu' \cdot \vec{e_1}. \tag{6.48}$$

We repeat this process for the shorter pair $(\vec{e_2},\ \vec{e_3})$ until the vector lengths stay constant. The pseudocode for this procedure is shown in algorithm 1. My implementation of this algorithm in Python 3 is in the appendix A.1.

Algorithm 1 Lagrange-Gauss algorithm to reduce primitive lattice vectors. Adapted from Steven Galbraith.[89]

Ensure: $|\vec{e_1}| < |\vec{e_2}|$

```
procedure ReduceBasis(e⃗1, e⃗2)
    A ← |e⃗1|^2
    μ ← (e⃗1 ∘ e⃗2)/A
    e⃗1 ← e⃗1 − Round(μ) · e⃗2                ▷ Shorten one basis vector
    B ← |e⃗2|^2
    while B < A do                          ▷ Check if the basis can be reduced
        Swap(e⃗1, e⃗2)
        A ← B
        μ ← (e⃗1 ∘ e⃗2)/A
        e⃗1 ← e⃗1 − Round(μ) · e⃗2            ▷ Shorten one basis vector
        B ← |e⃗2|^2
    end while
    return e⃗1, e⃗2                          ▷ Return the reduced primitive basis
end procedure
```

6.2.6 Bravais Type of Stretched Monolayer Samples

Real samples are never perfect. They always contain a certain amount of noise, distortions, and defects. We had to determine the Bravais type of a stretched monolayer from SEM images. The challenge was the slight disorder in the non-close packed lattices. Without the disorder, it would be possible to directly measure the primitive translation vectors and determine the Bravais type from the SEM images.

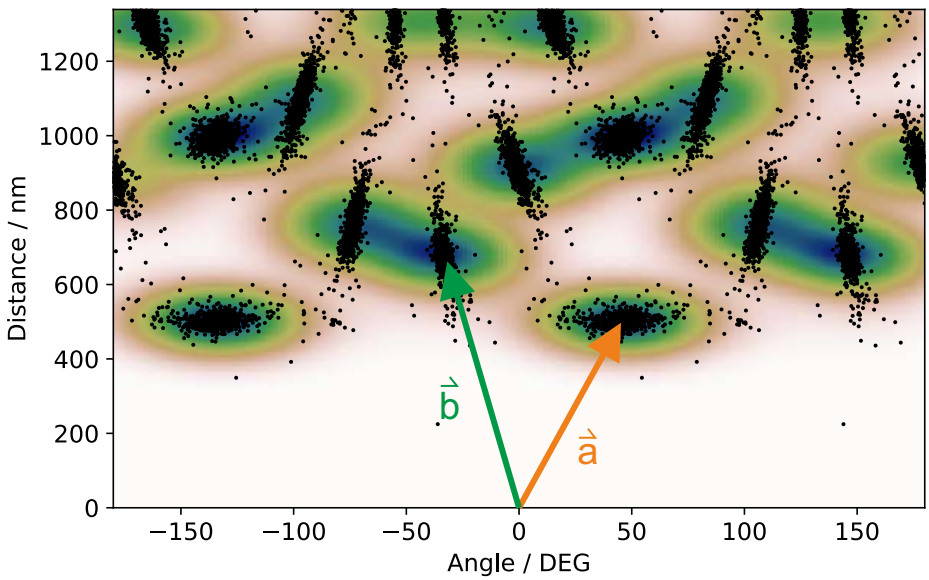


Fig. 6.8.: 2D histogram of the vectors between particle pairs from the SEM image in Figure 6.9. Each black dot represents an inter-particle vector. Clusters in the distribution are candidates for the primitive translation vectors. Reproduced from Hummel *et al.*[81] with permission from the American Chemical Society.

I extracted the average primitive translation vectors with the scatter plot shown in Figure 6.8. Each black dot represents an inter particle vector. The dots form clusters with finite size due to the slight disorder in the sample. The centers of these clusters are candidates for the average primitive translation vectors, $\vec{a}$ and $\vec{b}$. Only vectors that create a unit cell with minimum size are primitive translation vectors. The area of a unit cell, A, is

$$A = |\vec{a}| \cdot |\vec{b}| \cdot \sin\left(\sphericalangle(\vec{a}, \vec{b})\right). \tag{6.49}$$

Therefore, we have to choose the smallest vectors, defined by the cluster centers in Figure 6.8. We then compute the reduced primitive translations vectors with the Lagrange-Gauss algorithm from section 6.2.5. Finally, we determine the Bravais type from the angle between the translations vectors and their length ratio.

6.2.7 Extracting Particle Positions from Images

We primarily used scanning electron microscopy (SEM) to image colloidal monolayers. SEM is superior to optical microscopy due to its higher resolution. We needed the particle positions and their size distribution from the SEM images. The particle positions allow us to determine the Bravais type of the stretched lattice. We quantify the, usually low, particle polydispersity from their size distribution. To facilitate these goals, I programmed a graphical user interface around Matlab's circle detection function. We call it the *ParticleSizer*.

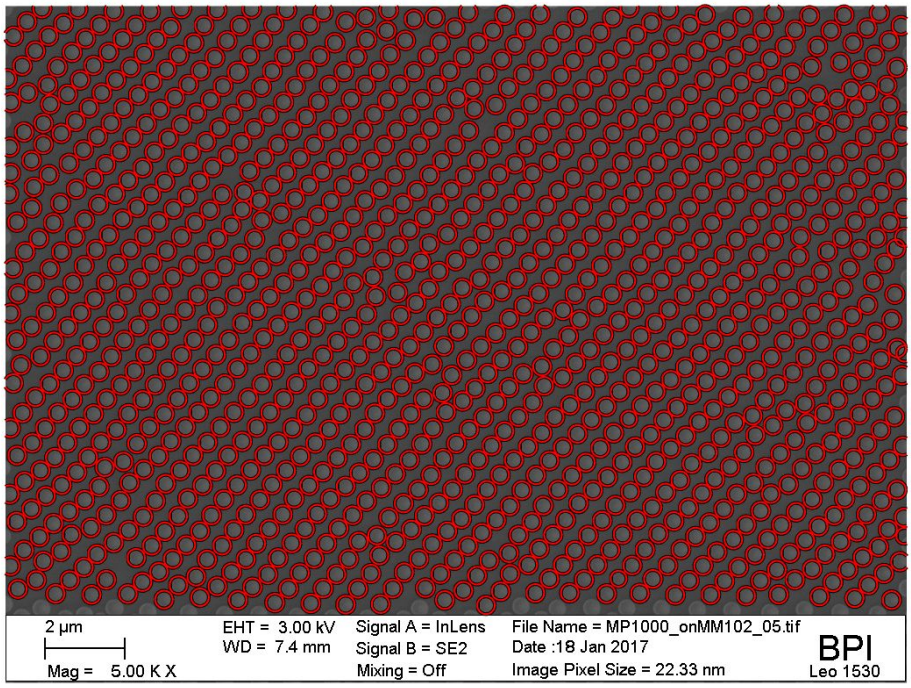


Fig. 6.9.: SEM image of a stretched monodomain. The red circles indicate the particles found by MATLAB's circle detection function after optimizing the search parameters manually. Reproduced from Hummel *et al.*[81] with permission from the American Chemical Society.

My software (see appendix A.4) detects all circles in an image as shown in Figure 6.9. The user optimizes different search parameters until most of the particles are correctly identified. Both particle size and position are recorded. The size distribution is automatically fitted with a log-normal distribution. The large number of particles in an image ensure statistically significant results with minimal effort. All data and figures are automatically saved.

6.2.8 Next Neighbor Analysis

Contact between particles determines the exchange of matter and energy between them. For instance, heat and electricity flow from particle to particle through the contact between them.[1] Another example is particle diffusion in foams with connected pores. The open pores can be interpreted as connected particles. The connection area and density strongly influence particle diffusion.[90]

The radial distribution function, $g(r)$, can show us local order around each particle. However, we needed to take an in depth look at the first coordination sphere around each particle. We needed to extract both the particle positions and the contact between the particles. Our samples contained small and large particles. This was an additional challenge that had to be addressed.

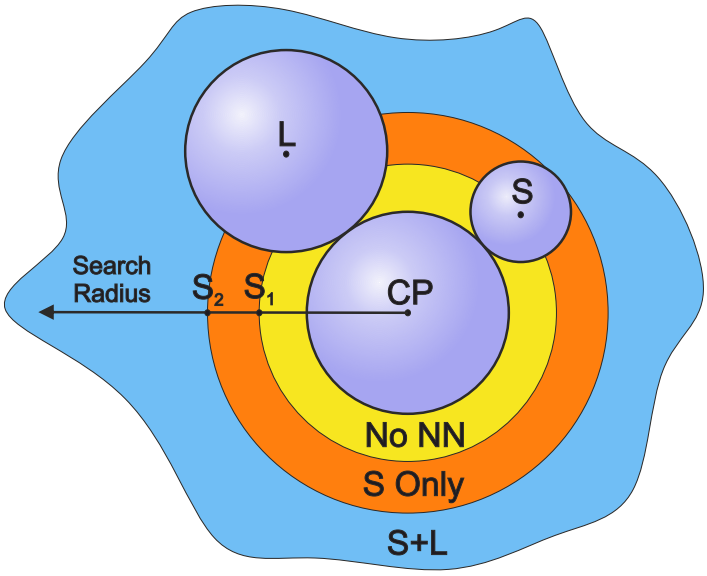


Fig. 6.10.: Different regions around a large central particle in a mixture of large and small particles. The innermost region (yellow) contains no other particles then the central particle (excluded volume). Only small particles are present in the intermediate, green, range. Both particle types are available as neighbors for radial distances larger than the large particle diameter. The amount, and type, of particles found as next neighbors strongly depends on the search radius (coordination sphere).

I developed custom software to analyze the particles in the first coordination sphere. We defined the first coordination sphere as a sphere with twice the radius of the large particles. This coordination sphere contains small particles that can be both contact and non-contact neighbors as shown in Figure 6.10. The advantage of my analysis was the distinction between contact and non-contact neighbors.

[1] We assume the space between particles is empty and we neglect radiative heat transport.

A random close jammed set of particles with two sizes contains two types of neighbors: touching neighbors and next neighbors (see Figure 6.10). Touching neighbors are in contact with the central particle. Next neighbors are particles inside the first coordination sphere around a central particle. A next neighbor is not necessarily touching the central particle. Two particles are in contact if the sum of their radii (r_i) is larger than their center to center distance:

$$d \leq (r_1 + r_2). \tag{6.50}$$

Based on this distinction, I computed the contact areas between the particles. The contact area, A, between a pair of particles is given, without proof, as

$$A = \pi \cdot \left[r_1^2 - \left(\frac{d^2 - r_2^2 + r_1^2}{2 \cdot d} \right)^2 \right]. \tag{6.51}$$

Furthermore, I computed the effective density of the particle assemblies. The true density has to take the particle overlap into account. Therefore, I had to subtract the intersection volume between each pair (i, j) of particles. This intersection volume, $V_{\text{intersect}}(i, j)$, is given by the sum of two spherical caps:

$$\begin{aligned} V_{\text{intersect}}(i,j) = \pi & \left[\frac{2r_i^3}{3} - r_i^2 \cdot (r_i - r_j) - \frac{(r_i - r_j)^3}{3} \right] + \\ \pi & \left[\frac{2r_j^3}{3} - r_j^2 \cdot (r_j - r_i) - \frac{(r_j - r_i)^3}{3} \right]. \end{aligned} \tag{6.52}$$

The effective density, ρ_{eff}, is then given by

$$\rho_{\text{eff}} = \rho_{\text{particle}} \cdot \frac{\sum_{k=1}^{N} \frac{4}{3}\pi r_k^3 - \sum_{i,j;i\neq j}^{N} V_{\text{intersect}}(i,j)}{V_{\text{assembly}}}, \tag{6.53}$$

where ρ_{particle} is the density of the particles.

6.3 Computing Custom Colloidal Glasses

We simulated heat transport in colloidal assemblies to unveil microscopic effects that influence heat flux. We discovered percolation thresholds in binary colloidal glasses (see chapter 12) and a reduction of the effective thermal conductivity through increased thermal pathways (see chapter 13). These simulations are an indispensable tool for our research.

We used COMSOL Multiphysics,[91] a finite element software package for our simulations. These simulations require a detailed 3D model of our colloidal assemblies. Therefore, we computed the particle coordinates for a random close jammed colloidal glass with molecular dynamics.[92] The challenge was to decorate the coordinates randomly with different particles. These particles could vary in size and can be either filled or hollow spheres. Overlap between different materials had to be removed. Finally, we needed to import the structure into COMSOL.

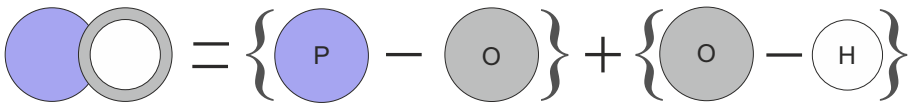


Fig. 6.11.: Graphical equation for the creation of complex 3D particle assemblies. The blue circle represents a solid polymer particle. The hollow sphere is created by subtracting a hole (white) from a solid sphere (gray).

I developed Python scripts (see section A.2) that generate random mixtures of solid and hollow particles. We can adjust both the mixing ratio and the particle type. The result is an AutoCAD script[93] with three separate sets of *solid* spheres:

- one for polymer colloids (P),
- one for the outer shell of hollow silica particles (O),
- and one for the holes in the hollow silica particles (H).

These sets were then added and subtracted according to the scheme shown in Figure 6.11. The resulting structure, in the AutoCAD file can then be imported into COMSOL.

The hollow particles in the actual samples consist of relatively hard silica. In contrast, the solid particles are softer polymer colloids. We therefore assumed that the hard hollow spheres push into the soft solid spheres. We modeled this behavior by assigning the overlap volume between two particles fully to the silica particle.

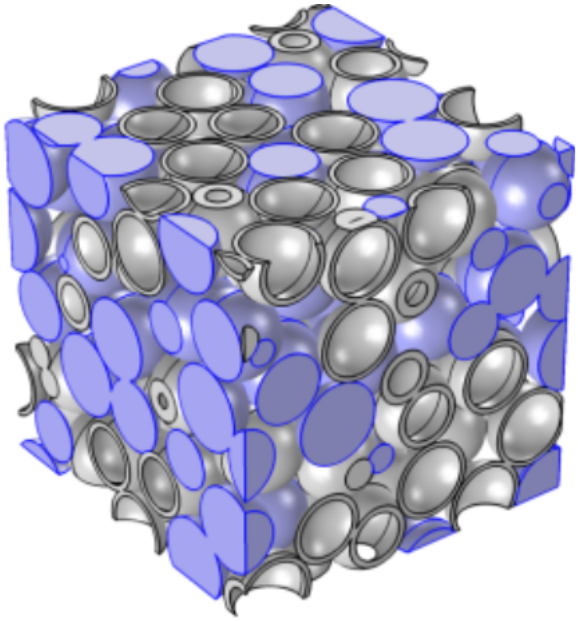


Fig. 6.12.: 3D random assembly of solid and hollow sphere particles. The hard hollow spheres penetrate into the softer solid spheres, similar to the real sample. Reproduced from Ruckdeschel *et al.*[49] with permission from the American Physical Society.

A rendering of the resulting 3D particle assembly is shown in Figure 6.12. We cut a cubic subset from the particle assembly to simulate heat transport in all three directions. We can then either average the results for isotropic samples or compare heat flux in different direction. This was especially useful during our discovery of percolation thresholds. Percolation paths, in these small samples, are first formed in one direction. A significant difference in the direction dependent thermal conductivity hinted the presence of a percolation path.

The particle overlap was a critical factor. Heat flux in colloidal glasses is strongly influenced by the size of the connection areas between particles.[45] My script allows us to adjust the relative particles sizes to ensure a consistent overlap. In addition, I computed the contact area distribution for further analysis.

6.4 Xenon Flash Analysis

Xenon flash analysis is a commercially available method to determine the thermal diffusivity of bulk materials.[94] We used a XFA500 from Linseis with a cooled InSb infrared detector. This instrument can change both temperature, gas pressure and composition during a measurement. The samples have to be around 5 mm in diameter with a *homogeneous* thickness of at least 1 mm.

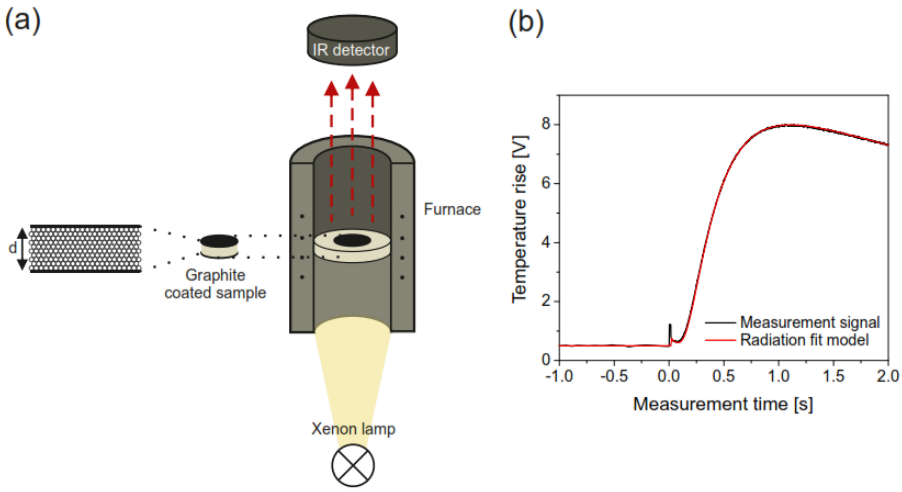


Fig. 6.13.: a) Components of a Xenon flash analysis instrument. The sample with homogeneous thickness is coated with graphite on both sides. This coating ensures good absorption of the light flash on the bottom and strong IR emission on the top. b) Raw data from a Xenon flash analysis measurement and the corresponding fit. Reproduced from Ruckdeschel *et al.*[49] with permission from the American Physical Society.

A single measurement proceeds as follows: A short light flash is absorbed on the bottom of the sample. The absorbed energy increases the surface temperature by a few Kelvin. Heat starts to diffuse through the sample and heats up the top surface. An infrared detector then measures the temperature rise on the top as a function of time (see Figure 6.13b).

Different methods have been developed to analyze the shape of the surface temperature evolution. Parker *et al.*[94] used the half rise time, $t_{1/2}$ to compute the thermal diffusivity, α, from a Xenon flash measurement

$$\alpha = 1.38 \cdot \frac{d^2}{\pi^2 \cdot t_{1/2}}. \tag{6.54}$$

Note that the sample thickness, d, contributes quadratically to the end result. A homogeneous, and well characterized, thickness is important for accuracy.

6.5 The Photoacoustic Technique

The conversion of intensity modulated light into pressure waves (sound) in matter is called the photoacoustic effect. A prominent use is photoacoustic tomography,[95] a 3D medical imaging method based on ultrasound.

We use photoacoustic to characterize through plane heat flux in thin films.[96] Our setup is optimized for thin films between 500 nm and 1 µm. Films outside of this range create an unacceptably large error. The films are on top of a well characterized reference substrate, like quartz. A thin ($\approx$ 100 nm) gold layer acts as a heat transducer during the measurement. Both the precise sample and gold thickness are required for the data analysis.

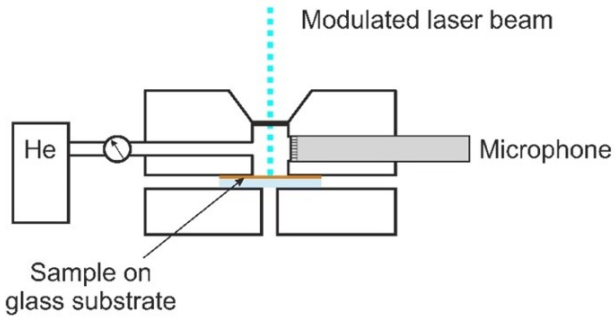


Fig. 6.14.: Crosssection through a photoacoustic measurement cell. The sample forms the bottom of a gas tight cell, filled with helium. A modulated laser beam is coupled into the cell through a sapphire window and periodically heats the sample surface. This creates pressure waves in the helium. Helium creates strong pressure waves due to its large thermal conductivity. The microphone on the right converts the pressure waves into an electric signal. Reproduced from Wang *et al.*[97]

A crosssection through a photoacoustic cell is shown in Figure 6.14. The photoacoustic signal is generated as follows: An intensity modulated laser beam periodically heats the sample surface. The heat diffuses both into the sample and into the helium, depending on the samples thermal properties. This increases the temperature of the helium. As a result, the helium periodically expands and creates pressure waves. A sensitive microphone converts the pressure waves into an electric signal. We use a lock-in detector to measure the phase, ϕ, of the pressure waves. Sweeping the excitation frequency, ω, gives us $\phi(\omega)$. A second measurement of a thermally thick substrate yields a reference phase, $\phi_0(\omega)$. We then fit a model to the phase difference,

$$\Delta\phi(\omega) = \phi(\omega) - \phi_0(\omega), \tag{6.55}$$

to obtain the thermal diffusivity and conductivity of the thin film. The general fit equation for a multilayered structure was developed by Hu *et al.*[98]

6.6 Lock-in Thermography

6.6.1 Introduction

Lock-in thermography is the combination of thermography with lock-in detection[99] to measure heat transfer in solid materials. Thermography measures the surface temperature of a sample with high temporal (milliseconds) and spatial (micrometer) resolution. Lock-in detection is superior to other detection methods due to its good noise rejection.

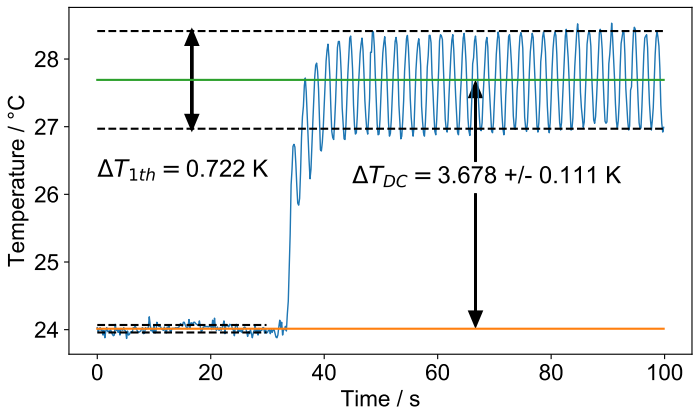


Fig. 6.15.: Surface temperature vs. time recorded with our IR camera near the heating center during a lock-in thermography measurement. I included the baseline (orange) to compute the true DC temperature offset. The line laser starts to periodically heat the sample at t = 34.8 s. Thermal equilibrium is then reached after less than 30 s.

The basic idea behind lock-in thermography is to periodically heat the sample and analyze its temperature response in the frequency domain. Periodic excitations have a long history of use for heat transfer measurements. They were first used by Ångström in 1863.[100] Measurements with periodic perturbations are advantageous over constant or impulse perturbations. The amplitude and phase of periodic signals can be measured with high precision by lock-in detection.

A focused line laser periodically heats the sample during the measurement. The temperature oscillations approach equilibrium within a few periods as shown in Figure 6.15. There are two important quantities: the DC temperature offset, ΔT_{DC} and the amplitude of the periodic signal ΔT_{1st}. We employ lock-in detection to extract ΔT_{1st} once equilibrium has been reached. We then compute the thermal diffusivity from the change of ΔT_{1st} with distance from the excitation source.

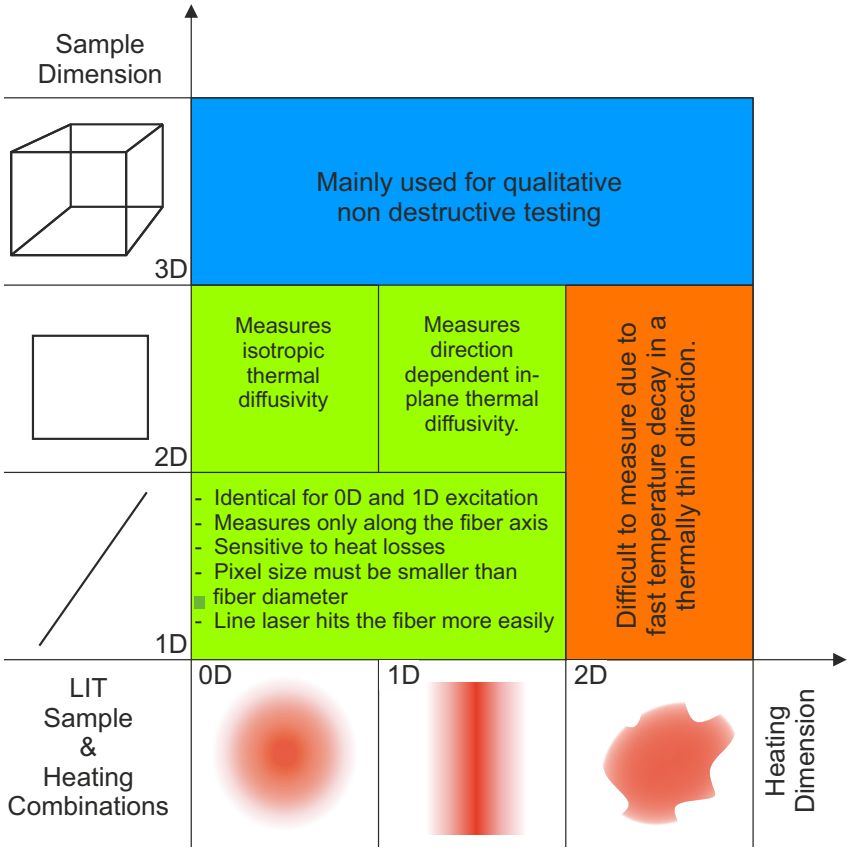


Fig. 6.16.: Possible combinations of sample and heating dimensionality. Useful heating geometries range from point (0D) to line (1D) and area (2D). A sample dimension is considered negligible thin when its thickness is small relative to the thermal decay length μ of the excitation. This results in three distinct sample geometries: fiber (1D), thin film (2D), and bulk (3D).

Bulk samples (see Figure 6.16) are mostly investigated for non destructive testing.[101] Cracks and air bubbles impede heat flow. Lock-in thermography enables us to detect cracks and bubbles in construction materials like concrete[102] or fiber composites.[103] Another application of non destructive testing are electronic components. A good overview on fault detection in electronics is the book by Breitenstein.[104]

Samples for thermal diffusivity measurements with lock-in thermography fall into two categories: thin films (2D) and fibers (1D). We can excite such samples with either a point or a line source (see Figure 6.16). A 2D excitation is not useful because the thermal oscillations would decay along a thermally thin dimension. A dimension is thermally thin when it is much smaller than the thermal decay length:

$$\mu = \sqrt{\frac{\alpha}{\pi f}}, \tag{6.56}$$

at frequency, f, and thermal diffusivity, α.

Line sources are generally superior to point sources. A line source can fully replace a point source. In addition, a line source increases the signal-to-noise ratio by exciting a larger sample area. A line source creates a 1D heat flow profile, even in anisotropic samples. This simplifies the data analysis for anisotropic samples. Rotating the line source gives us access to the full 2D thermal diffusivity tensor. Thin fibers are easily excited by a line source whereas a points source requires careful alignment.

Lock-in thermography requires samples that are:

- Sufficiently thin, $d \ll 2 \cdot \sqrt{\frac{\alpha}{\pi f}}$, in at least one dimension
- Sufficiently strong emissivity (can be improved by evaporating 25 nm carbon)
- Homogeneous and sufficiently high absorption at the laser wavelength
- At least 10×10 mm in size
- Mechanically stable (freestanding films or fibers holding their own weight)
- Stable in vacuum (to reduce thermal losses)

6.6.2 Excitation Signals in Lock-in Thermography

Different excitation signals are used for lock-in thermography (see Fig. 6.17). The standard signal is a sine wave. Sine waves contain only one spectral component. However, they are fairly difficult to produce. Square waves are often used instead due to their simple on/off modulation. Square waves contain higher harmonics that can be measured and analyzed. I developed a new, multiplexed, signal (see chapter 17). The idea is to combine sine waves with the desired frequencies and amplitudes into one signal. This creates a smaller DC offset and higher signal yield. Short pulses are mostly used for non-destructive testing. They are commonly produced by flash lamps.

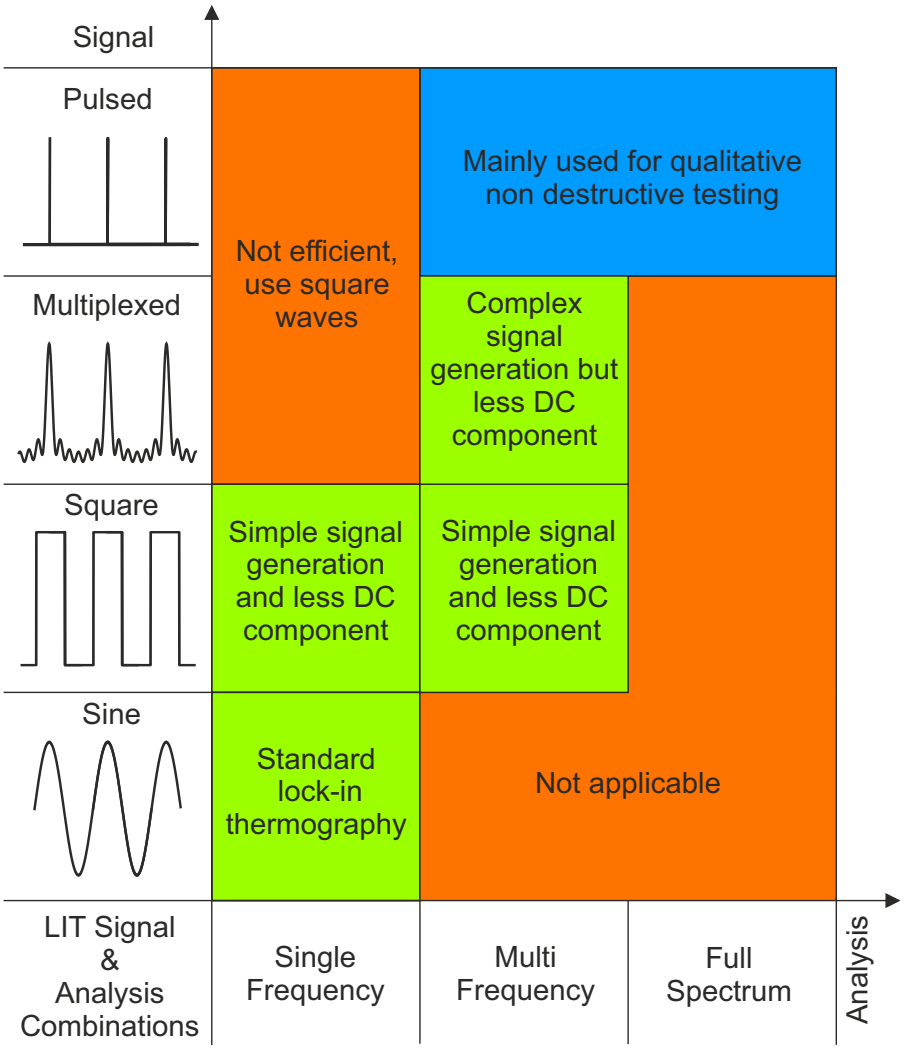


Fig. 6.17.: Different excitation signals and data analysis used for lock-in thermography. Standard lock-in thermoraphy uses a sinusoidal excitation and analyzes the resulting amplitude and phase. The other signals contain higher harmonics that can be measured and analyzed for different purposes.

6.6.3 Infrared Thermography

Every object emits thermal radiation.[105] The amount of thermal radiation depends on the temperature. Thermography uses the thermal radiation to image an object.[106] However, IR cameras do not measure temperature. Instead they measure the radiated power emitted *and* reflected by the object within a certain wavelength range.[107],[108] A calibration to the objects emissivity, ϵ, allows us to infer the surface temperature. The Stefan-Boltzmann law[109] describes the radiated power in all directions, P, per surface area, A, as a function of absolute temperature, T

$$P = \epsilon \cdot \sigma \cdot A \cdot T^4. \tag{6.57}$$

σ is the Stefan-Boltzmann constant and has a value of $5.67 \cdot 10^{-8}$ $\mathrm{W/m^2K^4}$.[110] Note that the radiated power is proportional to T^4. The emissivity, $\epsilon(\lambda)$, measures how good a surface emits radiation with a certain wavelength. Black bodies have a constant emissivity of 1. Their thermal radiation spectrum[111] follows Planck's law:

$$P(\lambda) = \frac{2 \cdot \pi \cdot h \cdot c^2}{\lambda^5} \cdot \frac{1}{\exp\left(\frac{h \cdot c}{\lambda \cdot k_\mathrm{b} \cdot T}\right) - 1} \tag{6.58}$$

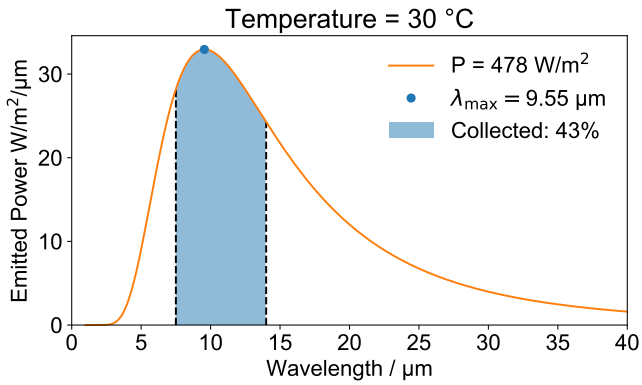


Fig. 6.18.: Thermal radiation power per area and wavelength emitted by a surface with emissivity 1 (black body) at 30 °C. The maximum power is emitted around 9.55 µm. The total power per square meter is 478 $\mathrm{W/m^2}$. Our IR camera collects and detects light in the wavelength range from 7.5 to 14 µm (blue region). This amounts to around 43 % of the received power from the surface.

Our IR camera uses a microbolometer array to detect infrared radiation. Figure 6.18 shows the spectrum of a black body at 30 °C. Only radiation between 7.5 to 14 µm is captured by the microbolometer. The radiation is absorbed by the surface of a bolometer pixel. This changes its temperature. The temperature change is detected through a change in electric resistance. The main advantage of a microbolometer array is the operation at room temperature. No cooling is required.

6.6.4 Our Lock-In Thermography Setup

Lock-in thermography requires a fairly simple, robust, and compact setup relative to techniques like frequency- or time domain thermo reflectance.[112],[113]

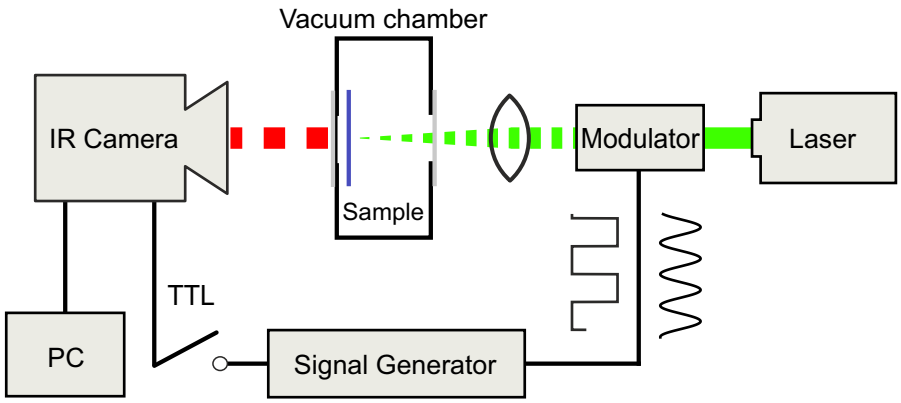


Fig. 6.19.: Components of our lock-in thermography setup. An intensity modulated laser beam is focused on the thin film inside the vacuum chamber. The laser periodically heats the sample. The intensity of our diode laser is modulated with an analog signal (e.g. sine or square) from the signal generator. Our IR camera can be phase locked to the signal generator via a TTL signal. Lock-in detection is handled by the software.

The components of our setup are shown in Figure 6.19. We use a VarioCAM HD 875 from Infratec for thermal imaging. The pixel size with the close-up lens is 29 µm at a working distance of 33 mm. We placed the IR camera onto a micrometer controlled translation stage aligned with the laser axis. This allows us to manually focus the camera by adjusting the camera-sample distance instead of changing the focus. We designed a custom vacuum chamber and sample holder. The vacuum chamber is mounted on a x-y-z micrometer translation stage for alignment. A rotary vane pump from Vacuubrand, type RZ 9, ensures a gas pressure of less than 10^{-3} hPa. The low pressure significantly reduces the heat losses from the sample.

The sample is periodically heated by the laser from the back. We use a diode laser type 13LR12-m125+55CM-520-56-O08-T15-PS-7 from Schäfter+Kirchhoff. This laser has a wavelength of 520 nm and creates a 40 µm ($1/e^2$) laser line at a working distance of 122 mm. Both the beam shaping optics and the modulation capability are encapsulated within the small laser module. We create the analog modulation signal with a RIGOL DG1000 signal generator. Arbitrary signals can be defined with .raf files. I generate the .raf files with the Python script shown in appendix A.3.12. In addition, we can generate a square wave TTL signal with the IR camera. This allows us to phase lock the laser modulation to the IR camera.

6.6.5 Thermogram Pixel Size Calibration

Measuring thermal diffusivity, with dimension ($^{\text{length}^2}/_{\text{time}}$), requires information about temperature, distance, and time. Therefore, we need to know the precise length scale in our thermograms. We decided to calibrate the pixel size in our lock-in thermography setup with a length standard.

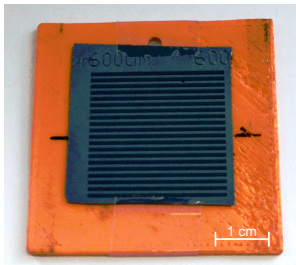


Fig. 6.20.: Length standard with 600 µm wide lines made from SU-8 photoresist on a silicon waver, mounted on a 3D printed frame. Silicon and SU-8 transmit and emit IR radiation differently which creates contrast.

I prepared the length standard with photolithography. Figure 6.20 shows a photograph of our length standard. The parallel lines are spaced 600 µm apart. SU-8 photoresist and the silicon waver substrate have different optical properties in the IR. This creates contrast in the resulting thermograms (see Fig. 6.21a).

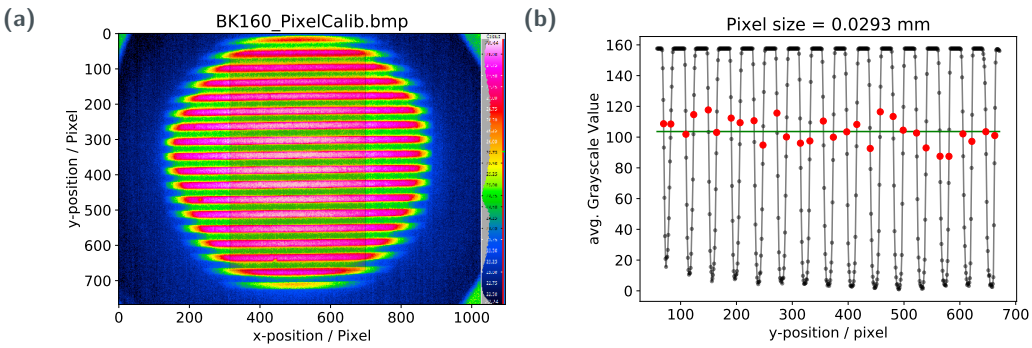


Fig. 6.21.: a) Thermogram of the calibration sample inside the vacuum chamber. The round line boundary is the edge of the IR window. The user selects a rectangular region, containing only lines, for averaging. b) Horizontally averaged intensity profile as a function of vertical position. My software automatically computes the pixel size from the averaged profile.

We insert the reference at the sample position and take an IR image (Fig 6.21a) after manually focusing. The user selects a rectangular area containing only lines. My software then computes the pixel size from the averaged horizontal cross section (Fig. 6.21b). Our IR camera has a pixel size of 29 µm at closest focus. My source code for the pixel calibration procedure is in appendix A.3.5.

6.6.6 Analysis for Isotropic Films Heated by a Point Laser

The data analysis for isotropic thin films, excited by a point source, was developed by Fabbri *et al.*[114] and Zhang *et al.*[115] I modified their analysis such that all image pixels above the noise background are used. My software allows rapid data analysis (< 1 min per measurement) with minimal user input. This increases productivity and ensures repeatability. My modified version follows these steps:

1. Detect the center pixel by searching for the highest amplitude
2. Convert pixel positions to radial distance from the center
3. Linearize the amplitude as a function of radial distance
4. Extract the slopes of amplitude and phase vs. radial distance
5. Compute the thermal diffusivity from the slope product with Eq. (6.63)

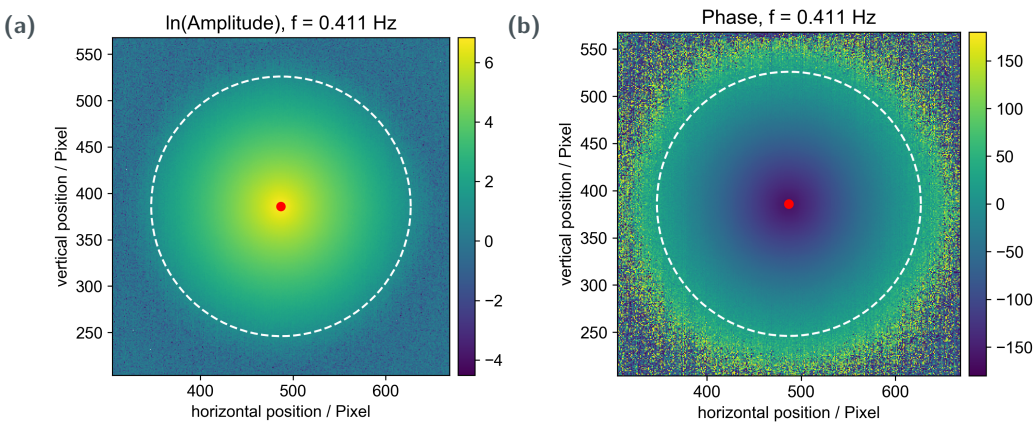


Fig. 6.22.: Position dependent amplitude (a) and phase (b) of the temperature oscillations on the sample surface. The sample is heated by a focused laser beam. The red dot is the center pixel with the highest amplitude. A user defined circle (white dashed line) shows the region used for the radial conversion.

Step 1: I detect the center, where the focused laser hits the sample, from the amplitude image (Fig. 6.22a). The center pixel, (x_c, y_c), is the pixel with the highest amplitude. A linear search through the amplitude image yields the center pixel. I assume, that the amplitude and phase image (Fig. 6.22b) have the same center.

Step 2: We then convert the 2D amplitude and phase images into radial data. We use the Pythagorean theorem

$$(x - x_c)^2 + (y - y_c)^2 = r^2, \text{ and solve for } \quad r = \sqrt{(x - x_c)^2 + (y - y_c)^2} \tag{6.59}$$

to convert the pixel coordinates (x, y) into radial distance, r, from the center (x_c, y_c). Each pixel now has a radial distance assigned to it.

Step 3: We linearize the amplitude by computing

$$A_{\text{linearized}} = \ln\left(A_{\text{radial}} \cdot \sqrt{r}\right). \tag{6.60}$$

The phase already follows a linear trend when plotted against the radial distance.

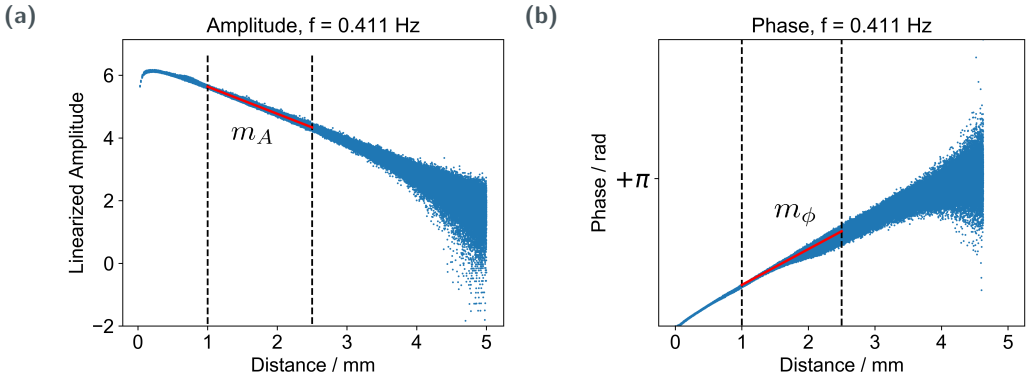


Fig. 6.23.: Linearized amplitude (a) and phase (b) of the temperature oscillations as a function of radial distance. The black dashed lines indicate the fit boundaries. The slopes of amplitude (m_A) and phase (m_ϕ) are obtained by a linear fit.

Step 4: Next, we perform a linear fit of the linearized data. The fit boundaries ($r_{\min}, r_{\max}$) are limited by the thermal decay length

$$\mu = \sqrt{\frac{\alpha}{\pi f}}. \tag{6.61}$$

The minimal fit distance should be: $r_{\min} > 2 \cdot \mu$.[116] The maximal distance is usually limited by the noise floor. Note, that the thermal decay length must be smaller than the sample dimension, L. Reflections from the sample boundary would otherwise disturb the measurement. The sample boundary can be neglected when[76]

$$L/\mu < 5. \tag{6.62}$$

In this case, the thermal oscillations decayed before they reached the sample boundary. In practice, the user selects the linear range by hand, having in mind the minimal distance, $r_{\min}$. I use all selected pixels for the linear fit.

Step 5: Finally, we compute the thermal diffusivity, α, from the amplitude and phase slopes (m_A and m_ϕ) according to

$$\boxed{\alpha = \frac{\pi \cdot f}{|m_\mathrm{A} \cdot m_\phi|}}. \tag{6.63}$$

Measuring at different excitation frequencies, f, can reduce the error of the thermal diffusivity, α.

6.6.7 Analysis for Anisotropic Films Heated by a Line Laser

The data analysis for isotropic thin films, excited by a line source, was developed by Wolf *et al.*[73] I extended their analysis to anisotropic samples. I use a line laser as the line source. The laser line can point in an arbitrary orientation perpendicular to the sample. My software automatically detects the laser line position and orientation. My modified version follows these steps:

1. Detect the laser line position and orientation
2. Convert the pixel positions to linear distance from the laser line
3. Linearize the amplitude data
4. Extract the slope from a linear fit of amplitude and phase vs. linear distance
5. Compute the thermal diffusivity from the slope product with Eq. (6.72)

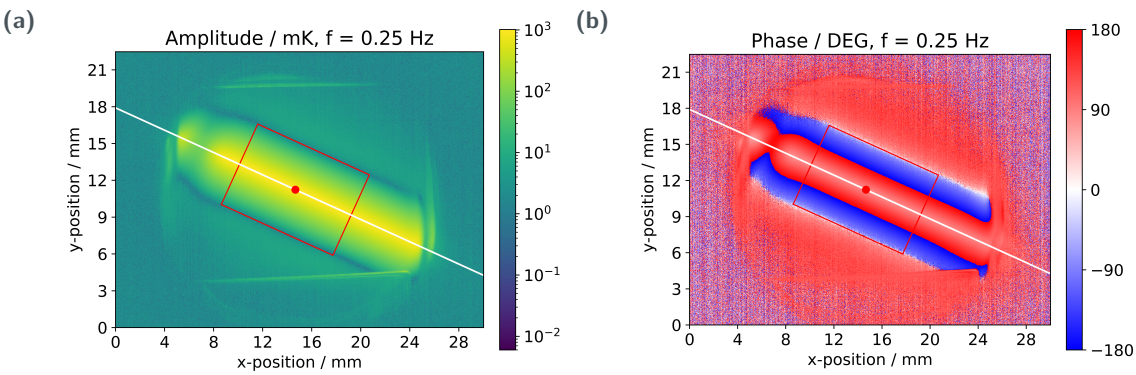


Fig. 6.24.: Position dependent amplitude (a) and phase (b) of the temperature oscillations on the sample surface. The automatically detected laser lines are indicated by the white line and red dot. Only data within the red rectangle is used for the linear conversion.

Step 1: I detect the line center (x_c, y_c) by averaging the $n \approx 5000$ pixels with the highest amplitude:

$$x_c = \frac{1}{n}\sum_{i=0}^{n} x_i, \quad y_c = \frac{1}{n}\sum_{i=0}^{n} y_i. \tag{6.64}$$

Next, I fit a line through the n pixel positions. I assume, that the line passes through the center. This reduces the fit parameters to the direction of the line. There are two challenges during the line fit:

1. vertical lines have an infinitely large slope, and
2. linear regression only minimizes the offset in the y-direction.

I avoid large slopes by representing the line by its angle, β, with the x-axis:

$$y = \tan(\beta) \cdot (x - x_c) + y_c. \tag{6.65}$$

If the fit produces a sometimes imperfect slope outside of [-1, 1], I switch the x and y coordinates, redo the fit, and recompute the direction β. This procedure ensures that the *fitted* slope is always between [-1, 1].

Linear regression only minimizes the y-difference between the fit line and the pixels. This creates a bias toward differences in the y-direction. I solved this issue with orthogonal distance regression.[117] This ensures that the orthogonal distance from each pixel to the laser line is minimized.

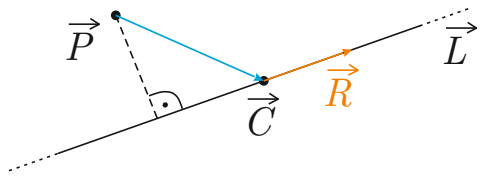


Fig. 6.25.: Vector representation of the line laser, $\vec{L}$, with the center, $\vec{C}$, as the reference and $\vec{R}$ as the direction vector. The black dashed line indicates the shortest distance from a pixel, $\vec{P}$, to the laser line.

Overall, I use a vector representation (see Fig. 6.25) of the laser line

$$\vec{L}(\lambda) = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + \lambda \cdot \begin{pmatrix} \sin(\beta) \\ \cos(\beta) \end{pmatrix} = \vec{C} + \lambda \cdot \vec{R}(\beta), \ \lambda \text{ in } \mathbb{R}, \tag{6.66}$$

for further computations. The resulting laser line and center are shown in Figure 6.24. I assume that the amplitude and phase images have the same laser line.

Step 2: Convert the pixel positions to linear distance from the laser line. We need both the shortest distance to the line, d_{Shortest}, and the distance from the closest point on the line to the center, d_{Center}. The center distance, on the line, is the projected length of the connection between the pixel position and the center:

$$d_{\text{Center}} = \left| \left(\vec{C} - \vec{P} \right) \circ \vec{R}(\beta) \right| \tag{6.67}$$

We then use the Pythagorean theorem to compute the shortest distance to the line:

$$d_{\text{Shortest}} = \sqrt{\left| \left(\vec{C} - \vec{P} \right) \right|^2 - d_{\text{Center}}^2} \tag{6.68}$$

The two sides of the laser line are mirror images. Therefore, we can use pixels from both sides of the line for analysis to reduce the noise.

Step 3: We linearize the amplitude data by computing

$$A_{\text{linearized}} = \ln\left(A_{\text{line}}\right). \tag{6.69}$$

The phase is already linear as a function of distance from the laser line.

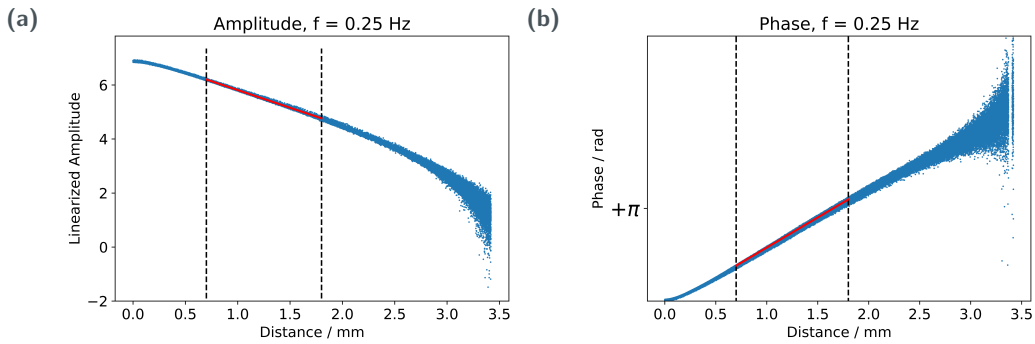


Fig. 6.26.: Position dependent amplitude (a) and phase (b) of the temperature oscillations on the sample surface. The black dashed lines indicate the fit boundaries. The slopes of amplitude (m_{A}) and phase (m_ϕ) are obtained by a linear fit.

Step 4: Next, we perform a linear fit of the linearized data. The fit boundaries ($d_{\text{min}}, d_{\text{max}}$) are limited by the laser line width, d_{laser}, the thermal decay length

$$\mu = \sqrt{\frac{\alpha}{\pi f}}, \tag{6.70}$$

and the sample dimension perpendicular to the laser line, L. The sample boundary can be neglected when[76]

$$L/\mu < 5. \tag{6.71}$$

In this case, the thermal oscillations have decayed long before they reached the sample boundary. However, the thermal diffusivity, α, is not know before the measurement. In practice, the user selects the linear range by hand, having in mind the minimal distance, d_{min}. I use all individual pixels within the linear region for the linear fit.

Step 5: We then compute the thermal diffusivity, α, from the amplitude and phase slopes (m_{A} and m_ϕ) according to

$$\boxed{\alpha = \frac{\pi \cdot f}{|m_A \cdot m_\phi|}}. \tag{6.72}$$

We can repeat this process for different orientation of the line laser to obtain the direction dependent thermal diffusivity.

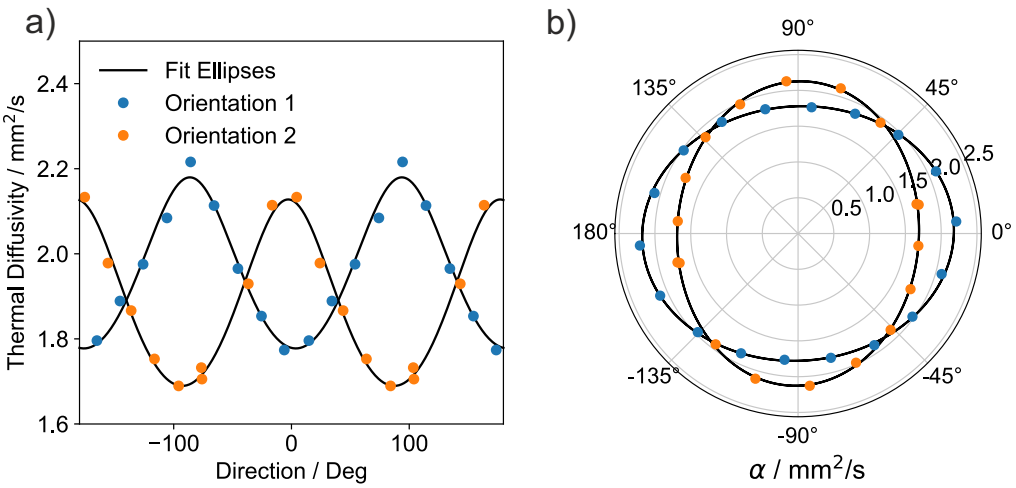


Fig. 6.27.: Direction dependent thermal diffusivity of MT200+ Kapton foil in a cartesian coordinates system (a) and in polar coordinates (b). The black lines are fitted ellipses to the data. I measured the sample in two orientations that are approximately 90° rotated. The results are very similar in both orientation. This test ensures that the measured anisotropy is intrinsic to the Kapton foil and not a measurement artifact.

Figure 6.27 shows the direction dependent thermal diffusivity of Kapton. I fit an ellipse to the data to extract the thermal diffusivity along the major and minor axis together with the direction of the major axis (see Table 6.1). The thermal diffusivity in the major axis direction is approximately 23 % larger than along the minor axis. Rotating the sample by 90° rotates the measured anisotropy also by ≈ 90°.

Tab. 6.1.: Anisotropic thermal diffusivity of Kapton (MT200+) measured with line laser lock-in thermography. The sample was rotated by ≈ 90° between orientation 1 and 2. The measured anisotropy rotates accordingly.

		Thermal Diffusivity [mm²/s]		
Orientation	Direction	Major Axis	Minor Axis	Ratio
1	87°	2.128	1.690	1.259
2	3°	2.180	1.778	1.226

6.6.8 Analysis Software Architecture

During my PhD, I used programming to build and automate scientific tools. The analysis software library is an essential part or our lock-in thermography setup. Developing our own software allows us to tailor its functionality to our specific needs.

My software reduces the time for data analysis from hours to minutes. Only minimal user input is required. The different steps and results are automatically saved and documented in a report. Plots of important quantities are automatically generated. The plots and reports can be directly used in lab notes by the user.

Fig. 6.28.: Class hierarchy of the lock-in thermography analysis software. The object oriented design facilitates maintenance, extension, and reuse of the code. Red arrows indicate inheritance between a parent class and the derived child classes. Black arrows indicate composition of classes.

I used object oriented programming for the software layout. This is the current industry standard for software development. A good overview on object-oriented design is the book by Gammaet al.[118] The main idea behind object-oriented programming is encapsulation and reuse. Both data and methods are grouped by task and encapsulated into different classes. Composition (the use of one class inside another) and inheritance (the transfer of data and functions from a parent to a child class) make it easy to reuse code for different tasks. A clear hierarchy (see Fig. 6.28) between the classes makes the code easier to understand, extend, reuse, and maintain. The entire source code for the lock-in thermography analysis is in appendix A.3.

7.2 Synopsis

During my PhD, I utilized programming and mathematics to advance measurement and analysis techniques. I applied computational tools to a variety of research projects. They range from particle self-assembly to create nanostructured materials and consecutive thermal transport characterization to the development of new thermal transport metrology. The following paragraphs contain a summary of the diverse research challenges I faced and how I approached them during my PhD.

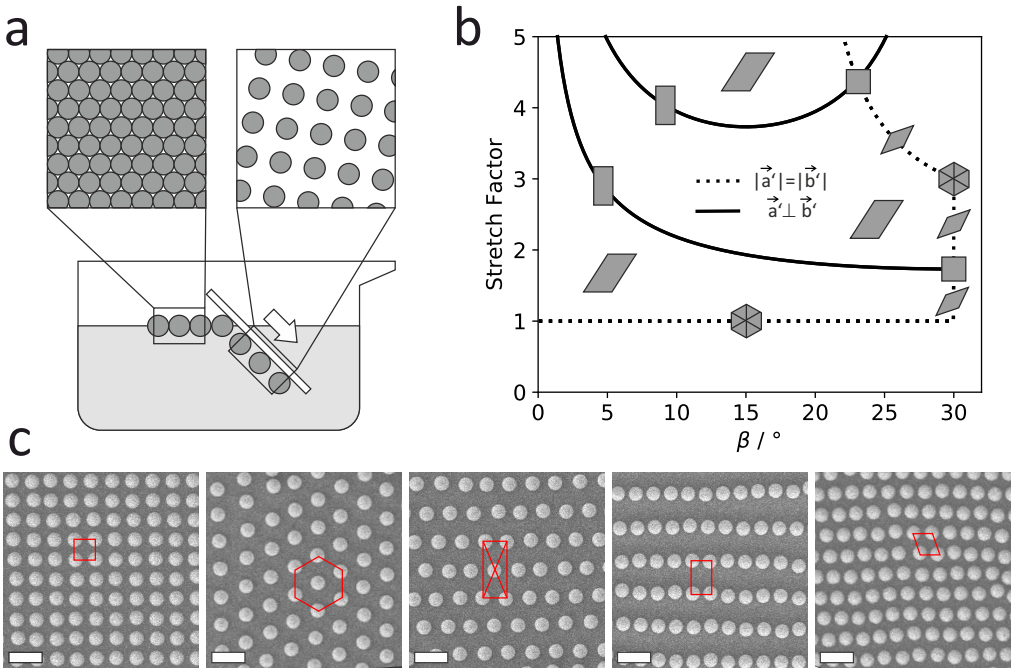


Fig. 7.2.: a) Z-type transfer of a hexagonal monolayer onto a hydrophobic substrate. The original close packed hexagonal monolayer (left) on the water is transformed into a non-close packed monolayer (right) on the substrate. b) Stretch factor - direction phase diagram for initially hexagonal close packed lattices. Square and hexagonal lattices occur at specific points, rectangular lattices appear on the solid lines. The remaining space is filled with oblique lattices. c) SEM images non-close packed particle monolayers after the transfer to a hydrophobic substrate. All five Bravais types are accessible. Scale bars are 1 µm. Reproduced and adapted from Hummel *et al.*[81] with permission from the American Chemical Society.

My colleagues and I discovered a new technique to produce non-close packed colloidal monolayers (see chapter 10). Our technique starts from close packed, hexagonal, monolayers of colloidal particles floating on the surface of water. These monolayers are usually transferred to a substrate for further processing as shown in Figure 7.2a. We found that the monolayer experiences an uniaxial stretch when transfered to a *hydrophobic* substrate. The stretch factor is only determined by the contact angle between the substrate and water.

All five Bravais lattices are accessible with our technique (see Fig. 7.2b, c). This opens up new possibilities to tune the optical band structure of colloidal photonic crystals. Another application would be anti-reflective coatings similar to the concept from Stelling *et al.*[119]

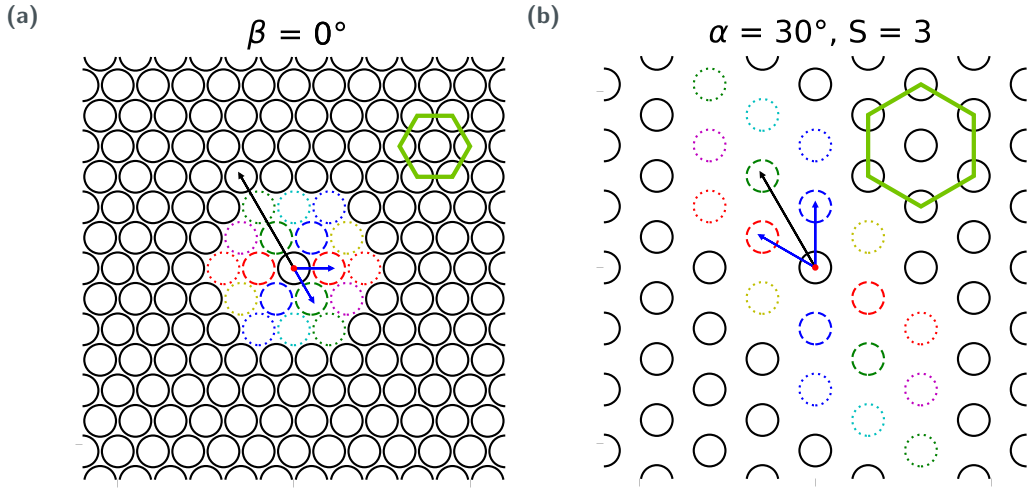


Fig. 7.3.: 2D lattices of the original, densely packed hexagonal lattice (a) and the resulting, stretched lattice (b). The unit cell vectors are shown in blue. The stretch direction and amount is indicated by the black arrow. Particles in the first two coordination spheres are color coded. Note, that a simple, uniaxial stretch transforms a closed packed hexagonal lattice into a non-close packed hexagonal lattice.

In this work, I developed different tools to analyze the lattice transformation. Our hypothesis was, that the lattice transformation was a simple uniaxial stretch. We developed a theoretical description of uniaxial stretching with linear algebra. I later simplified our approach by using only vectors to describe stretching (see section 6.2). There are three important parameters that define an uniaxial stretch:

α: the direction of the uniaxial stretch, relative to the transfer direction,

β: the original orientation of the lattice, relative to the transfer direction,

S: the stretch factor.

I developed a Python script (see appendix A.1.2) to visualize the effect of an uniaxial stretch on a close packed hexagonal lattice. Figure 7.3 shows an example computed with my script. A simple, uniaxial, stretch can transform a close packed hexagonal lattice into a non-close packed hexagonal lattice. This was unintuitive to us. In fact, every Bravais lattice type is accessible from a non-close packed hexagonal lattice by uniaxial stretching.

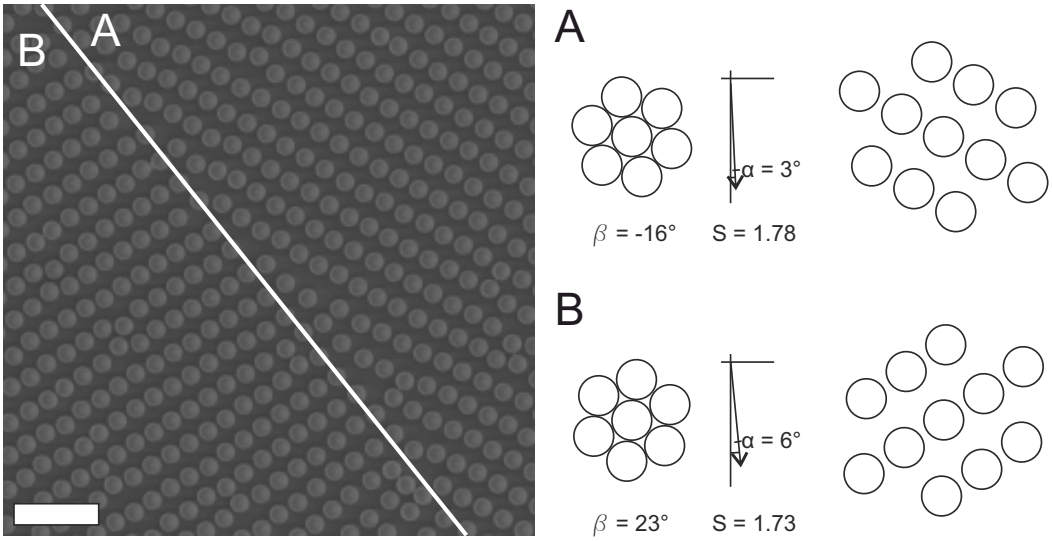


Fig. 7.4.: SEM images of non-close packed domains. The transfer direction is vertical. Both domains are from the same sample. Therefore, we assume that they experienced the same transformation. Our theory predicts that both domains are stretched by a very similar amount (1.73-1.78) in the same direction as the transfer. The only difference is the relative orientation of the original close packed hexagonal domains. The original domains were rotated relative to each other by around 30°. Scale bar is 2 µm. Adapted from the PhD thesis of Miriam E. J. Hummel.[120]

We had to proof that the lattice transformation is an uniaxial stretch. We used the following line of reasoning:

The monolayers contain domains with different lattice orientations. Note, that we had no control over the orientation of the close packed hexagonal lattice during the transfer. We assume that all domains experience the same transformation. We analyzed two domains on the same substrate, assuming the transformation is an uniaxial stretch. Figure 7.4 shows two non-close packed domains on the same sample and the corresponding analysis. We found that:

- Both domains experience very similar stretch factors and directions.
- The stretch direction is parallel to the transfer direction (vertical).
- The original lattices are rotated by 30° relative to each other. This is common in polycrystalline close packed colloidal monolayers.[121]

From these results, we concluded that the transformation is indeed an uniaxial stretch in the transfer direction.

We now move from colloidal particles that are arranged in 2D monolayers to 3D assemblies. These assemblies can be ordered crystals or unordered glasses. Both types find applications ranging from photonic crystals[122] to thermal insulation materials[123] and sensors.[124] Their function is strongly coupled to their structure. The standard tool to analyze their structure is the radial distribution function, $g(r)$, also called pair correlation function.[125]

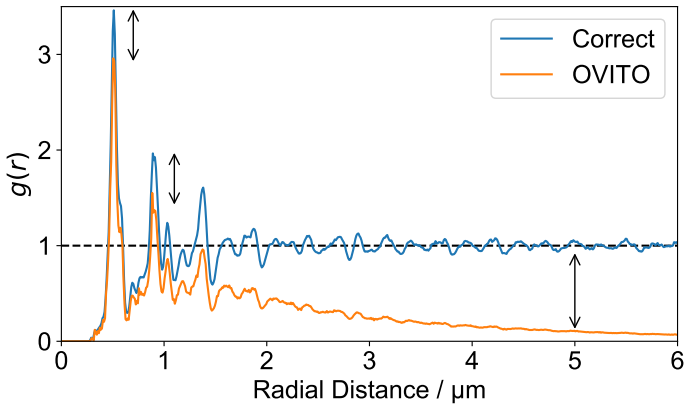


Fig. 7.5.: Comparison of two radial distribution functions computed from the same set of particle coordinates. Note that $g(r)$ computed with OVITO[126] artificially decays to zero at large radii. Adapted from Kopera *et al.*[127]

I helped my colleague Astrid Rauh, to extract particle positions from confocal microscopy images of colloidal crystals. We then used the radial distribution function, computed with OVITO,[126] to analyze these particle positions (see Figure 7.5). The resulting $g(r)$ showed an *artificial* decay to zero at large radii. We realized that this artifact was caused by the small, finite sample volumes.

Building on my work with Astrid, I developed a new algorithm to compute the radial distribution function in finite volumes (see chapter 11). This was an unsolved problem for over 35 years.[1]

Important milestones during this project were:

1. Automatic extraction of 3D particle positions from confocal images
2. Understanding the origin of the $g(r)$ decay to zero at large radii
3. Derivation of the analytic solution for the sphere - box intersection volume
4. Discovery of a natural upper bound for the radial distance
5. Proof that the computational requirements are similar to known algorithms

[1] see Larsen and Shaw[43] on page 4264, left column, fourth paragraph.

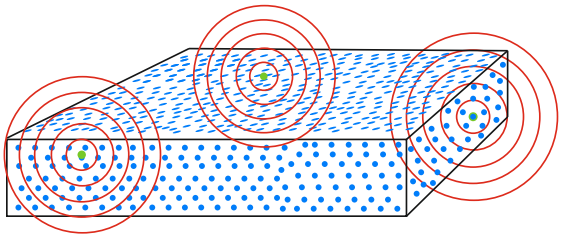


Fig. 7.6.: Particles (blue) confined in a finite, rectangular sample volume. The red shells are bins for the local radial distribution function around individual particles. Note that parts of the shell extend beyond the sample boundary. Ignoring the finite sample boundaries causes the artificial decay of $g(r)$ to zero at large radii. Adapted from Kopera *et al.*[127]

The radial distribution function is, in essence, a histogram of the average normalized radial particle density. Figure 7.6 show the radial bins (red circles) around some selected particles. We compute the particle density inside each radial bin by dividing the number of particles in the bin through the bin (shell) volume. These bins can extend beyond the finite sample volume where no particle coordinates are known. Neglecting this fact during the computation of $g(r)$ causes an underestimated particle density and therefore, the decay to zero at large radii.

My solution addresses this issue. I use the intersection volume between a radial bin and the sample volume when I compute the particle density. I focused on rectangular cuboids (boxes) as the sample volume. However, my approach works for arbitrary sample shapes. I derived an analytic solution for the intersection volume between a shell and a box. There are many different cases to consider. However, I capture all of these cases with analytic equations and if-else logic (see SI in chapter 11).

The essential part of my algorithm is the analytic equation for a subvolume of a sphere. Imagine two perpendicular planes cutting into a sphere centered around the origin. The planes are a distance a and b, respectively, away from the origin. The volume of the smallest wedge is then given by

$$\begin{aligned} V_{\text{cut}}(R_{\text{s}}, a, b) = \frac{R_{\text{s}}^3}{6} \cdot \left[\pi - 2 \cdot \arctan\left(\frac{a \cdot b}{R_{\text{s}} \cdot \sqrt{R_{\text{s}}^2 - a^2 - b^2}}\right)\right] \\ + \frac{1}{2} \cdot \left[\arctan\left(\frac{a}{\sqrt{R_{\text{s}}^2 - a^2 - b^2}}\right) - \frac{\pi}{2}\right] \cdot \left[R_{\text{s}}^2 \cdot b - \frac{b^3}{3}\right] \\ + \frac{1}{2} \cdot \left[\arctan\left(\frac{b}{\sqrt{R_{\text{s}}^2 - a^2 - b^2}}\right) - \frac{\pi}{2}\right] \cdot \left[R_{\text{s}}^2 \cdot a - \frac{a^3}{3}\right] \\ + \frac{a \cdot b \cdot \sqrt{R_{\text{s}}^2 - a^2 - b^2}}{3}, \text{ with } a, b \geq 0, \text{ and } R_{\text{s}}^2 > a^2 + b^2 \end{aligned} \tag{7.1}$$

with R_{s} as the sphere radius. This analytic solution gives my approach a superior computational efficiency compared to numeric integration.

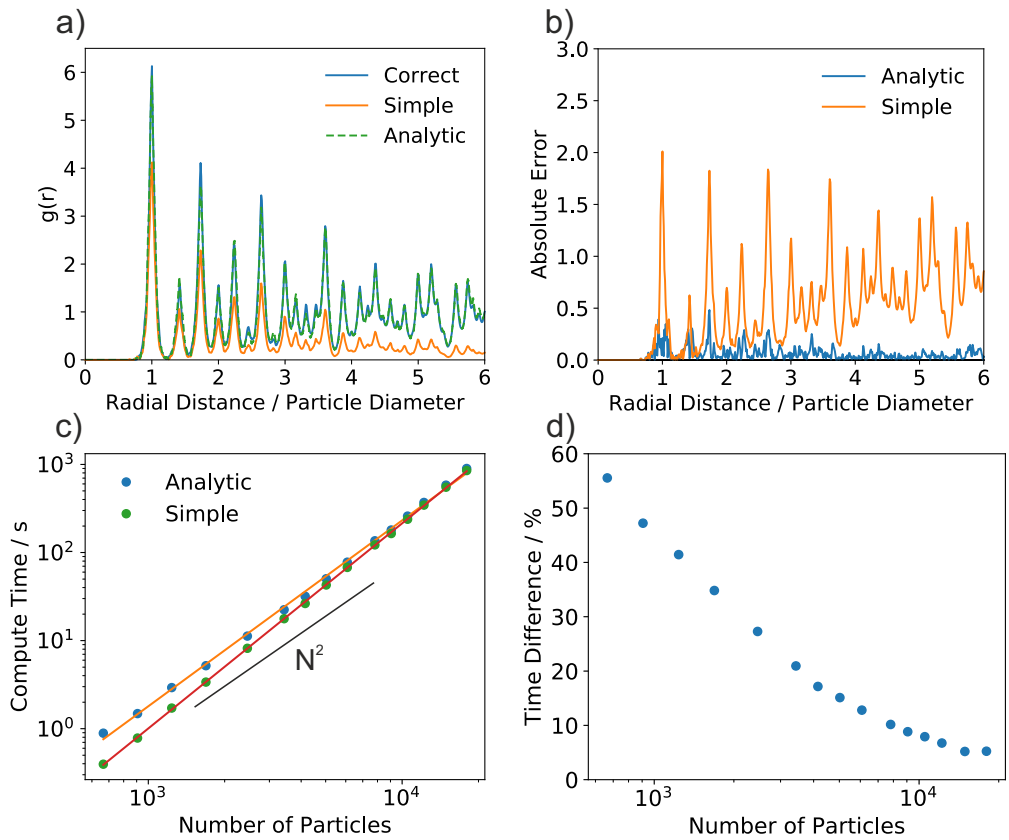


Fig. 7.7.: a) Radial distribution functions computed from an FCC test dataset with three different methods. The correct (blue) $g(r)$ originates from a larger dataset where the radial bins never extend beyond the sample volume. The simple (orange) $g(r)$ ignores the finite nature of the sample volume. My analytic algorithm produces an $g(r)$ that is almost indistinguishable from the correct $g(r)$. The absolute error (b) reinforces this result. c) Comparison of the compute time for the simple algorithm and my new analytic algorithm as a function of particle number, N. Both have a computation complexity of $\mathcal{O}(N^2)$. My analytic algorithm is slightly slower. However, the relative time difference (d) becomes negligible for large datasets. Adapted from Kopera *et al.*[127]

I tested both the accuracy and computational complexity of my new analytic algorithm. The results are shown in Figure 7.7. My analytic algorithm produces a $g(r)$ that is almost indistinguishable from the correct $g(r)$. The absolute error for my algorithm vanishes in the noise background as compared to the large error from the simple algorithm. The error decreases for large radial distances in contrast to the simple algorithm. One concern regarding my new algorithm was the computational efficiency. My new algorithm is slightly slower than the wrong, simple algorithm (see Fig. 7.7c). However, the computational complexity is $\mathcal{O}(N^2)$, similar to the simple algorithm. This means, that the absolute runtime difference becomes negligible for large datasets with many particles (see Fig. 7.7d).

Building and analyzing particulate systems was an essential part of the following two projects. We asked the question how the particle type and arrangement influences heat transport in colloidal assemblies. To answer these questions, we first synthesized binary colloidal glasses that consisted of solid polymer and hollow silica colloids with similar size. Our goal was to understand the influence of the mixing ratio on heat flux (see chapter 12). Therefore, we measured their thermal conductivity with Xenon flash analysis. In addition, we simulated heat flux in binary colloidal glasses with COMSOL Multiphysics.[91] My interface to COMSOL allowed us to simulate the effective thermal conductivity of our colloidal glasses.

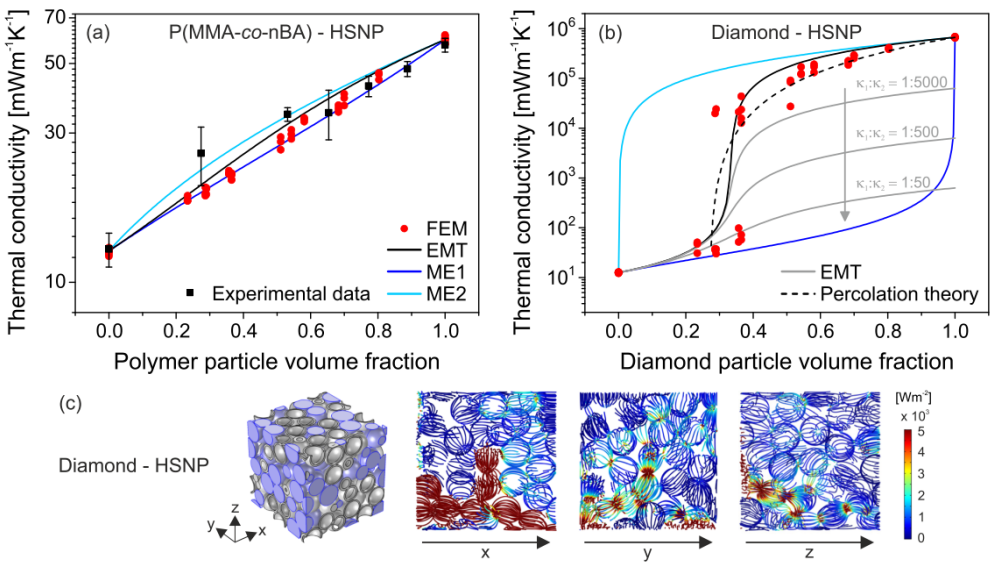


Fig. 7.8.: a) Thermal conductivity as a function of polymer particle volume fraction. Both finite element simulations (FEM) and mixing models describe the influence of the mixing ration on the thermal conductivity well. b) Simulated thermal conductivity for a mixture of diamond and hollow silica particles. Effective medium theory (EMT) is able to describe the percolation threshold around 30 % diamond particles. c) Thermal percolation paths emerging in a colloidal glass made from diamond and hollow silica particles. Note the strong heat flux through diamond particles in the x-direction. Reproduced from Ruckdeschel *et al.*[49] with permission from the American Physical Society.

Figure 7.8 shows the results of our COMSOL simulations and the experiments. The data show a linear change in the thermal conductivity as a function of the solid sphere volume fraction. Effective medium theory[128],[129] and different mixing models describes the influence of the solid sphere volume fraction on the effective thermal conductivity well (see Figure 7.8a). Percolation theory[130] is able to describe the onset of percolation in our simulations. However, only a very high thermal conductivity contrast, on the order of 1:500, would create a measurable percolation threshold. Our results add to the understanding of heat transport in binary particulate systems.

Colloidal glasses are random colloidal assemblies. They can be prepared reproducibly by mixing particles of different sizes. We hypothesized that the amorphous structure of colloidal glasses creates superior thermal insulation. Therefore, we prepared a series of colloidal glasses with different ratios of large and small particles (see chapter 13). Martin Dulle conducted molecular dynamic simulations to gain insight into the structure of a binary colloidal glass.

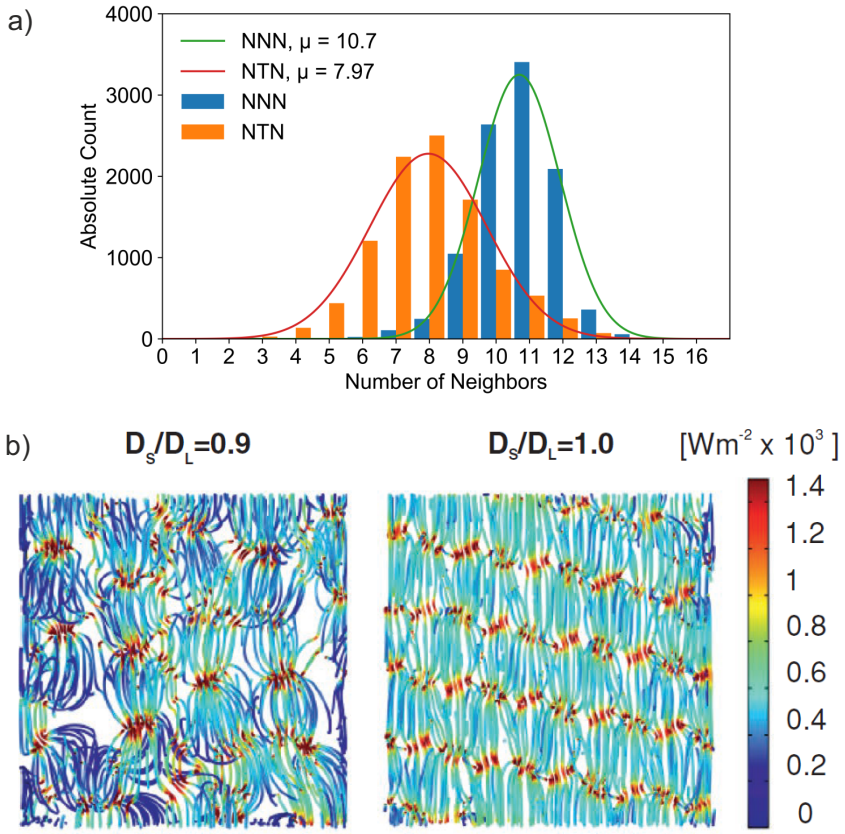


Fig. 7.9.: a) Histograms of next neighbors (NNN) and touching neighbors (NTN) for a binary mixture of 80 % small and 20 % large particles. The particle size ratio is 0.79. On average, there are more next neighbors than touching neighbors. This indicates that a significant amount of particles is close to each other, but they do not touch. b) Thermal pathways in a colloidal glass (left) and a colloidal crystal (right). The random jamming in the glass increases the conduction paths for heat. Reproduced from Nutz *et al.*[48] with permission from John Wiley and Sons.

My analysis yielded the distribution of next neighbors and touching neighbors. An exemplary histogram for a binary mixture of 80 % small and 20 % large particles is shown in Figure 7.9. There are at least two non-contact neighbors inside the first coordination sphere. This reduces the number of connections between the particles and increases the thermal path length. We confirmed the presence of tortuous paths with COMSOL simulations (see Figure 7.9b).

We discovered that an addition of ≈ 10 % of particles with different size destroys the colloidal crystallization. The resulting colloidal glass has a lower density and extended thermal pathways. This in turn reduces the effective thermal conductivity by ≈ 50 %. Our findings provide new design guidelines for thermal insulation materials.

The two previous projects employed Xenon flash analysis (XFA) as a tool to assess the thermal diffusivity in three-dimensional colloidal crystals. I gave a brief description of XFA in section 6.4. XFA is a well established method for monolithic samples of appropriate size and thickness. However, XFA is not suited for thin films, in-plane thermal transport, or anisotropic thermal conductivities. Therefore, we built and acquired additional thermal transport metrology to characterize nanostructured materials. I will focus on thermography, and in particular, lock-in thermography, since this is the area where I contributed most. I will first present two recent publications that rely on results from (lock-in) thermography. I then highlight my contributions to method development around lock-in thermography.

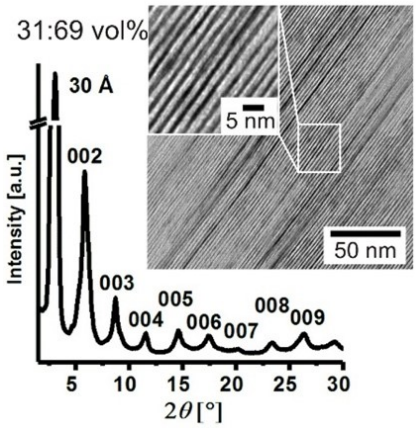


Fig. 7.10.: Structure of a Hectorite/PVP Bragg stack. The TEM crosssections show the large scale arrangement of clay nanosheets into perfectly ordered stacks. Small angle x-ray scattering reveals diffraction peaks up to the 9th order. This further underlines the high degree of order present in the samples. Reproduced from Wang *et al.*[97]

Synthetic clays have been developed into a versatile material class.[131]–[134] These clays can be produced in bulk quantities with platelets larger than 10 µm. Delaminating the clay down to sheets with intrinsic thickness (1.0 nm) creates individual nanosheets with aspects ratios[135] larger than $2 \cdot 10^4$. This large aspect ratio forces the nanosheets into almost perfect parallel stacks when the dispersion dries. We used the nanosheet dispersions to create 1D hybrid Bragg stacks. We choose polyvinylpyrrolidon (PVP) as the filler between the nanosheets. This ensures a large contrast in both the thermal conductivity and the mechanical properties of the layers.

We used lock-in thermography to measure the isotropic in-plane thermal diffusivity and the photoacoustic method[136],[137] for the through plane thermal diffusivity. In addition, Brillouin light scattering[138] was used to probe phonons in the GHz range. These measurements provide information on the direction dependent sound velocities, which ultimately yield the full elastic stiffness tensor of this hybrid material.

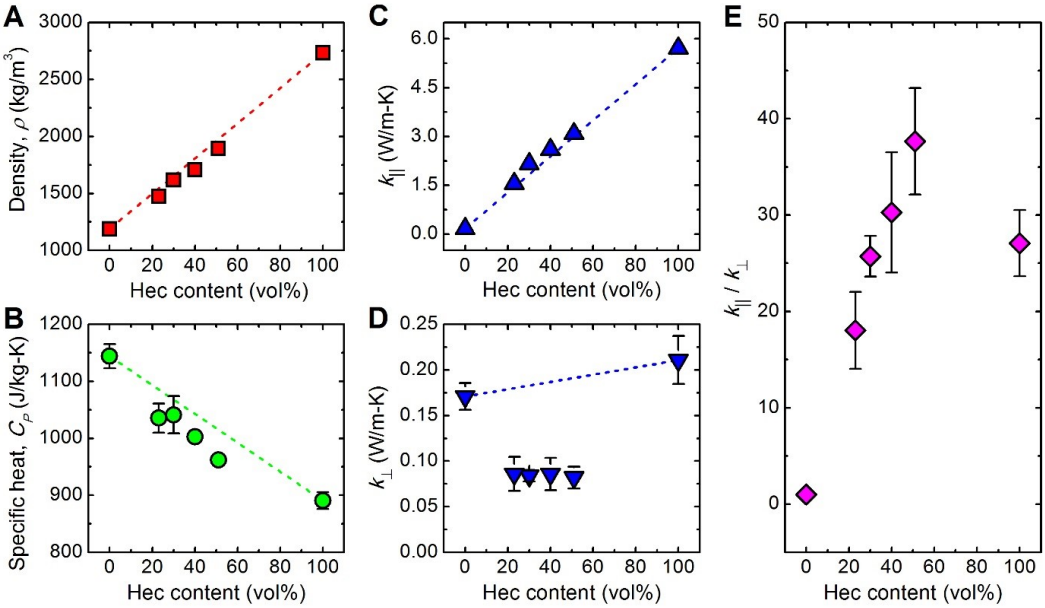


Fig. 7.11.: The density (a), the specific heat capacity (b), and the in-plane thermal conductivity (c) of our composites are a linear function of the hectorite content. In contrast, the thermal conductivity through the hectorite/PVP stack shows a broad minimum for all mixtures compared to the pure components. The thermal conductivity anisotropy (e) reaches values as high as 40 for a 1:1 volume mixture of hectorite and PVP. Reproduced from Wang *et al.*[97]

Measuring the in-plane thermal conductivity efficiently and reproducible was an important task in this project. The data analysis procedure that I outlined in chapter 6.6.6 fulfilled this requirement. Based on my development, we found that the in-plane thermal diffusivity is 40 times higher than the through plane thermal diffusivity. This is an exceptionally large anisotropy for a dielectric material.[139] The strong anisotropy between the in-plane and through plane direction is mirrored in the mechanical properties. In addition, the through plane thermal conductivity ($0.09\,{}^{\mathrm{W}}/_{\mathrm{mK}}$) is very low for a bulk material.

Another interesting and potentially anisotropic material class are electrospun nonwovens. They are created by spinning a polymer solution from a needle with the help of an electric field. The fibers in these nonwovens are usually randomly oriented and form a porous network. Applications of nowovens range from filters[140] to superhydrophobic surfaces[141] and reinforcement for composite materials.[142] We use nonwovens, made from thermoplastic polyurethane (TPU), to create smart

textiles. Polyurethane fibers are flexible and elastic. Those properties are preserved in the composite nonwovens. We add functionality by filtering a dispersion of silver nanowires directly through the nonwoven. This creates a thin, electrically conductive layer on top that can be used as a strain sensor or an electric conductor. We protect the silver nanowire layer by electro spinning a second layer of TPU on top.

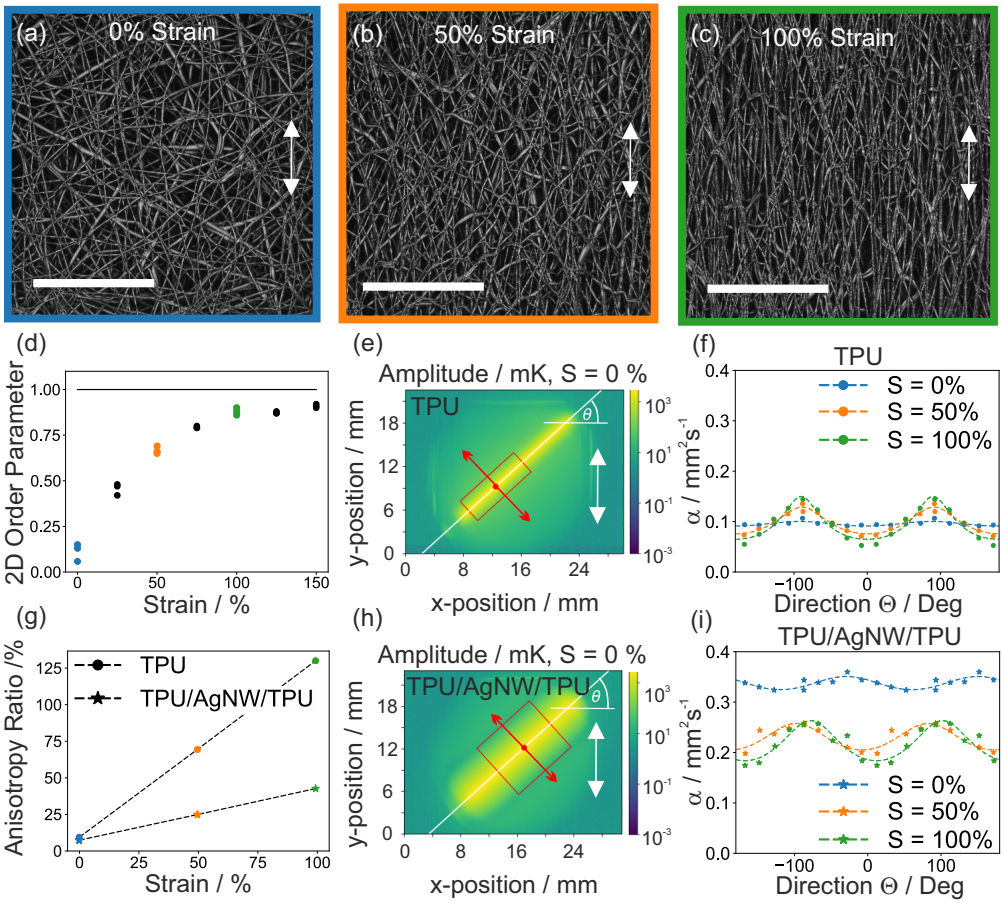


Fig. 7.12.: (a-c) Laser scanning microscopy images of thermoplastic polyurethane nonwovens under different amounts of strain. (d) 2D order parameter computed from images like (a-c). The fibers align almost perfectly above 100 % strain. (e, h) Amplitude of the thermal oscillations during a lock-in thermography measurement. The white line represents the location of the line laser. Changing the angle θ gives us access to the direction dependent thermal diffusivity (f, i). (g) Ratio of the thermal diffusivity in the major and minor axis of the ellipse fits in (f, i). Note that the anisotropy changes linearly for both pure thermoplastic polyurethane fibers and with added silver nanowires on top. However, the pure polymer fibers show a much stronger increase in anisotropy. Reproduced from Gao *et al.*(doi: 10.1002/adfm.201907555)

We used lock-in thermography and laser scanning microscopy to probe the anisotropic alignment of TPU fibers during stretching. The laser scanning microscopy images (a-c) show an increasing alignment of the TPU fibers in the stretching direction. I quantified the alignment with the 2D order parameter, S_{2D}, computed from these

images (Fig. 7.12d). S_{2D} approaches 1, at a strain of around 100 %. This indicates near perfect alignment of the fibers. Lock-in thermography with a newly implemented line laser allows us to probe heat flux in every in-plane direction. Heat travels faster along a fiber than between the fibers. The anisotropy in the fiber orientation is therefore mirrored as an anisotropic heat flux (Fig. 7.12f, i).

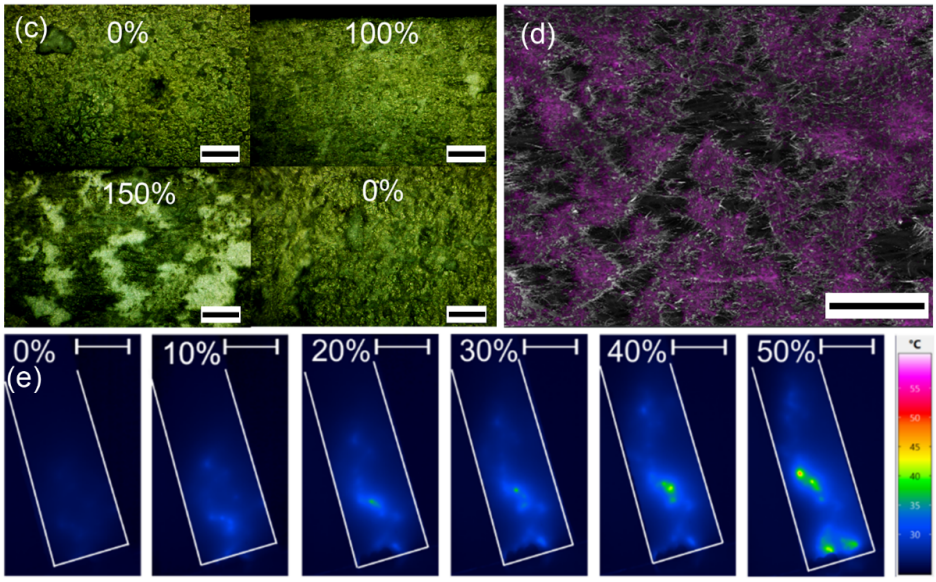


Fig. 7.13.: c) Optical microscopy images of TPU and silver nanowire composite nonwovens. The light and dark regions at increasing strain are created by cracks in the silver nanowire layer. d) SEM image of a cracked silver nanowire layer ontop of a TPU nonwoven. The violet color marks silver deposits measured by EDX. e) Thermograms of our composite films. The sample is heated with a constant current. We can visualize the bottlenecks as hot spots due to their larger heating power. Reproduced from Gao *et al.*(doi: 10.1002/adfm.201907555)

In addition to the thermal transport along the polymer fibers the electrical conductivity along the silver nanowire network in this composite material is important. The silver nanowire layer gradually looses its initially high electric conductivity during stretching. We found that stretching creates cracks in the silver nanowire layer. We confirm the crack formation both with optical microscopy (Fig. 7.13c) and scanning electron microscopy coupled with energy dispersive x-ray spectroscopy (Fig. 7.13d). The cracks create bottlenecks for an electric current. A constriction with higher electric resistance manifests itself as a hot spot. These hot spots dynamically change during stretching. Thermography allowed us to look at hot spots in the otherwise encapsulated silver nanowire layer. We think that such conductive and breathable nonwovens find application as both strain sensors and current conductors in smart textiles.

The previous two publications showed that lock-in thermography is a valuable technique for thermal characterization. Structural anisotropies can be revealed by measuring heat flux in different directions. One of the primary advantages of lock-in thermography is the simple data analysis called *the slope method* invented by Zhang *et al.*[115] and Fabbri *et al.*[114] We extended and validated their approach to semi-transparent samples. We discovered that the slopes method is still valid for thermally thin films even if they are translucent to both visible and IR radiation. Consequently, lock-in thermography is a robust characterization technique that can be applied to a wide range of materials.

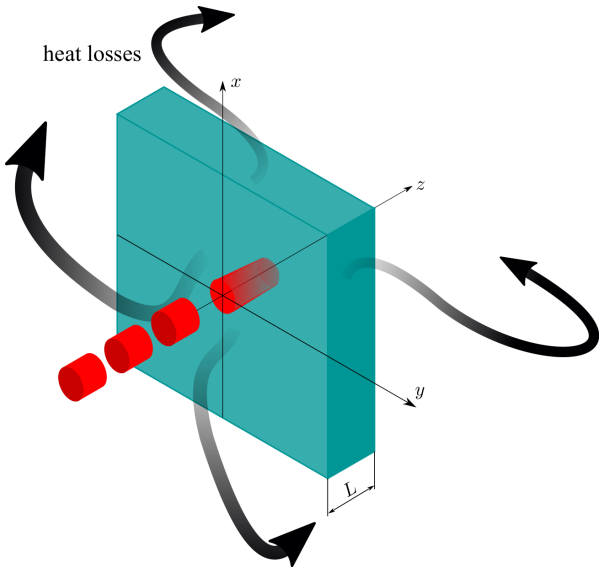


Fig. 7.14.: Model for a semi-transparent film excited by a modulated point laser in the center. The film is translucent for both the laser wavelength and infrared radiation. This potentially complicates the analysis of lock-in thermography measurements. Black arrows indicate possible heat losses through radiation on both sides. Reproduced from Philipp *et al.*[116] with permission from American Chemical Society.

Our analytic, 2D heat conduction model (see Figure 7.14) includes heat losses on both sides by radiation and convection, a Lambert-Beer type absorption of the laser beam in the sample, and heat transport by thermal radiation in the sample. In addition to theoretical modeling, we prepared and measured thin polymer films with different absorption coefficients and thicknesses. We varied the absorption by melt blending phenol red into PMMA and LDPE. Phenol red absorbs both the laser light with a wavelength of 488 nm and infrared radiation in the range where our IR camera is sensitivity. Figure 7.15 shows the results of our validation experiments. We observed no dependence of the measured thermal diffusivity on the dye content. Therefore, we concluded that the changing laser absorption and radiative thermal transport have no influence on the lock-in thermography measurements.

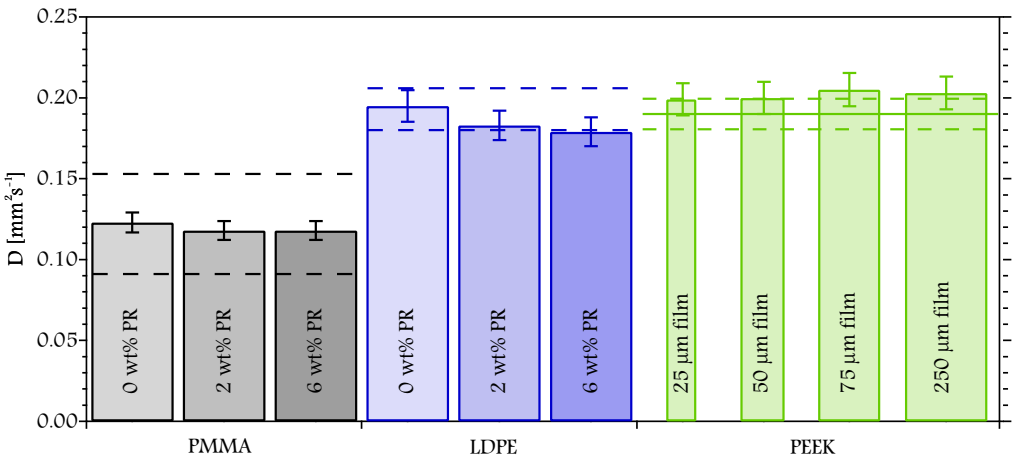


Fig. 7.15.: Thermal diffusivity measured by lock-in thermography for three different polymers. The dashed lines represent the accepted literature values. We controlled the laser absorption by blending different amounts of Phenol Red (PR) into the films. We tested the influence of the film thicknesses with the PEEK samples. All measured thermal diffusivities are well within the accepted literature values. Adapted from Philipp *et al.*[116] with permission from the American Chemical Society.

My latest project (see chapter 17) addresses DC heating in frequency domain thermal characterization methods. DC heating is an inevitable side-effect when the sample can only be heated and not cooled. This increases the average measurement temperature. The measured thermal properties are changed because they generally depend on temperature. In addition, the DC heating wastes energy and therefore reduces the signal efficiency. My new signals reduce DC heating and allow us to measure multiple lock-in frequencies simultaneously. This concept is very likely applicable to other characterization techniques operating in the frequency domain. I present two optimized excitation signals: a square wave with shorter duty cycle and a sum of sine waves. The square wave can be represented by its Fourier series as

$$I(t) = I_0 \cdot \left[D + \sum_{n=1}^{\infty} 2 \cdot \frac{\sin(\pi n D)}{\pi n} \cdot \cos\left(\frac{2\pi n t}{T}\right) \right]. \tag{7.2}$$

We can use the higher harmonics to measure lock-in thermography at multiple frequencies simultaneously. Reducing the duty cycle directly reduces the DC heating. The sum of sine waves is given by

$$I(t) = S + \sum_{n=1}^{N} A_n \cdot \cos(2\pi f_n \cdot t + \phi_n). \tag{7.3}$$

This signal has many degrees of freedom. I describe the different steps necessary to optimized both the signal-to-noise ratio and the DC component of such a multiplexed

signal. The key idea is to add partially negative sine waves into a fully positive signal. This interference between the sine waves reduces the DC component.

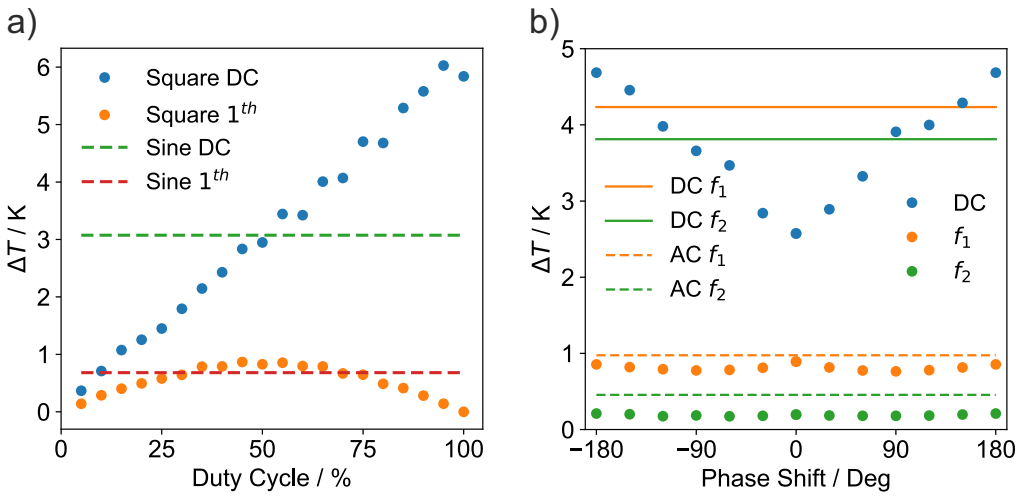


Fig. 7.16.: a) Temperature amplitudes of the DC component and 1st harmonic of a square wave as a function of duty cycle. The dashed lines represent the corresponding values for a single sine wave with equal amplitude and frequency. Note, that a square wave with 50 % duty cycle creates temperature amplitudes similar to a sine wave. Reducing the duty cycle reduces the DC component of the square wave. b) Temperature amplitudes of the DC component and 1st and 2nd harmonic of the sum of two sine waves as a function of relative phase shift. The sum of two sine waves has a minimal DC component at zero phase shift. The DC component is even smaller than for individual sine waves (solid line).

Figure 7.16a shows the measured DC heating (blue dots) for a square wave as a function of duty cycle. Reducing the duty cycle from 50 % to 20-25 % decreases the DC heating from 3 to 1 K. The relative power in the 1st harmonic doubles (see chapter 17.4.2). The power in the signal is then used more efficiently. A square wave with a duty cycle of around 20-25 % is therefore superior to a sine wave. Figure 7.16b shows the measured DC heating (blue dots) for the sum of two sine waves as a function of relative phase shift between the sine waves. A pronounced minimum occurs at a relative phase shift of 0°. The amplitudes of the two sine waves (orange and green dots) show no phase shift dependence. The DC component is reduced by about 46 % relative to a phase shift of 180°. I also show that multiplexed lock-in thermography with a square wave or the sum of four sine waves yields similar results as a sequential measurement with individual sine waves. The resulting thermal diffusivities are within a ± 2 % margin of error. We conclude that multiplexed excitation signals are beneficial for frequency domain techniques. They allow us to reduce DC heating and measure at multiple frequencies simultaneously. We think that this method will be useful for other researchers working with frequency domain heating techniques.

7.3 My Contributions to Joint Publications

7.3.1 Ordered Particle Arrays via a Langmuir Transfer Process: Access to Any Two-Dimensional Bravais Lattices

Miriam E. J. Hummel, Christian Stelling, **Bernd A. F. Kopera**, Fabian A. Nutz, Matthias Karg, Markus Retsch, Stephan Förster
We published this article in Langmuir **2018**, 35, 973-979

Miriam E. J. Hummel and Christian Stelling first observed stretching of colloidal monolayers during the water to substrate transfer. They manufactured the colloidal monolayers and characterized the samples with scanning electron microscopy and optical microscopy. I developed the *ParticleSizer* software in Matlab to extract particle sizes and positions from the microscopy images. Together with Miriam, I developed the analysis procedure for extracting unit cell parameters from colloidal monolayers. We also derived the analytic equations for phase boundaries in the direction - stretch factor phase diagrams. I developed a python script to compute the continuous direction - stretch factor phase diagrams and predict the particle arrangement in the stretched state. Fabian A. Nutz synthesized the colloidal particles. Markus Retsch and Stephan Förster supervised this project. The manuscript was written through contributions from all authors.

7.3.2 Computing the 3D Radial Distribution Function from Particle Positions: An Advanced Analytic Approach

Bernd A. F. Kopera and Markus Retsch
We published this article in Analytical Chemistry **2018**, 90, 23, 13909-13914

I identified the need for a better way to compute the radial distribution function in finite volumes. I derived the analytic solution for the intersection volume between a sphere and a rectangular box and wrote software to test and benchmark my new algorithm. I wrote the manuscript, created all figures and the supporting information. Markus Retsch proofread the manuscript and guided this paper through the submission and peer-review process.

7.3.3 Low Thermal Conductivity through Dense Particle Packings with Optimum Disorder

Fabian A. Nutz, Alexandra Philipp, **Bernd A. F. Kopera**, Martin Dulle and Markus Retsch

We published this communication in Advanced Materials **2018**, 30, 1704910

Fabian A. Nutz and Markus Retsch conceived the project. Fabian A. Nutz synthesized the colloidal particles and glasses, characterized the structure with scanning electron microscopy and measured thermal diffusivity with xenon flash analysis. Alexandra Philipp took lead on the finite element simulations and corrected the manuscript. I analyzed particle positions computed with molecular dynamic simulations. I developed custom software to distinguish between contact and non-contact neighbors and analyze the contact area for different particle size mixtures. I helped writing the manuscript, interpret the results and prepare figures. Marting Dulle performed the molecular dynamic simulations and helped writing the manuscript. Markus Retsch supervised this project and wrote the manuscript.

7.3.4 Thermal transport in binary colloidal glasses: Composition dependence and percolation assessment

Pia Ruckdeschel, Alexandra Philipp, **Bernd A. F. Kopera**, Flora Bitterlich, Martin Dulle, Nelson Pech-May and Markus Retsch

We published this full paper in Physical Review E **2018**, 97, 022612

Pia Ruckdeschel synthesized all particles and characterized the colloidal glasses with scanning electron microscopy. She prepared the manuscript together with Alexandra Philipp. Finite element simulations using COMSOL Multiphysics were performed by Alexandra Philipp and Flora Bitterlich. I supported these simulations with Python software that generated the 3D mixtures of solid and hollow sphere particles. This included an essential interface between particle position lists and AutoCAD. The particle positions were generated by Martin Dulle using molecular dynamics simulations. I also analyzed the particle positions by computing the radial distribution function. Nelson Wilbur Pech-May reformated the manuscript in LaTeX and contributed to the scientific discussion. Markus Retsch supervised this project and corrected the manuscript.

7.3.5 Tunable thermoelastic anisotropy in hybrid Bragg stacks with extreme polymer confinement

Zuyuan Wang, Konrad Rolle, Theresa Schilling, Patrick Hummel, Alexandra Philipp, **Bernd A.F. Kopera**, Anna M. Lechner, Markus Retsch, Josef Breu, George Fytas

We published this article in Angewandte Chemie Int. Ed., 10.1002/anie.201911546

Zuyuan Wang and Konrad Rolle performed the mechanical characterization with Brillouin light scattering. Theresa Schilling prepared the composite Hectorite samples. Partick Hummel measured the through plane thermal diffusivity with the photo-accoustic method. Alexandra Philipp measured the in-plane thermal diffusivity with lock-in thermography. Alexandra Philipp and I implemented and built the lock-in thermography setup. Anna Lechner measured differential scanning calorimetry. Markus Retsch, Josef Breu, and George Fytas supervised the work and corrected the manuscript.

7.3.6 Breathable and flexible polymer membranes with mechanoresponsive electric resistance

Qiang Gao, **Bernd A. F. Kopera**, Jian Zhu, Xiaojian Liao, Chao Gao, Markus Retsch, Seema Agarwal, Andreas Greiner

Our manuscript (adfm.201907555) is accepted in Advanced Functional Materials.

Andreas Greiner conceived the concept and guided the project. Qiang Gao and Seema Agarwal designed experiments regarding sample preparation. Qiang Gao prepared all samples, and performed the initial electric and mechanic characterization. Furthermore, he contributed the electron microscopy images and prepared the first draft of the manuscript. Jian Zhu assisted in the design of the electric characterization and discussed the results. Xiaojian Liao assisted in the preparation of polycaprolactone short fibers and discussed the results. I analyzed the fiber orientation in the nonwovens from optical microscopy images to obtain the strain dependent order parameter. I measured the direction and strain dependent thermal diffusivity with my self built line laser lock-in thermography setup. I characterized the strain dependent ohmic heating in the samples using thermography. I prepared some of the figures and helped writing the manuscript. Chao Gao discussed and evaluated the work. Markus Retsch, Seema Agarwal, and Andreas Greiner discussed with the other coauthors, evaluated the work and revised the first draft of the manuscript.

7.3.7 Direct measurement of the in-plane thermal diffusivity of semitransparent polymers by lock-in thermography: an extension of the slopes method

by Alexandra Philipp, Nelson W. Pech-May, **Bernd A. F. Kopera**, Anna Lechner, Sabine Rosenfeldt, Markus Retsch

We published this article in Analytical Chemistry **2019**, 91, 13, 8476-8483

Alexandra Philipp prepared all samples. She also measured lock-in thermography to obtain the thermal diffusivity and characterized the optical properties with infrared and optical spectroscopy. Nelson W. Pech May derived the theoretical framework. I developed the software to efficiently analyze lock-in thermography measurements. I prepared and tested the pixel size calibration procedure for thermographic imaging. I helped developing the vacuum chamber and build the laser setup. I also measured part of the thin film samples with lock-in thermography. Anna Lechner measured differential scanning calorimetry on all samples. Sabine Rosenfeldt measured small angle x-ray scattering and analyzed the corresponding data. The manuscript was written through contributions from all authors.

7.3.8 Reducing DC heating in lock-in thermography: the advantage of thermal multiplexing

Bernd A. F. Kopera and Markus Retsch

in preparation

I developed the idea to reduce DC heating in frequency domain techniques by destructive interference of partially negative excitation signals. I built the lock-in thermography setup, programed the numeric exploration of the phase parameter space. I also developed a method to measure the DC temperature rise and used this method to characterize the different excitation signals. I wrote the manuscript, created all figures and the supporting information. Markus Retsch proofread the manuscript and provided valuable guidance during the inception of this idea.

8 Outlook

There are many more project ideas that originated during my PhD. I use this last chapter to outline some of the promising ideas for future work. These projects will all depend on future development and improvement of tailor-made code for the measurement and analysis of upcoming material systems.

8.1 Lock-In Thermography

- **Combine lock-in thermography with laser flash analysis.** This would allow us to obtain in-plane and through-plane thermal diffusivity on the same sample. This project requires: an IR camera with a higher frame rate due to the fast thermal equilibration in the thin films necessary for lock-in thermography and an intense pulsed light source with an illumination are of around 1 cm^2. The challenge is to balance two mutually exclusive goals: A short pulse length and large thermal energy deposited in the sample.

- **Extend fiber lock-in thermography to smaller fiber diameters.** Extending the characterization of non-woven materials in chapter 15 requires thermal transport characterization down to individual fibers. This would additionally open the possibility to understand highly interesting materials such as: spider silk threads, carbon fiber rovings, electrospun polymer fiber assemblies, and animal hair with good thermal insulation. This project requires a smaller pixel size in the IR images. Otherwise, the partial cover of the pixels by the fiber distorts the temperature measurement. Furthermore, the working distance for high resolution IR microscope objectives is small. Redesigning the vacuum chamber with a thinner IR window and a sample mount close to the window is therefore necessary. Thermal losses for such high surface area materials may add an additional source of complication.

- **Implement thermal *conductivity* measurements.** Pure lock-in thermography can only measure thermal diffusivity because only the temperature distribution is recorded. Measuring thermal conductivity requires a known heat flux through the sample. The publication on line laser lock-in thermography by Brendel *et al.*[73] includes an extension for measuring thermal conductivity. This extension requires precise knowledge of the absorbed laser

power in the sample. We could calibrate the absorbed power by measuring reference materials with known thermal conductivity. Another way is to replace the line laser with a line heater, a thin metallic line on the sample surface with a known resistance. Applying a known AC current to this line generates a precise heating power.

8.2 Particle Position Analysis

- **Analyze the internal structure of different colloidal assemblies.** Colloidal particles can be arranged by a variety of techniques: evaporation induced self-assembly,[46] filtration,[49] and dip coating.[143] In addition, destroying the colloidal stability might lead to fractal structures.[144] Mixing colloidal particles with different sizes also creates glassy assemblies.[48] We could compare the internal structure of the resulting colloidal assemblies. The experimental challenge lies in determining the particle positions e.g. from fluorescent confocal microscopy images. Afterwards, the colloidal particle network could be analyzed with graph theory.[145]
- **Investigate the effect of missing particles.** Sometimes, particle positions are missing in a dataset. This might cause problems during the radial distribution function computation. It is therefore important to develop a robust framework for error estimation in this case.
- **Derive analytic intersection volumes for different sample shapes.** My work on $g(r)$ was recently cited by Catarineu *et al.*[7] Their particle set (helium bubbles) is bounded by a cone. My algorithm is still applicable once the analytic intersection volume between a sphere and a cone is known.
- **Analyze the effect of the user defined bin size in $g(r)$.** The radial distribution function is in essence a histogram. The bin width is choosen by the researcher. A large bin width averages $g(r)$ whereas a small bin width increases the noise. There is currently no rule available to choose the bin width. Quantifying the effects of a finite bin width would therefore be an interesting research project.
- **Develop image analysis for elliptical particles.** The challenge is the efficient detection and separation of elliptic particle boundaries. Spherical particles have a high symmetry. This facilitates boundary detection. Elliptical particles have two important characteristics. Their position and their orientation. Extracting the orientation is important to detect liquid crystalline order.

References

9

[1] J.-M. Bergheau and R. Fortunier, *Finite Element Simulation of Heat Transfer*. ISTE, 2008.

[2] B. Kuhlman and P. Bradley, „Advances in protein structure prediction and design,“ *Nature Reviews Molecular Cell Biology*, vol. 20, no. 11, pp. 681–697, 2019.

[3] A. A. Emery and C. Wolverton, „High-throughput DFT calculations of formation energy, stability and oxygen vacancy formation energy of ABO_3 perovskites,“ *Scientific Data*, vol. 4, no. 1, 2017.

[4] K. Yim, Y. Youn, M. Lee, *et al.*, „Computational discovery of p-type transparent oxide semiconductors using hydrogen descriptor,“ *npj Computational Materials*, vol. 4, no. 1, 2018.

[5] V. Tshitoyan, J. Dagdelen, L. Weston, *et al.*, „Unsupervised word embeddings capture latent knowledge from materials science literature,“ *Nature*, vol. 571, no. 7763, pp. 95–98, 2019.

[6] C. A. L. Bailer-Jones, J. Rybizki, M. Fouesneau, G. Mantelet, and R. Andrae, „Estimating distance from parallaxes. IV. distances to 1.33 billion stars in gaia data release 2,“ *The Astronomical Journal*, vol. 156, no. 2, p. 58, 2018.

[7] N. R. Catarineu, N. C. Bartelt, J. D. Sugar, *et al.*, „Three-dimensional maps of helium nanobubbles to probe the mechanisms of bubble nucleation and growth,“ *The Journal of Physical Chemistry C*, vol. 123, no. 31, pp. 19 142–19 152, 2019.

[8] R. A. Shaw, „Particle-turbulence interactions in atmospheric clouds,“ *Annual Review of Fluid Mechanics*, vol. 35, no. 1, pp. 183–227, 2003.

[9] T. S. Majmudar and R. P. Behringer, „Contact force measurements and stress-induced anisotropy in granular materials,“ *Nature*, vol. 435, no. 7045, pp. 1079–1082, 2005.

[10] A. van Blaaderen and P. Wiltzius, „Real-space structure of colloidal hard-sphere glasses,“ *Science*, vol. 270, no. 5239, pp. 1177–1179, 1995.

[11] A. Soper, „The radial distribution functions of water and ice from 220 to 673 K and at pressures up to 400 MPa,“ *Chemical Physics*, vol. 258, no. 2-3, pp. 121–137, 2000.

[12] V. P. Voloshin, G. G. Malenkov, and Y. I. Naberukhin, „Long-time correlations in the diffusive motion of liquid argon atoms," *Journal of Structural Chemistry*, vol. 55, no. 6, pp. 1097–1098, 2014.

[13] S. Slomkowski, J. V. Alemán, R. G. Gilbert, *et al.*, „Terminology of polymers and polymerization processes in dispersed systems (IUPAC recommendations)," *Pure and Applied Chemistry*, vol. 83, no. 12, pp. 2229–2259, 2011.

[14] N. Vogel, S. Utech, G. T. England, *et al.*, „Color from hierarchy: Diverse optical properties of micron-sized spherical colloidal assemblies," *Proceedings of the National Academy of Sciences*, vol. 112, no. 35, pp. 10 845–10 850, 2015.

[15] M. T. Barako, A. Sood, C. Zhang, *et al.*, „Quasi-ballistic electronic thermal conduction in metal inverse opals," *Nano Letters*, vol. 16, no. 4, pp. 2754–2761, 2016.

[16] D. K. Rhee, B. Jung, Y. H. Kim, *et al.*, „Particle-nested inverse opal structures as hierarchically structured large-scale membranes with tunable separation properties," *ACS Applied Materials & Interfaces*, vol. 6, no. 13, pp. 9950–9954, 2014.

[17] F. Yan, A. Ding, M. Gironès, *et al.*, „Hierarchically structured assembly of polymer microsieves, made by a combination of phase separation micromolding and float-casting," *Advanced Materials*, vol. 24, no. 12, pp. 1551–1557, 2012.

[18] D. V. Talapin, E. V. Shevchenko, M. I. Bodnarchuk, *et al.*, „Quasicrystalline order in self-assembled binary nanoparticle superlattices," *Nature*, vol. 461, no. 7266, pp. 964–967, 2009.

[19] C. Stelling, A. Mark, G. Papastavrou, and M. Retsch, „Showing particles their place: Deterministic colloid immobilization by gold nanomeshes," *Nanoscale*, vol. 8, no. 30, pp. 14 556–14 564, 2016.

[20] J. Wang, U. Sultan, E. S. A. Goerlitzer, *et al.*, „Structural color of colloidal clusters as a tool to investigate structure and dynamics," *Advanced Functional Materials*, p. 1 907 730, 2019.

[21] D. Gueon, D.-Y. Kang, J. S. Kim, *et al.*, „Si nanoparticles-nested inverse opal carbon supports for highly stable lithium-ion battery anodes," *Journal of Materials Chemistry A*, vol. 3, no. 47, pp. 23 684–23 689, 2015.

[22] L. Mahler, A. Tredicucci, F. Beltram, *et al.*, „Quasi-periodic distributed feedback laser," *Nature Photonics*, vol. 4, no. 3, pp. 165–169, 2010.

[23] N. Vogel, C. K. Weiss, and K. Landfester, „From soft to hard: The generation of functional and complex colloidal monolayers for nanolithography," *Soft Matter*, vol. 8, pp. 4044–4061, 15 2012.

[24] S. G. Romanov, N. Vogel, K. Bley, *et al.*, „Probing guided modes in a monolayer colloidal crystal on a flat metal film,“ *Physical Review B*, vol. 86, p. 195 145, 19 2012.

[25] B. Ai, Y. Yu, H. Möhwald, G. Zhang, and B. Yang, „Plasmonic films based on colloidal lithography,“ *Advances in Colloid and Interface Science*, vol. 206, pp. 5–16, 2014.

[26] J. C. Crocker and D. G. Grier, „Methods of digital video microscopy for colloidal studies,“ *Journal of Colloid and Interface Science*, vol. 179, no. 1, pp. 298–310, 1996.

[27] M. von Ardenne, „Das elektronen-rastermikroskop,“ *Zeitschrift für Physik*, vol. 109, no. 9-10, pp. 553–572, 1938.

[28] K. Volk, J. P. S. Fitzgerald, P. Ruckdeschel, *et al.*, „Reversible tuning of visible wavelength surface lattice resonances in self-assembled hybrid monolayers,“ *Advanced Optical Materials*, vol. 5, no. 9, p. 1 600 971, 2017.

[29] M. Rey, T. Yu, K. Bley, *et al.*, „Amphiphile-induced anisotropic colloidal self-assembly,“ *Langmuir*, vol. 34, no. 34, pp. 9990–10 000, 2018.

[30] M. Knoll and E. Ruska, „Das elektronenmikroskop,“ *Zeitschrift für Physik*, vol. 78, no. 5-6, pp. 318–339, 1932.

[31] P. Matvija, F. Rozboril, P. Sobotik, I. Ostadal, and P. Kocan, „Pair correlation function of a 2D molecular gas directly visualized by scanning tunneling microscopy,“ *The Journal of Physical Chemistry Letters*, vol. 8, no. 17, pp. 4268–4272, 2017.

[32] G. Binnig, C. F. Quate, and C. Gerber, „Atomic force microscope,“ *Physical Review Letters*, vol. 56, no. 9, pp. 930–933, 1986.

[33] I. Popa, R. Longtin, P. Maroni, G. Papastavrou, and M. Borkovec, „Adsorption and self-organization of dendrimers at water–solid interfaces,“ *CHIMIA International Journal for Chemistry*, vol. 63, no. 5, pp. 279–282, 2009.

[34] Y. Brasse, V. Gupta, H. C. T. Schollbach, *et al.*, „Mechanotunable plasmonic properties of colloidal assemblies,“ *Advanced Materials Interfaces*, p. 1 901 678, 2019.

[35] M. A. Fernandez-Rodriguez, R. Elnathan, R. Ditcovski, *et al.*, „Tunable 2D binary colloidal alloys for soft nanotemplating,“ *Nanoscale*, vol. 10, no. 47, pp. 22 189–22 195, 2018.

[36] Lord Rayleigh, „The theory of optical images with special reference to the microscope,“ *Philosophical Magazine*, vol. 42, pp. 167–195, 1896.

[37] E. Abbe, „Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung,“ *Archiv für mikroskopische Anatomie*, vol. 9, no. 1, pp. 413–418, 1873.

[38] F. Bossler, J. Maurath, K. Dyhr, N. Willenbacher, and E. Koos, „Fractal approaches to characterize the structure of capillary suspensions using rheology and confocal microscopy,“ *Journal of Rheology*, vol. 62, no. 1, pp. 183–196, 2018.

[39] K. Hauf and E. Koos, „Structure of capillary suspensions and their versatile applications in the creation of smart materials,“ *MRS Communications*, vol. 8, no. 02, pp. 332–342, 2018.

[40] S. Gruber, N. Vorhauer, M. Schulz, *et al.*, „Estimation of the local sublimation front velocities from neutron radiography and tomography of particulate matter,“ *Chemical Engineering Science*, vol. 211, p. 115 268, 2020.

[41] J. Wang, C. F. Mbah, T. Przybilla, *et al.*, „Magic number colloidal clusters as minimum free energy structures,“ *Nature Communications*, vol. 9, no. 1, 2018.

[42] M. Chau, B. A. F. Kopera, V. R. Machado, *et al.*, „Reversible transition between isotropic and anisotropic thermal transport in elastic polyurethane foams,“ *Materials Horizons*, vol. 4, no. 2, pp. 236–241, 2017.

[43] M. L. Larsen and R. A. Shaw, „A method for computing the three-dimensional radial distribution function of cloud particles from holographic images,“ *Atmospheric Measurement Techniques*, vol. 11, no. 7, pp. 4261–4272, 2018.

[44] M. Knudsen, „Die molekulare Wärmeleitung der Gase und der Akkommodationskoeffizient,“ *Annalen der Physik*, vol. 339, no. 4, pp. 593–656, 1911.

[45] P. Ruckdeschel, T. W. Kemnitzer, F. A. Nutz, J. Senker, and M. Retsch, „Hollow silica sphere colloidal crystals: Insights into calcination dependent thermal transport,“ *Nanoscale*, vol. 7, no. 22, pp. 10 059–10 070, 2015.

[46] F. A. Nutz, P. Ruckdeschel, and M. Retsch, „Polystyrene colloidal crystals: Interface controlled thermal conductivity in an open-porous mesoparticle superstructure,“ *Journal of Colloid and Interface Science*, vol. 457, pp. 96–101, 2015.

[47] F. A. Nutz and M. Retsch, „Interfacial and volumetric sensitivity of the dry sintering process of polymer colloidal crystals: A thermal transport and photonic bandgap study,“ *Physical Chemistry Chemical Physics*, vol. 19, no. 24, pp. 16 124–16 130, 2017.

[48] F. A. Nutz, A. Philipp, B. A. F. Kopera, M. Dulle, and M. Retsch, „Low Thermal Conductivity through Dense Particle Packings with Optimum Disorder,“ *Advanced Materials*, vol. 30, no. 14, p. 1 704 910, 2018.

[49] P. Ruckdeschel, A. Philipp, B. A. F. Kopera, *et al.*, „Thermal transport in binary colloidal glasses: Composition dependence and percolation assessment,“ *Physical Review E*, vol. 97, p. 022 612, 2 2018.

[50] Y. S. Touloukian, *Thermophysical Properties of Matter*. New York: IFI/Plenum, 1973.

[51] A. A. Penzias and R. W. Wilson, „A measurement of excess antenna temperature at 4080 Mc/s.," *The Astrophysical Journal*, vol. 142, p. 419, 1965.

[52] R. S. Mulliken, „The band spectrum of boron monoxide," *Nature*, vol. 114, no. 2862, pp. 349–350, 1924.

[53] S. Carnot, „Réflexions sur la puissance motrice du feu et sur les machines propres à développer cette puissance," *Annales scientifiques de l'École normale supérieure*, vol. 1, pp. 393–457, 1872.

[54] H. Wang, D. W. Ihms, S. D. Brandenburg, and J. R. Salvador, „Thermal conductivity of thermal interface materials evaluated by a transient plane source method," *Journal of Electronic Materials*, vol. 48, no. 7, pp. 4697–4705, 2019.

[55] D. Suh, C. M. Moon, D. Kim, and S. Baik, „Ultrahigh thermal conductivity of interface materials by silver-functionalized carbon nanotube phonon conduits," *Advanced Materials*, vol. 28, no. 33, pp. 7220–7227, 2016.

[56] D. Suh, S. Lee, C. Xu, A. A. Jan, and S. Baik, „Significantly enhanced phonon mean free path and thermal conductivity by percolation of silver nanoflowers," *Physical Chemistry Chemical Physics*, vol. 21, no. 5, pp. 2453–2462, 2019.

[57] B. Nykvist and M. Nilsson, „Rapidly falling costs of battery packs for electric vehicles," *Nature Climate Change*, vol. 5, no. 4, pp. 329–332, 2015.

[58] J. Kim, J. Oh, and H. Lee, „Review on battery thermal management system for electric vehicles," *Applied Thermal Engineering*, vol. 149, pp. 192–212, 2019.

[59] N. Martin, A. Mitros, C. Mellone, and I. Dimen, „Multi-channel and bidirectional battery management system," U.S. Patent PCT/US2018/040140.

[60] M. Hao, J. Li, S. Park, S. Moura, and C. Dames, „Efficient thermal management of Li-ion batteries with a passive interfacial thermal regulator based on a shape memory alloy," *Nature Energy*, vol. 3, no. 10, pp. 899–906, 2018.

[61] (2019). „Energieverbrauch der privaten Haushalte für Wohnen," [Online]. Available: https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Umwelt/Materialfluesse-Energiefluesse/Tabellen/energieverbrauch-haushalte.html.

[62] „Zweite Verordnung zur Änderung der Energieeinsparverordnung," *Bundesgesetzblatt*, vol. 67, p. 3951, 2013.

[63] „Kriterien für den passivhaus-, enerphit- und phi-energiesparhaus-standard," *Passivhaus Institut*, 2016.

[64] *CRC Handbook of Chemistry and Physics*. CRC Press, 2003.

[65] R. Baetens, B. P. Jelle, J. V. Thue, *et al.*, „Vacuum insulation panels for building applications: A review and beyond,“ *Energy and Buildings*, vol. 42, no. 2, pp. 147–172, 2010.

[66] S. Jennings, „The mean free path in air,“ *Journal of Aerosol Science*, vol. 19, no. 2, pp. 159–166, 1988.

[67] S. Liu, J. Duvigneau, and G. J. Vancso, „Nanocellular polymer foams as promising high performance thermal insulation materials,“ *European Polymer Journal*, vol. 65, pp. 33–45, 2015.

[68] B. Notario, J. Pinto, E. Solorzano, *et al.*, „Experimental validation of the knudsen effect in nanocellular polymeric foams,“ *Polymer*, vol. 56, pp. 57–67, 2015.

[69] A. Gasparrini, Y. Guo, M. Hashizume, *et al.*, „Mortality risk attributable to high and low ambient temperature: A multicountry observational study,“ *The Lancet*, vol. 386, no. 9991, pp. 369–375, 2015.

[70] D. P. DeMaster and I. Stirling, „Ursus maritimus,“ *Mammalian Species*, no. 145, p. 1, 1981.

[71] Y. Cui, H. Gong, Y. Wang, D. Li, and H. Bai, „A thermally insulating textile inspired by polar bear hair,“ *Advanced Materials*, vol. 30, no. 14, p. 1 706 807, 2018.

[72] DEUTER, *Hollow fiber sleeping bags*, https://www.deuter.com/de-de/shop/schlafsaecke/c27786-kunstfaserschlafsaecke, Dec. 2019.

[73] A. Wolf, P. Pohl, and R. Brendel, „Thermophysical analysis of thin films by lock-in thermography,“ *Journal of Applied Physics*, vol. 96, no. 11, pp. 6306–6312, 2004.

[74] D. N. Simavilla and D. C. Venerus, „Investigation of anisotropic thermal conductivity in polymers using infrared thermography,“ *Journal of Heat Transfer*, vol. 136, no. 11, 2014.

[75] C. Pradère, J. Goyhénèche, J. Batsale, S. Dilhaire, and R. Pailler, „Thermal diffusivity measurements on a single fiber with microscale diameter at very high temperature,“ *International Journal of Thermal Sciences*, vol. 45, no. 5, pp. 443–451, 2006.

[76] A. Mendioroz, R. Fuente-Dacal, E. Apiñaniz, and A. Salazar, „Thermal diffusivity measurements of thin plates and filaments using lock-in thermography,“ *Review of Scientific Instruments*, vol. 80, no. 7, p. 074 904, 2009.

[77] X. Huang, G. Liu, and X. Wang, „New secrets of spider silk: Exceptionally high thermal conductivity and its abnormal change under stretching,“ *Advanced Materials*, vol. 24, no. 11, pp. 1482–1486, 2012.

[78] R. Fuente, A. Mendioroz, and A. Salazar, „Revising the exceptionally high thermal diffusivity of spider silk,“ *Materials Letters*, vol. 114, pp. 1–3, 2014.

[79] A. Salazar, A. Mendioroz, and R. Fuente, „The strong influence of heat losses on the accurate measurement of thermal diffusivity using lock-in thermography,“ *Applied Physics Letters*, vol. 95, no. 12, p. 121 905, 2009.

[80] A. Salazar, A. Mendioroz, R. Fuente, and R. Celorrio, „Accurate measurements of the thermal diffusivity of thin filaments by lock-in thermography,“ *Journal of Applied Physics*, vol. 107, no. 4, p. 043 508, 2010.

[81] M. E. J. Hummel, C. Stelling, B. A. F. Kopera, *et al.*, „Ordered particle arrays via a langmuir transfer process: Access to any two-dimensional bravais lattice,“ *Langmuir*, vol. 35, no. 4, pp. 973–979, 2019.

[82] G. van Rossum, „Python tutorial,“ *Technical Report CS-R9526*, 1995.

[83] J. D. Hunter, „Matplotlib: A 2D graphics environment,“ *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[84] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[85] E. Jones, T. Oliphant, and P. Peterson, *SciPy: Open source scientific tools for Python*, 2001.

[86] E. Wilson and J. Gibbs, *Vector Analysis: A Text-book for the Use of Students of Mathematics & Physics: Founded Upon the Lectures of J. W. Gibbs*, ser. Yale bicentennial pub. Scribner, 1901.

[87] P. Q. Nguyen and D. Stehlé, „Low-dimensional lattice basis reduction revisited,“ *ACM Transactions on Algorithms*, vol. 5, no. 4, pp. 1–48, 2009.

[88] P. Q. Nguyen and B. Vallée, Eds., *The LLL Algorithm*. Springer Berlin Heidelberg, 2010.

[89] S. D. Galbraith, *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[90] R. Raccis, A. Nikoubashman, M. Retsch, *et al.*, „Confined diffusion in periodic porous nanostructures,“ *ACS Nano*, vol. 5, no. 6, pp. 4607–4616, 2011.

[91] COMSOL Inc., „COMSOL Multiphysics Reference Manual,“

[92] S. Plimpton, „Fast parallel algorithms for short-range molecular dynamics,“ *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, 1995.

[93] AutoDesk Inc. (Sep. 2019). „Autocad,“ [Online]. Available: https://www.autodesk.de/products/autocad/overview.

[94] W. J. Parker, R. J. Jenkins, C. P. Butler, and G. L. Abbott, „Flash method of determining thermal diffusivity, heat capacity, and thermal conductivity,“ *Journal of Applied Physics*, vol. 32, no. 9, pp. 1679–1684, 1961.

[95] M. Xu and L. V. Wang, „Photoacoustic imaging in biomedicine,“ *Review of Scientific Instruments*, vol. 77, no. 4, p. 041 101, 2006.

[96] V. Singh, T. L. Bougher, A. Weathers, *et al.*, „High thermal conductivity of chain-oriented amorphous polythiophene," *Nature Nanotechnology*, vol. 9, no. 5, pp. 384–390, 2014.

[97] Z. Wang, K. Rolle, T. Schilling, *et al.*, „Tunable thermoelastic anisotropy in hybrid bragg stacks with extreme polymer confinement," *Angewandte Chemie International Edition*, Dec. 2019.

[98] H. Hu, X. Wang, and X. Xu, „Generalized theory of the photoacoustic effect in a multilayer material," *Journal of Applied Physics*, vol. 86, no. 7, pp. 3953–3958, 1999.

[99] W. C. Michels and N. L. Curtis, „A pentode lock-in amplifier of high frequency selectivity," *Review of Scientific Instruments*, p. 444, 12 1941.

[100] A. J. Angström, „XVII. new method of determining the thermal conductibility of bodies," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 25, no. 166, pp. 130–142, 1863.

[101] X. P. Maldague, *Theory and Practice of Infrared Technology for Nondestructive Testing*. Wiley-Interscience, 2001.

[102] M. Clark, D. McCann, and M. Forde, „Application of infrared thermography to the non-destructive testing of concrete and masonry bridges," *NDT & E International*, vol. 36, no. 4, pp. 265–275, 2003.

[103] D. Bates, G. Smith, D. Lu, and J. Hewitt, „Rapid thermal non-destructive testing of aircraft components," *Composites Part B: Engineering*, vol. 31, no. 3, pp. 175–185, 2000.

[104] O. Breitenstein, W. Warta, and M. C. Schubert, *Lock-in Thermography: Basics and Use for Evaluating Electronic Devices and Materials*, ser. Springer Series in Advanced Microelectronics. Springer International Publishing, 2018, vol. 10.

[105] K. Huang, *Statistical Mechanics, 2nd Edition*. John Wiley & Sons, 1987.

[106] R. Gade and T. B. Moeslund, „Thermal cameras and applications: A survey," *Machine Vision and Applications*, vol. 25, no. 1, pp. 245–262, 2014.

[107] C. Meola and G. M. Carlomagno, „Recent advances in the use of infrared thermography," *Measurement Science and Technology*, vol. 15, no. 9, R27–R58, 2004.

[108] M. Vollmer and K.-P. Möllmann, *Infrared Thermal Imaging: Fundamentals, Research and Applications*. Wiley-VCH, 2018.

[109] L. Boltzmann, „Ableitung des Stefan'schen Gesetzes, betreffend die Abhängigkeit der Wärmestrahlung von der Temperatur aus der electromagnetischen Lichttheorie," *Annalen der Physik*, vol. 258, no. 6, pp. 291–294, 1884.

[110] P. J. Mohr, D. B. Newell, and B. N. Taylor, „CODATA recommended values of the fundamental physical constants: 2014,“ *Reviews of Modern Physics*, vol. 88, no. 3, 2016.

[111] M. Planck, „Zur Theorie des Gesetzes der Energieverteilung im Normalspectrum,“ *Verhandlungen der Deutschen Physikalischen Gesellschaft*, p. 237, 1900.

[112] A. J. Schmidt, X. Chen, and G. Chen, „Pulse accumulation, radial heat conduction, and anisotropic thermal conductivity in pump-probe transient thermoreflectance,“ *Review of Scientific Instruments*, vol. 79, no. 11, p. 114 902, 2008.

[113] A. J. Schmidt, R. Cheaito, and M. Chiesa, „A frequency-domain thermoreflectance method for the characterization of thermal properties,“ *Review of Scientific Instruments*, vol. 80, no. 9, p. 094 901, 2009.

[114] L. Fabbri and P. Fenici, „Three-dimensional photothermal radiometry for the determination of the thermal diffusivity of solids,“ *Review of Scientific Instruments*, vol. 66, no. 6, pp. 3593–3600, 1995.

[115] B. Zhang and R. E. Imhof, „Theoretical analysis of the surface thermal wave technique for measuring the thermal diffusivity of thin slabs,“ *Applied Physics A Materials Science & Processing*, vol. 62, no. 4, pp. 323–334, 1996.

[116] A. Philipp, N. W. Pech-May, B. A. F. Kopera, *et al.*, „Direct measurement of the in-plane thermal diffusivity of semitransparent thin films by lock-in thermography: An extension of the slopes method,“ *Analytical Chemistry*, vol. 91, no. 13, pp. 8476–8483, 2019.

[117] C. H. Kummell, „Reduction of observation equations which contain more than one observed quantity,“ *The Analyst*, vol. 6, no. 4, p. 97, 1879.

[118] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[119] C. Stelling, C. Bernhardt, and M. Retsch, „Subwavelength etched colloidal monolayers: A model system for tunable antireflective coatings,“ *Macromolecular Chemistry and Physics*, vol. 216, no. 16, pp. 1682–1688, 2015.

[120] M. Hummel, „Formation of ordered structures with symmetries other than hexagonally close-packed,“ 2019.

[121] J. T. Kindt, „Simulations of grain boundaries between ordered hard sphere monolayer domains: Orientation-dependent stiffness and its correlation with grain coarsening dynamics,“ *The Journal of Chemical Physics*, vol. 149, no. 4, p. 044 503, 2018.

[122] G. von Freymann, V. Kitaev, B. V. Lotsch, and G. A. Ozin, „Bottom-up assembly of photonic crystals,“ *Chemical Society Reviews*, vol. 42, no. 7, pp. 2528–2554, 2013.

[123] P. Ruckdeschel, A. Philipp, and M. Retsch, „Understanding thermal insulation in porous, particulate materials,“ *Advanced Functional Materials*, vol. 27, no. 38, p. 1 702 256, 2017.

[124] J. Hou, M. Li, and Y. Song, „Recent advances in colloidal photonic crystal sensors: Materials, structures and analysis methods,“ *Nano Today*, vol. 22, pp. 132–144, 2018.

[125] K. Younge, C. Christenson, A. Bohara, J. Crnkovic, and P. Saulnier, „A model system for examining the radial distribution function,“ *American Journal of Physics*, vol. 72, no. 9, pp. 1247–1250, 2004.

[126] A. Stukowski, „Visualization and analysis of atomistic simulation data with OVITO–the open visualization tool,“ *Modelling and Simulation in Materials Science and Engineering*, vol. 18, no. 1, p. 015 012, 2009.

[127] B. A. F. Kopera and M. Retsch, „Computing the 3D Radial Distribution Function from Particle Positions: An Advanced Analytic Approach,“ *Analytical Chemistry*, vol. 90, no. 23, pp. 13 909–13 914, 2018.

[128] J. K. Carson, S. J. Lovatt, D. J. Tanner, and A. C. Cleland, „Thermal conductivity bounds for isotropic, porous materials,“ *International Journal of Heat and Mass Transfer*, vol. 48, no. 11, pp. 2150–2158, 2005.

[129] J. Wang, J. K. Carson, M. F. North, and D. J. Cleland, „A new approach to modelling the effective thermal conductivity of heterogeneous materials,“ *International Journal of Heat and Mass Transfer*, vol. 49, no. 17-18, pp. 3075–3083, 2006.

[130] D. Stauffer and A. Aharony, *Introduction to percolation theory*. New York: John Wiley & Sons Canada, Limited, 1995.

[131] M. Rieß, J. Senker, R. Schobert, and J. Breu, „Microporous organically pillared layered silicates (MOPS): A versatile class of functional porous materials,“ *Chemistry - A European Journal*, vol. 25, no. 9, pp. 2103–2111, 2018.

[132] D. A. Kunz, J. Schmid, P. Feicht, *et al.*, „Clay-based nanocomposite coating for flexible optoelectronics applying commercial polymers,“ *ACS Nano*, vol. 7, no. 5, pp. 4275–4280, 2013.

[133] E. Doblhofer, J. Schmid, M. Rieß, *et al.*, „Structural insights into water-based spider silk protein–nanoclay composites with excellent gas and water vapor barrier properties,“ *ACS Applied Materials & Interfaces*, vol. 8, no. 38, pp. 25 535–25 543, 2016.

[134] E. S. Tsurko, P. Feicht, F. Nehm, *et al.*, „Large scale self-assembly of smectic nanocomposite films by doctor blading versus spray coating: Impact of crystal quality on barrier properties,“ *Macromolecules*, vol. 50, no. 11, pp. 4344–4350, 2017.

[135] M. Stöter, D. A. Kunz, M. Schmidt, *et al.*, „Nanoplatelets of sodium hectorite showing aspect ratios of 20000 and superior purity,“ *Langmuir*, vol. 29, no. 4, pp. 1280–1285, 2013.

[136] H. Hu, X. Wang, and X. Xu, „Generalized theory of the photoacoustic effect in a multilayer material,“ *Journal of Applied Physics*, vol. 86, no. 7, pp. 3953–3958, 1999.

[137] B. A. Cola, J. Xu, C. Cheng, *et al.*, „Photoacoustic characterization of carbon nanotube array thermal interfaces,“ *Journal of Applied Physics*, vol. 101, no. 5, p. 054 313, 2007.

[138] S. Cusack and A. Miller, „Determination of the elastic constants of collagen by brillouin light scattering,“ *Journal of Molecular Biology*, vol. 135, no. 1, pp. 39–51, 1979.

[139] Y.-F. Huang, Z.-G. Wang, H.-M. Yin, *et al.*, „Highly anisotropic, thermally conductive, and mechanically strong polymer composites with nacre-like structure for thermal management applications,“ *ACS Applied Nano Materials*, vol. 1, no. 7, pp. 3312–3320, 2018.

[140] M. Langner and A. Greiner, „Wet-laid meets electrospinning: Nonwovens for filtration applications from short electrospun polymer nanofiber dispersions,“ *Macromolecular Rapid Communications*, vol. 37, no. 4, pp. 351–355, 2016.

[141] M. Doimoto and A. Greiner, „Sustainable approach to superhydrophobic surfaces based on water-born electrospinning,“ *Macromolecular Materials and Engineering*, vol. 303, no. 9, p. 1 700 621, 2018.

[142] S. Jiang, Y. Chen, G. Duan, *et al.*, „Electrospun nanofiber reinforced composites: A review,“ *Polymer Chemistry*, vol. 9, no. 20, pp. 2685–2720, 2018.

[143] M. Retsch and U. Jonas, „Hierarchically structured, double-periodic inverse composite opals,“ *Advanced Functional Materials*, vol. 23, no. 43, pp. 5381–5389, 2013.

[144] M. M. Murr and D. E. Morse, „Fractal intermediates in the self-assembly of silicatein filaments,“ *Proceedings of the National Academy of Sciences*, vol. 102, no. 33, pp. 11 657–11 662, 2005.

[145] R. Diestel, *Graphentheorie*. Springer-Diestel, 2013.

[146] E. A. Jagla, „Phase behavior of a system of particles with core collapse,“ *Physical Review E*, vol. 58, no. 2, pp. 1478–1486, 1998.

10

Ordered Particle Arrays via a Langmuir Transfer Process: Access to Any Two-Dimensional Bravais Lattices

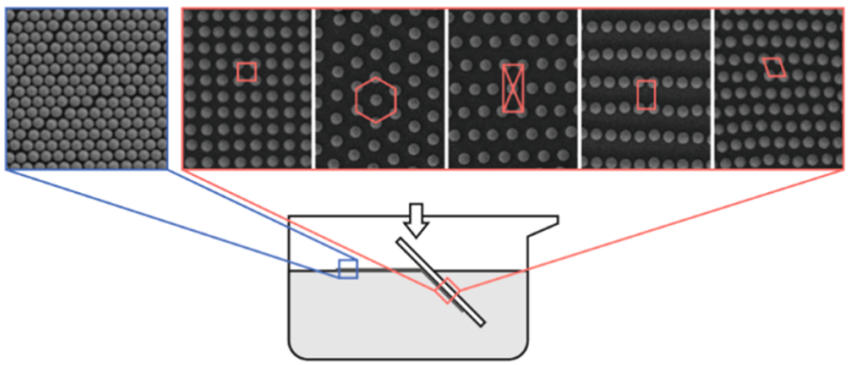
Miriam E. J. Hummel,[1,3] Christian Stelling,[1] Bernd A. F. Kopera,[1] Fabian Nutz,[1] Matthias Karg,[2] Markus Retsch[1] and Stephan Förster[1,3]

[1] Department of Chemistry, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

[2] Physical Chemistry I, Heinrich-Heine-University, 40204 Düsseldorf, Germany

[3] JCNS-1/ICS-1, Forschungszentrum Jülich, 52425 Jülich, Germany

[4] Physical Chemistry, RWTH University, 52074 Aachen, Germany



Published in *Langmuir*, **2019**, 35, 973-979
Reproduced with permission from ACS Publications

10.1 Abstract

We demonstrate how to directly transform a close-packed hexagonal colloidal monolayer into nonclosepacked particle arrays of any two-dimensional symmetry at the air/water interface. This major advancement in the field of nanoparticle self-assembly is based on a simple one-dimensional stretching step in combination with the particle array orientation. Our method goes far beyond existing strategies and allows access to all possible two-dimensional Bravais lattices. A key element of our work is the possibility to macroscopically stretch a particle array in a truly one-dimensional manner, which has not been possible up to now. We achieve this by stretching the nanoparticle array at an air/water interface during the transfer process. The degree of stretching is simply controlled by the wettability of the transfer substrate. To retain the symmetry of the transferred structure, the capillary forces upon drying have to be circumvented. We demonstrate two concepts based on thermal fixation for this. It allows for the first time to fabricate nonclose-packed, nonhexagonal colloidal monolayers on a macroscopic length scale.

10.2 Introduction

Solid substrates patterned with particle arrays are a key component for the fabrication of functional surfaces and thin film devices. They have applications in lithography,[1]–[3] optics,[4] photonics,[5] high-density data storage, as well as adhesive/nonadhesive surfaces.[6] Established preparation methods are based on block copolymers,[7],[8] direct assembly of colloidal particles on solid substrates,[9] or liquid interface-mediated assembly.[10] These techniques readily yield hexagonal closepacked (hcp) particle arrays. For many applications, it is highly desirable to use nonclose-packed particle arrays with tunable interparticle distances and nonhexagonal symmetries to control optical, adhesive, or magnetic surface properties, e.g. in the case of antireflective coatings[11] or photonic band gap devices.[12]

However, nonclose-packed particle arrays are not readily accessible. One fabrication route is based on the assembly of colloidal particles at the oil/water interface. Here, attractive capillary forces, which lead to the formation of close-packed arrays, are screened, and repulsive electrostatic interactions become dominant. This method allows increasing the interparticle distances up to several times the particle diameter.[13],[14] However, the colloidal monolayers lose their nonclose-packed character during transfer from the oil/water interface to a solid substrate due to the onset of attractive capillary forces.

A second fabrication method starts from hcp particle arrays on solid substrates. The monolayers are then transformed into nonclose-packed arrays by plasma etching, which reduces the diameter of the colloids without affecting their position.[10],[15] Here, the initial particle diameter predetermines the interparticle distance and the particle size cannot be reduced indefinitely due to limitations of the etching process.[16],[17] Further methods to prepare nonclose-packed particle monolayers comprise spin-coating,[18] substrate swelling,[19] shrinking of close-packed particles,[20],[21] or degradation[22],[23] of a polymer shell of core-shell particles. However, all these approaches merely produce hexagonal arrays.

Nonhexagonal ordered arrays are not accessible with standard self-assembly methods and usually require multiplestep procedures. One approach to nonclose-packed, nonhexagonal arrangements is based on the combination of isotropic swelling and anisotropic stretching of close-packed monolayers on elastomeric substrates. Depending on the individual stretching steps, the particles are separated along given directions yielding more complex arrays.[24] Besides, structured substrates have been used as templates to assemble colloidal particles in close-packed or nonclose-packed arrays with various lattice geometries.[25]–[27] The combination of hard and soft colloidal particles alongside with a restriction of the available surface area has also been demonstrated to result in nonhexagonal and nonclose-packed particle arrangements.[28] This experimental work is supported by modeling based on a hard core-soft shell interaction potential (Jagla potential).[29] Overall, there is currently no simple method available that could produce ordered nonclose-packed arrays of any symmetry in a controlled manner.

Here, we report a straightforward fabrication method for nonclose-packed particle arrays with any of the possible two-dimensional Bravais lattice symmetries. It starts from the readily accessible hcp monolayers, which are assembled at the water/air interface and collected on solid substrates. The transfer to the substrate induces a controlled, purely onedimensional stretching of the hcp-array leading to a variety of symmetries depending on the transfer angle and degree of stretching. This method can be applied to colloidal particles of various compositions and yields nanostructured areas of macroscopic size.

10.3 Experimental Section

Synthesis of Poly(*n*-butyl acrylate-*co*-methyl methacrylate) P(*n*BA-*co*-MMA) Particles

Monodisperse P(nBA-*co*-MMA) particles were prepared by emulsifier-free emulsion polymerization. 450 ml Milli-Q water was mixed with 35 ml MMA and 15 ml nBA, respectively. The mixture was equilibrated at 75 °C under a slight argon flow for 15 min. Subsequently, 2 ml of acrylic acid was added to the mixture followed by a further equilibration step for 5 min. The polymerization was started by a rapid injection of 150 mg potassium peroxodisulfate dissolved in 5 ml Milli-Q water. The reaction was carried out overnight. The polymerization was quenched by exposure to air. Residual educts were removed by dialysis against water for 5 days, changing water twice a day. Particles with a mean diameter of 434 ± 12 nm and with a glass transition temperature of 61 °C were obtained.

Preparation of Hydrophobic Substrates

Hydrophobic substrates were prepared by spin-coating glass slides with either polystyrene (PS), poly(n-butyl acrylate-*co*-styrene) (P(nBA-*co*-S)), or poly(methyl-methacrylate) and subsequent annealing at 200 °C for 5 min. SU-8 substrates were obtained by spin-coating SU-8 2050. After soft baking at 65 °C for 1 min and 95 °C for 7 min, the substrate was exposed to UV light for 2×4 s. A post-exposure bake was carried out at 65 °C for 1 min and 95 °C for 6 min. Poly(tetrafluoroethylene) (PTFE) films were used as received.

Preparation of Colloidal Crystals

The preparation of hcp monolayers is described in detail in ref [25]. Aqueous particle dispersions with a concentration of 2.5 wt% were spin-coated on cationically functionalized glass slides at a speed of 4000 rpm. Freely floating monolayers were assembled at the water/air interface by slow immersion of the particle-coated glass substrate into a 0.1 mM sodium dodecyl sulfate solution in Milli-Q water. The aqueous phase was adjusted to pH 12 by adding 30 % aqueous NH_3. A hydrophobic substrate was immersed through the floating monolayer at an angle of 45° relative to the water surface and was left at the bottom of the beaker with the monolayer facing upwards. After heating the water near the glass transition temperature of the particles for 5 min on a hotplate, the monolayers were removed from the solution and dried under ambient conditions.

Characterization

Scanning electron microscopy (SEM) images were obtained using a LEO 1530 Gemini Field Emission SEM (Carl Zeiss AG, Oberkochen, Germany) at 3.00 kV.

Particle positions and diameters were determined from the SEM images using MATLAB's circle detection function (Figure 10.11). From the particle positions, the average stretched basis vectors ($\vec{a}'$) and ($\vec{b}'$) were then determined by averaging all interparticle vectors with a Python script (Figure 10.11). Contact angle measurements were performed on an OCA 20 (DataPhysics Instruments GmbH, Filderstadt, Germany).

10.4 Results and Discussion

We prepared the particle monolayers at the water/air interface via an established method.[30] As model particles, we chose spherical polymer colloids (poly(*n*-butylacrylate-*co*-methylmethacrylate)) (P(*n*BA-*co*-MMA)) with a diameter of 434 ± 12 nm. The monolayers consist of large single-crystalline domains, resulting in intense iridescent colors. These monolayers were subsequently transferred onto solid substrates to fabricate the desired nonclose-packed and nonhexagonal particle arrays (Figure 10.1).

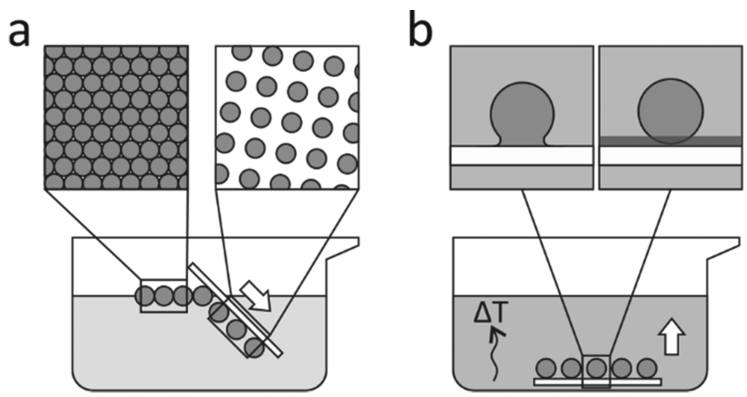


Fig. 10.1.: Fabrication of non-close packed monolayers. (a) A hcp monolayer from the water/air interface is transferred to the lower side of a hydrophobic substrate by submerging the substrate through the monolayer into the subphase. Thereby the particle monolayer is stretched into a nonclose-packed array. (b) The particles are immobilized on the substrate via thermal adhesion near the glass transition temperature (T_g) of the particles or the substrate. In the final step, the substrate with the monolayer can be recovered from the solution, whereas the structure is preserved.

We first tested the conventional transfer of a monolayer onto a hydrophilic substrate, which is withdrawn from the water (Z-type deposition [31]). As expected, this leads to the formation of ordered hcp particle arrays (Figure 10.5a).[30] The preparation of nonclose-packed arrays requires transferring the monolayer onto a hydrophobic substrate by immersion into water (X-type deposition). This is possible if the contact angle of the substrate is larger than a critical angle, in our case $\theta_c \sim 65°$. The monolayer is then deposited on the lower side of the substrate. We hypothesized, that during the transfer of the monolayer to the substrate, the monolayer would align and stretch into the transfer direction, such that nonclose-packed ordered arrays are accessible (Figure 10.5a). Unfortunately, the subsequent removal of the substrate from the aqueous solution resulted in the formation of a collapsed particle assembly (Figure 10.5b). This is a consequence of attractive capillary forces acting during the removal and drying. Nevertheless, a closer inspection of the SEM images revealed that the particles were not completely disordered, but showed a preferential alignment along the diagonals of the image. This indicated that the particles have indeed become aligned into nonclose-packed arrays, but then rearranged into close-packed structures and lost their order during removal of the substrate from the aqueous solution.

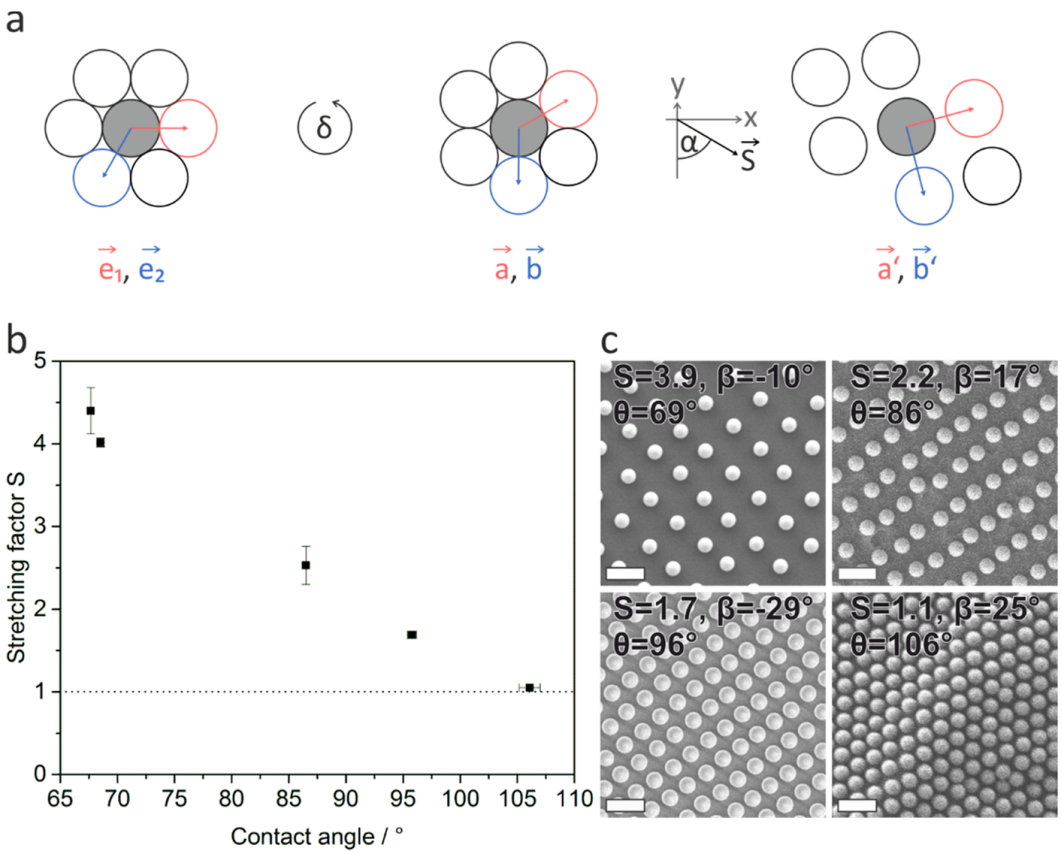


Fig. 10.2.: Stretched particle arrays. (a) Definition of the initial hcp domain orientation angle δ stretching angle α and stretching vector $\vec{S}$. (b) Dependence of the stretching factor S on the contact angle θ of the substrates. (c) Polymer particles on substrates with different contact angles: SU-8 ($\theta = 69°$), P(S-co-nBA) ($\theta = 86°$), PS ($\theta = 96°$), and PTFE film ($\theta = 106°$). The stretching factor S decreases with the increasing contact angle with at a constant stretching direction β. Scale bars are 1 µm.

To suppress the rearrangement, we immobilized the particle arrays prior to the removal from the aqueous solution by increasing their adhesive contact area on the substrate (Figure 10.6). For polymer particles with moderate glass transition temperatures (T_g), this can be conveniently done by thermal fixation near the T_g of the particles (Figure 10.5b). We also provide an alternative concept for arbitrary particles. These can be thermally immobilized on substrates coated with a thin polymer interlayer with adequately low glass transition temperature (Figure 10.7). One can imagine further fixation strategies such as chemical crosslinking or an exchange of the solvent.[13] After this fixation step, we were able to recover the substrate from the aqueous solution without affecting the particle positions (Figure 10.5c). An example for the successful immobilization of a nonclose-packed particle array with nonhexagonal order covering the entire substrate (1×1 cm^2) is shown in Figure 10.5. This demonstrates that the transfer onto hydrophobic substrates can generate large area, nonclosepacked particle arrays, and that a fixation step is crucial to maintain the structure.

The broad implication of this transfer method bases on the fact that the obtained type of particle array is completely defined by the transfer direction and the stretching factor and can be theoretically predicted as shown in Figure 10.2a. We derive this mathematically, starting from the base vectors $\vec{e_1}$ and $\vec{e_2}$ of the hexagonal unit cell of the hcp-layer which are given as

$$\vec{e_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \quad \vec{e_2} = \begin{pmatrix} -0.5 \\ -0.5\sqrt{3} \end{pmatrix} \tag{10.1}$$

The orientation of the hcp domain at the water/air interface before it is transferred to the substrate is specified by the rotation angle δ. The base vectors $\vec{a}$, $\vec{b}$ for the oriented hcp domain prior to transfer can be calculated using a rotation matrix

$$\vec{a} = \begin{bmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{bmatrix} \circ \vec{e_1} \tag{10.2}$$

$$\vec{b} = \begin{bmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{bmatrix} \circ \vec{e_2} \tag{10.3}$$

The stretching vector $\vec{S}$ with a stretching factor $S = |\vec{S}|$ determines the degree of stretching upon transfer of the hcp domain to the substrate. Additionally, the direction is defined by the angle α between the stretching vector $\vec{S}$ and the y-axis. (Figure 10.2a). Mathematically, the stretching process can be described by a sequence of rotation by an angle $-\alpha$ ($M_{-\alpha}$), followed by uniaxial stretching along the y-axis (M_S), and finally a back rotation (M_α) with the matrices

$$M_\alpha = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{10.4}$$

$$M_S = \begin{bmatrix} 1 & 0 \\ 0 & S \end{bmatrix} \tag{10.5}$$

This results in the following stretching matrix ($M = M_\alpha \circ M_S \circ M_{-\alpha}$) and the stretched vectors $\vec{a}$ and $\vec{b}$ the final particle array

$$M = \begin{bmatrix} \cos^2\alpha + \sin^2\alpha \cdot S & \sin\alpha \cdot \cos\alpha \cdot (1-S) \\ \sin\alpha \cdot \cos\alpha \cdot (1-S) & \cos^2\alpha + \sin^2\alpha \cdot S \end{bmatrix} \tag{10.6}$$

$$\vec{a}' = M \circ \vec{a} \tag{10.7}$$

$$\vec{b}' = M \circ \vec{b} \tag{10.8}$$

With this calculation, we show that the obtained two-dimensional arrays, characterized by the base vectors $\vec{a}'$ and $\vec{b}'$, are completely determined by the parameters δ,

α, and S. In the reversed case, where the stretching factor and direction need to be determined from the observed array, the equations to calculate δ, α, and S are given in the Supporting Information (eqs (10.10) and (10.11)). The two orientation angles δ and α are fully determined by the domain orientation and immersion direction and can be reduced to an angle $\beta = \alpha - \delta$, which describes the effective stretching direction with respect to the direction of the hcp array at the water/air interface. Because of its 6-fold rotational symmetry it is sufficient to consider an angular range $0° \leq |\beta| \leq 30°$. The same structures evolve for all angles $|\beta \pm n \cdot 60°|$, when keeping S constant, with n being an integer. We further demonstrate that also the stretching factor S can be well controlled experimentally. This is possible by adjusting the substrate contact angle Θ as shown in Figure 10.2b.[31] A wide range of contact angles was realized by spin-coating glass substrates with different polymers or directly using a poly(tetrafluoroethylene) (PTFE) film. We thus covered a range of contact angles from PTFE ($\Theta = 106°$), polystyrene (PS) ($\Theta = 96°$), poly(n-butyl acrylate-co-styrene) (P(nBA-co-S)) ($\Theta = 86°$), SU-8 photoresist ($\Theta = 69°$) to poly(methyl methacrylate) ($\Theta = 68°$). Using the different substrates leads to a large variation of the stretching parameter S at a constant stretching direction of $\Theta = 15°$, starting from the same floating hcp monolayer (Figure 10.2c). Substrates with a very high contact angle ($\Theta = 106°$) exhibit almost no stretching and the observed structures usually show a hexagonal close-packed arrangement. Lowering the contact angle leads to increasing stretching factors of the original monolayer. Contact angles near the critical angle ($\Theta = 69°$) result in remarkably large stretching factors, which promote the formation of particle lines and large tetragons. We want to stress two important properties of this interfacial stretching concept

(1) Since we work at the air/water interface, a truly one-dimensional stretching is possible. Lateral shear forces are negligible, which is in contrast to previous stretching strategies employing elastomeric substrates.[19],[32] This strongly increases the macroscopic homogeneity of the transferred structure.

(2) High stretching factors up to 400 % can be realized, simply by adjusting the hydrophobicity of the target substrate.

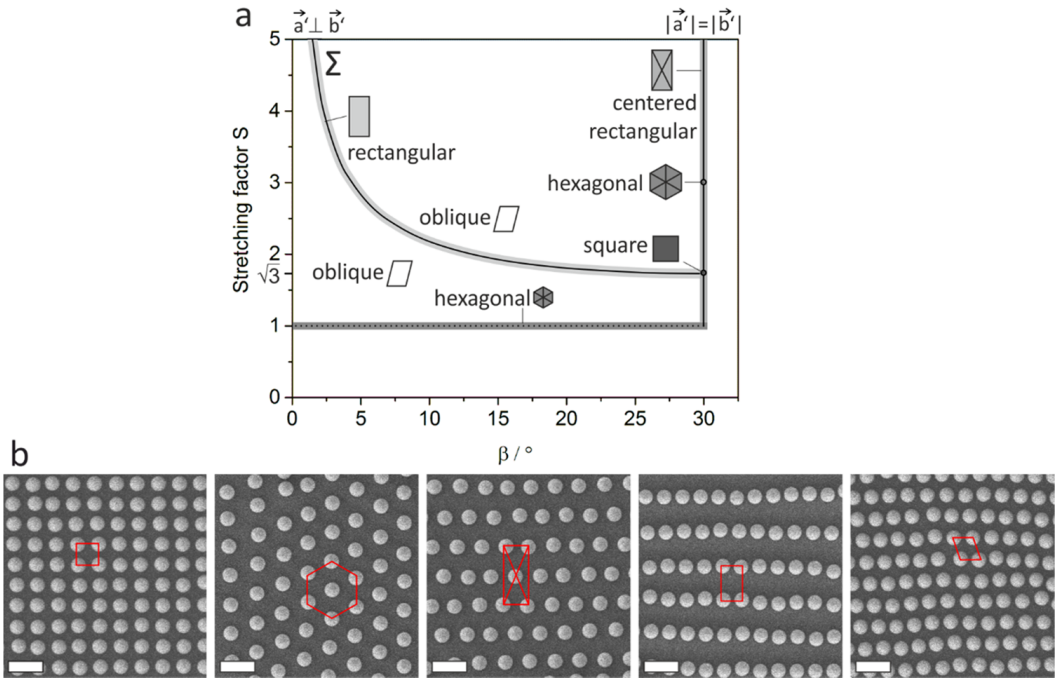


Fig. 10.3.: Particle array symmetries determined by β and S. (a) Phase diagram for Bravais lattice structures as a function of the stretching factor S and stretching angle β. (b) Particle monolayers are stretched to yield the five possible Bravais lattices in the two-dimensional space. (From left to right) Square, hexagonal, centered rectangular, rectangular, and oblique. Scale bars are 1 µm

The observed dependence of the stretching factor S on the substrate contact angle Θ can be well rationalized in terms of the flow pattern in the water sub phase during immersion of the substrate (Figure 10.9). This has been described in detail in the literature.[31] For contact angles smaller than $\theta_c \approx 65°$ the subphase shows a split-injection streamline such that the monolayer moves away from the contact line. Thus, X-type monolayer transfer is impossible. For slightly larger contact angles the monolayer starts to flow towards the contact line, but at a velocity smaller than the immersion velocity of the substrate. This leads to a large ratio between the immersed substrate area A_{sub} and transferred monolayer area A_{mon}, which determines the stretching factor $S = {}^{A_{sub}}/_{A_{mon}}$. Increasing the contact angle beyond θ_c increases the flow velocity of the monolayer towards the substrate, thus decreasing the stretching factor. At high contact angles, e.g., for PTFE, the monolayer transfer and substrate immersion velocity eventually become equal. This results in a transfer ratio close to unity, so that the stretching factors approach $S \approx 1$. With this, we show that the stretching factor can also be well controlled experimentally over a large range (between $S \approx 1$ and ≈ 4) via the substrate contact angle.

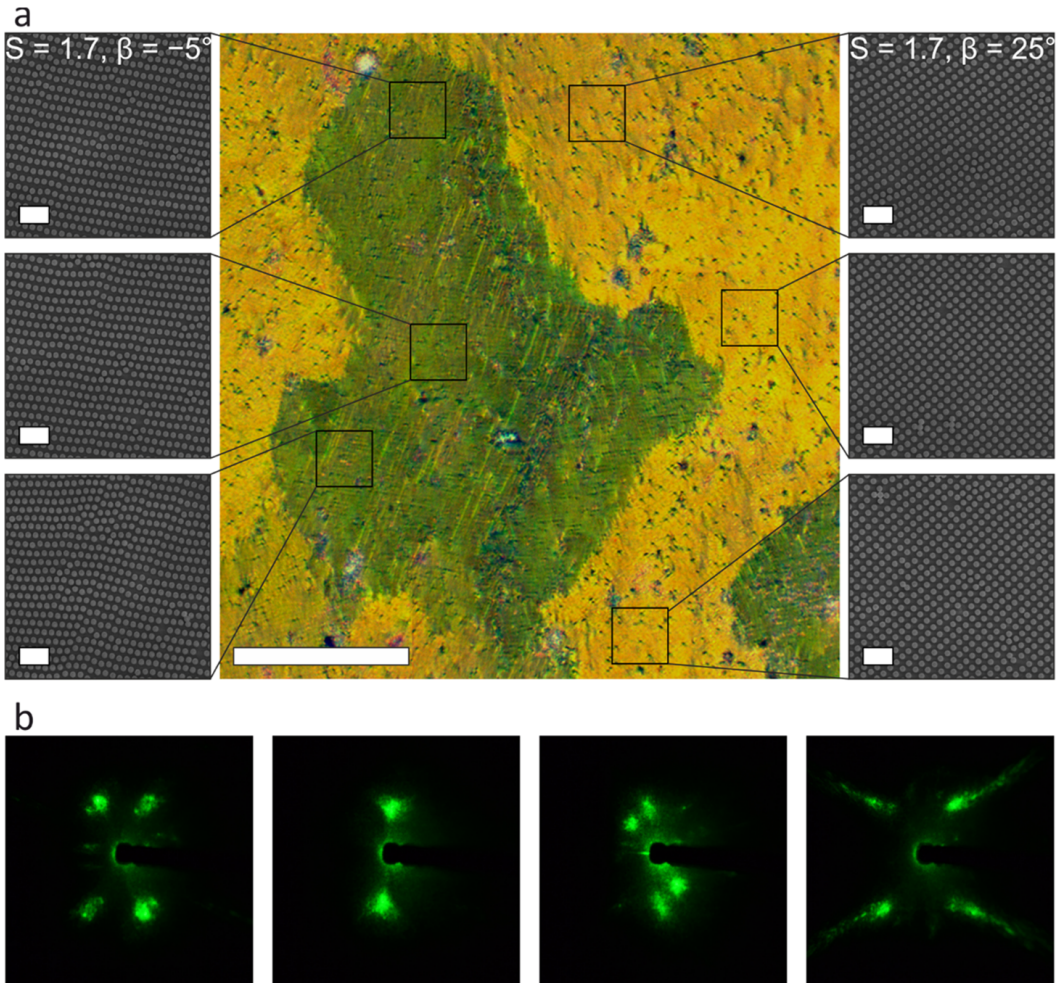


Fig. 10.4.: Uniform stretching of multicrystalline monolayers. (a) Optical microscopy image of large area, single-crystalline domains, and correlative electron microscopy on a sample with $S \approx 1.7$ (PS). Scale bars are 50 µm (optical microscopy) and 2 µm (electron microscopy). (b) Laser diffraction at various spots on one sample with $S \approx 1.7$. Four distinct array symmetries can be found when scanning across the substrate: (from left to right) rectangular, close-packed particle lines, oblique, and square.

Figure 10.3a demonstrates the full potential of this simple approach. All possible two-dimensional Bravais lattices symmetries are accessible by tuning both the stretching factor S and the effective stretching direction β. A square array of particles is obtained by stretching with a factor of $S = \sqrt{3} \approx 1.73$ and a stretching direction along one of the vectors defining the initial hexagonal unit cell ($\beta = 30°$). An ideal non close-packed hexagonal array can be realized by stretching with a factor of $S = 3$ along one of the initial vectors ($\beta = 30°$), whereas a centered rectangular array is the result for $S \neq 1.73$ and $S \neq 3$ at $\beta = 30°$.

For rectangular arrays, there exist various discrete combinations of stretching factors and directions that fulfill the following equation

$$S = \frac{1}{\sqrt{\frac{\sin\beta\cdot\left(\sqrt{3}-\tan\beta\right)}{\sqrt{3}\cdot\sin\beta+\cos\beta}}} \tag{10.9}$$

which is indicated by the branch $\sum$. For all other parameter combinations and $\beta \neq 30°$ and $\beta \neq 0°$ oblique lattices are obtained. Close-packed particle lines, corresponding to highly extended oblique lattices, can be fabricated by choosing $\beta = 0°$. Thus, nonclose-packed particle arrays for any of the five two-dimensional Bravais lattices (square, hexagon, rectangular, centered rectangular, and oblique) can be realized by a suitable combination of domain orientation and stretching factor.

Our findings have broad implications for the fabrication of ordered particle arrays, as the method should generally apply to all nano- and micro-scale particles that can be assembled at the water/air interface and transferred to a hydrophobic substrate. A crucial step is the immobilization of the particles on the substrate. For this purpose, we thermally fixated the monolayers to increase adhesive contact to the substrate, but also covalent or other noncovalent binding strategies are well conceivable. The fixation chemistry may thereby originate from the particle itself (e.g., its T_g) or the target substrate (e.g., spincoated interlayer with appropriate T_g). We note that the immersion velocity of the substrate ($1\,{}^{\mathrm{mm}}/_{\mathrm{min}}$ to $1\,{}^{\mathrm{cm}}/_{\mathrm{s}}$) has no significant influence on the resulting particle arrays. At these time scales the advancing contact angle can be assumed constant, which completely defines the magnitude of stretching.[31] Moreover, the immersion angle ($30° - 90°$) can also be neglected as only the local contact angle has to be considered.

In contrast to the stretching of monolayers on rubber substrates, which inherently show a contraction in the direction perpendicular to the strain direction, our method yields a uniform and purely one-dimensional stretching across the whole substrate. In Figure 10.4a, an optical microscopy image is shown of a monolayer transferred to a PS substrate. Two extended single-crystalline domains can be discerned from the distinct scattering colors corresponding to different domain orientations. We demonstrate that these distinct lattices are well-explained by the same degree of stretching; solely their initial orientation to the stretching direction differs. The absolute stretching value is constant across the whole image ($S \approx 1.7$) and identical structures are found within the whole singe-crystalline areas. A possible limitation for the preparation of large-area arrays could be the size limitation of the monocrystalline hcp domains. However, the area of monocrystalline domains formed by the method outlined above[30] is already sufficiently large for micro-optical applications.[33]

A series of diffraction measurements on our samples, using a focused laser beam shows distinct scattering patterns, which can be attributed to monocrystalline areas with specific symmetry (Figure 10.4b).

Finally, excellent techniques have been reported to reach uniform domain orientation of even larger size.[34],[35] The combination of such improved methods for monocrystalline colloidal monolayer formation with in situ techniques to assess, the monolayer lattice orientation will finally allow to fix the stretching direction β and consequently to deterministically fabricate a specific pre-defined Bravais lattice. Depending on the particle size (length scale) and composition (contrast), different methods can be envisioned for this task. Most straightforward will be laser diffraction experiments or direct optical microscopy at the air/water interface. More challenging will be grazing-angle X-ray scattering methods, which, however, will be able to address the sub-100 nm length scale as well.

10.5 Conclusion

In conclusion, we demonstrated a convenient and versatile method for the preparation of ordered nonclose-packed particle arrays with symmetries of any of the two-dimensional Bravais lattices. We are able to predict and experimentally realize a variety of ordered particle arrays by tuning only two parameters, the contact angle of the substrate and the stretching direction relative to the monolayer orientation. Compared to existing routes to nonclose-packed colloidal crystals, our approach is superior with respect to its simplicity, variability, and scalability. Up to now many of the fabricated array symmetries that we show have not been accessible by other self-assembly methods. The presented technique offers a new approach to enable the preparation of a large variety of defined ordered particle arrays to tune their optical, photonic, and wetting properties to specific applications. Furthermore, this method is compatible with batch and conti nuous soluti on surface processing methods with immediate implications for the generation of ordered particle arrays on a variety of solid and flexible substrates.

Associated Content

The Supporting Information is available free of charge on the ACS Publications website at DOI:10.1021/acs.lang-muir.8b03047. SEM images and photographs, calculations of stretching parameters and SEM image analysis are presented.

Author Information

Corresponding Authors

M.R. markus.retsch@uni-bayreuth.de

S.F. s.foerster@fz-juelich.de

ORCID

- Matthias Karg: 0000-0002-6247-3976
- Markus Retsch: 0000-0003-2629-8450
- Stephan Förster: 0000-0002-7323-2449

Author Contributions

The manuscript was written through contributions of all authors. All authors have given approval to the final version of the manuscript. M.E.J.H and C.S. contributed equally to this work.

Notes

The authors declare no competing financial interests.

Acknowledgments

This work was funded by the German Research Foundation (DFG) by the SFB840. M.E.J.H., C.S., and B.A.F.K. acknowledge support from the Elite Network of Bavaria (ENB). The authors thank Dr Beate Förster and Martina Heider from the Bavarian Polymer Institute (BPI) for their support in using the scanning electron microscopy facilities.

10.6 References

[1] A. Nemiroski, M. Gonidec, J. M. Fox, *et al.*, „Engineering shadows to fabricate optical metasurfaces," *ACS Nano*, vol. 8, no. 11, pp. 11 061–11 070, 2014.

[2] N. Vogel, C. K. Weiss, and K. Landfester, „From soft to hard: The generation of functional and complex colloidal monolayers for nanolithography," *Soft Matter*, vol. 8, pp. 4044–4061, 15 2012.

[3] S.-M. Yang, S. G. Jang, D.-G. Choi, S. Kim, and H. K. Yu, „Nanomachining by colloidal lithography," *Small*, vol. 2, no. 4, pp. 458–475, 2006.

[4] S. G. Romanov, N. Vogel, K. Bley, *et al.*, „Probing guided modes in a monolayer colloidal crystal on a flat metal film," *Physical Review B*, vol. 86, p. 195 145, 19 2012.

[5] B. Ai, Y. Yu, H. Möhwald, G. Zhang, and B. Yang, „Plasmonic films based on colloidal lithography," *Advances in Colloid and Interface Science*, vol. 206, pp. 5–16, 2014.

[6] D.-M. Drotlef, L. Stepien, M. Kappl, *et al.*, „Insights into the adhesive mechanisms of tree frogs using artificial mimics," *Advanced Functional Materials*, vol. 23, no. 9, pp. 1137–1146, 2013.

[7] M. Möller, J. P. Spatz, and A. Roescher, „Gold nanoparticles in micellar poly(styrene)-b-poly(ethylene oxide) films—size and interparticle distance control in monoparticulate films," *Advanced Materials*, vol. 8, no. 4, pp. 337–340, 1996.

[8] R. Glass, M. Müller, and J. P. Spatz, „Block copolymer micelle nanolithography," *Nanotechnology*, vol. 14, no. 10, pp. 1153–1160, 2003.

[9] N. D. Denkov, O. D. Velev, P. A. Kralchevsky, *et al.*, „Two-dimensional crystallization," *Nature*, vol. 361, no. 6407, pp. 26–26, 1993.

[10] N. Vogel, S. Utech, G. T. England, *et al.*, „Color from hierarchy: Diverse optical properties of micron-sized spherical colloidal assemblies," *Proceedings of the National Academy of Sciences*, vol. 112, no. 35, pp. 10 845–10 850, 2015.

[11] C. Stelling, C. Bernhardt, and M. Retsch, „Subwavelength etched colloidal monolayers: A model system for tunable antireflective coatings," *Macromolecular Chemistry and Physics*, vol. 216, no. 16, pp. 1682–1688, 2015.

[12] R. Fenollosa and F. Meseguer, „Non-close-packed artificial opals," *Advanced Materials*, vol. 15, no. 15, pp. 1282–1285, 2003.

[13] L. Isa, K. Kumar, M. Müller, *et al.*, „Particle lithography from colloidal self-assembly at liquid-liquid interfaces," *ACS Nano*, vol. 4, no. 10, pp. 5665–5670, 2010.

[14] A. D. Law, D. M. A. Buzza, and T. S. Horozov, „Two-dimensional colloidal alloys,“ *Physical Review Letters*, vol. 106, p. 128 302, 12 2011.

[15] N. Vogel, S. Goerres, K. Landfester, and C. K. Weiss, „A convenient method to produce close- and non-close-packed monolayers using direct assembly at the air-water interface and subsequent plasma-induced size reduction,“ *Macromolecular Chemistry and Physics*, vol. 212, no. 16, pp. 1719–1734, 2011.

[16] E. M. Akinoglu, A. J. Morfa, and M. Giersig, „Understanding anisotropic plasma etching of two-dimensional polystyrene opals for advanced materials fabrication,“ *Langmuir*, vol. 30, no. 41, pp. 12 354–12 361, 2014.

[17] A. Plettl, F. Enderle, M. Saitner, *et al.*, „Non-close-packed crystals from self-assembled polystyrene spheres by isotropic plasma etching: Adding flexibility to colloid lithography,“ *Advanced Functional Materials*, vol. 19, no. 20, pp. 3279–3284, 2009.

[18] P. Jiang, T. Prasad, M. J. McFarland, and V. L. Colvin, „Two-dimensional nonclose-packed colloidal crystals formed by spincoating,“ *Applied Physics Letters*, vol. 89, no. 1, p. 011 908, 2006.

[19] X. Yan, J. Yao, G. Lu, *et al.*, „Fabrication of non-close-packed arrays of colloidal spheres by soft lithography,“ *Journal of the American Chemical Society*, vol. 127, no. 21, pp. 7688–7689, 2005.

[20] K. Volk, J. P. S. Fitzgerald, M. Retsch, and M. Karg, „Time-controlled colloidal superstructures: Long-range plasmon resonance coupling in particle monolayers,“ *Advanced Materials*, vol. 27, no. 45, pp. 7332–7337, 2015.

[21] S. B. Quint and C. Pacholski, „Extraordinary long range order in self-healing non-close packed 2D arrays,“ *Soft Matter*, vol. 7, no. 8, p. 3735, 2011.

[22] N. Vogel, C. Fernández-López, J. Pérez-Juste, *et al.*, „Ordered arrays of gold nanostructures from interfacially assembled au@PNIPAM hybrid nanoparticles,“ *Langmuir*, vol. 28, no. 24, pp. 8985–8993, 2012.

[23] J. P. Spatz, S. Mössmer, C. Hartmann, *et al.*, „Ordered deposition of inorganic clusters from micellar block copolymer films,“ *Langmuir*, vol. 16, no. 2, pp. 407–415, 2000.

[24] X. Li, T. Wang, J. Zhang, *et al.*, „Modulating two-dimensional non-close-packed colloidal crystal arrays by deformable soft lithography,“ *Langmuir*, vol. 26, no. 4, pp. 2930–2936, 2010.

[25] L. Malaquin, T. Kraus, H. Schmid, E. Delamarche, and H. Wolf, „Controlled particle placement through convective and capillary assembly,“ *Langmuir*, vol. 23, no. 23, pp. 11 513–11 521, 2007.

[26] N. N. Khanh and K. B. Yoon, „Facile organization of colloidal particles into large, perfect one- and two-dimensional arrays by dry manual assembly on patterned substrates,“ *Journal of the American Chemical Society*, vol. 131, no. 40, pp. 14 228–14 230, 2009.

[27] C. Stelling, A. Mark, G. Papastavrou, and M. Retsch, „Showing particles their place: Deterministic colloid immobilization by gold nanomeshes,“ *Nanoscale*, vol. 8, no. 30, pp. 14 556–14 564, 2016.

[28] M. Rey, A. D. Law, D. M. A. Buzza, and N. Vogel, „Anisotropic self-assembly from isotropic colloidal building blocks,“ *Journal of the American Chemical Society*, vol. 139, no. 48, pp. 17 464–17 473, 2017.

[29] E. A. Jagla, „Phase behavior of a system of particles with core collapse,“ *Physical Review E*, vol. 58, no. 2, pp. 1478–1486, 1998.

[30] M. Retsch, Z. Zhou, S. Rivera, *et al.*, „Fabrication of large-area, transferable colloidal monolayers utilizing self-assembly at the air/water interface,“ *Macromolecular Chemistry and Physics*, vol. 210, no. 3-4, pp. 230–241, 2009.

[31] R. L. Cerro, „Moving contact lines and langmuir–blodgett film deposition,“ *Journal of Colloid and Interface Science*, vol. 257, no. 2, pp. 276–283, 2003.

[32] H. K. Choi, S. H. Im, and O. O. Park, „Shape and feature size control of colloidal crystal-based patterns using stretched polydimethylsiloxane replica molds,“ *Langmuir*, vol. 25, no. 20, pp. 12 011–12 014, 2009.

[33] A. Vega-Flick, R. A. Duncan, S. P. Wallen, *et al.*, „Vibrational dynamics of a two-dimensional microgranular crystal,“ *Physical Review B*, vol. 96, p. 024 303, 2 2017.

[34] K. Shinotsuka, Y. Kajita, K. Hongo, and Y. Hatta, „Crystal perfection of particle monolayer at the air–water interface,“ *Langmuir*, vol. 31, no. 42, pp. 11 452–11 457, 2015.

[35] X. Meng and D. Qiu, „Gas-flow-induced reorientation to centimeter-sized two-dimensional colloidal single crystal of polystyrene particle,“ *Langmuir*, vol. 30, no. 11, pp. 3019–3023, 2014.

10.7 Supporting Information

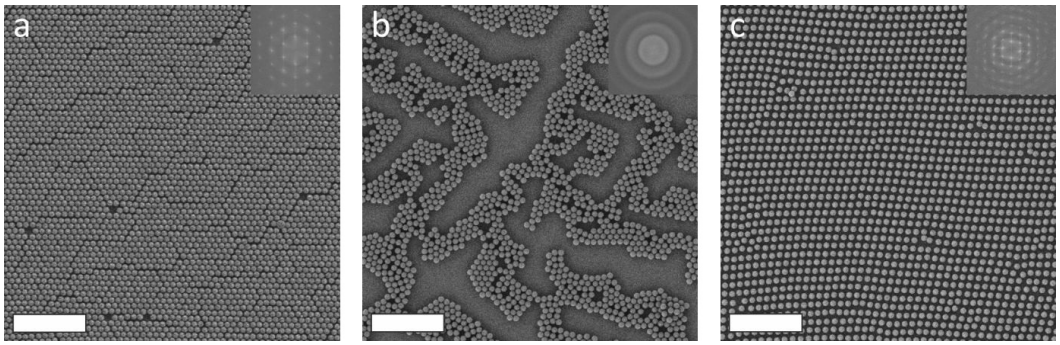


Fig. 10.5.: SEM images of particle monolayers on hydrophilic and hydrophobic substrates. The insets show the corresponding fast Fourier transform (FFT). (a) Particles on a hydrophilic glass substrate (hcp structure). (b) Particles on a hydrophobic substrate without thermal fixation (collapsed line structure). (c) Particles on a hydrophobic substrate with thermal fixation (non-close-packed oblique structure). Scale bars are 5 µm.

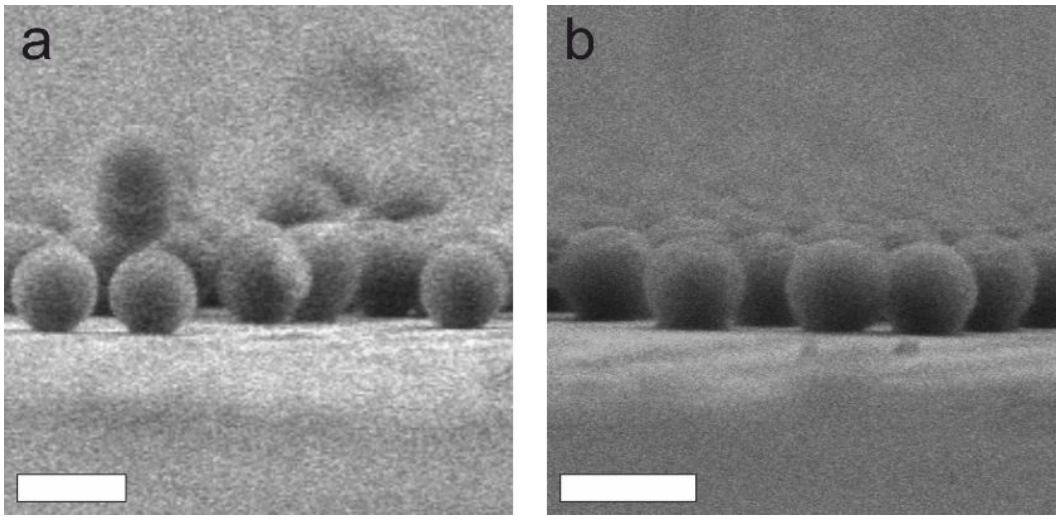


Fig. 10.6.: Side view SEM images of (a) non-immobilized and (b) immobilized particles. Without thermal annealing, the particles can be viewed as hard spheres which feature a minimum contact area on the substrate. Upon drying of the monolayer, these particles are subject to capillary forces that lead to a shift to new particle positions. Thermal annealing of the particles above their glass transition temperature T_g leads to an increase of the contact area on the substrate. In the viscous state, the particles fuse onto the substrate and become immobilized at their positions even under capillary stress. Scale bars are 500 nm.

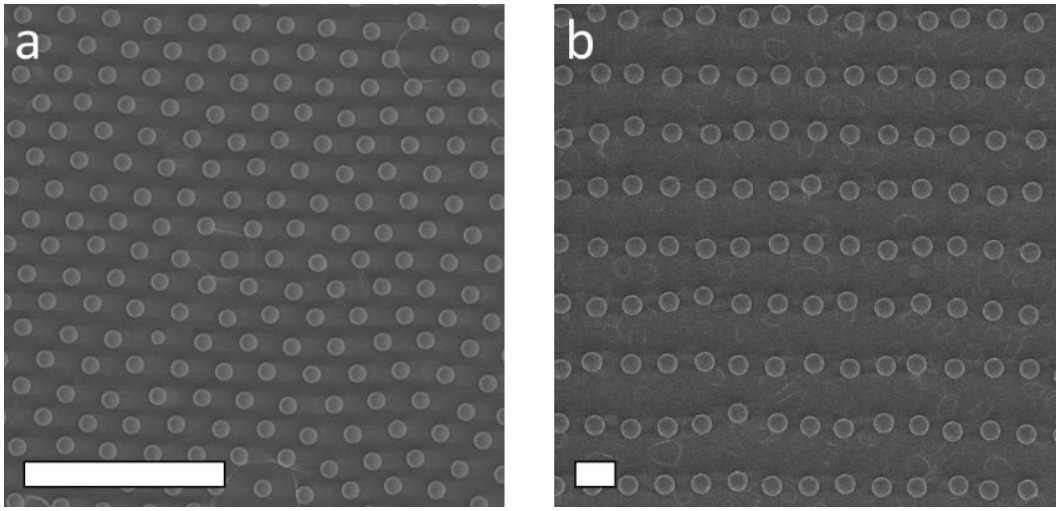


Fig. 10.7.: SEM images of particle monolayers immobilized on hydrophobic substrates with moderate glass transition temperature. PS Particles ($T_g \approx 105\,°C$) with a diameter of (a) 453 nm and (b) 2560 nm thermally fixated on spin-coated P(nBA-*co*-MMA) ($T_g \approx 55\,°C$) layer. Scale bars are 5 µm.

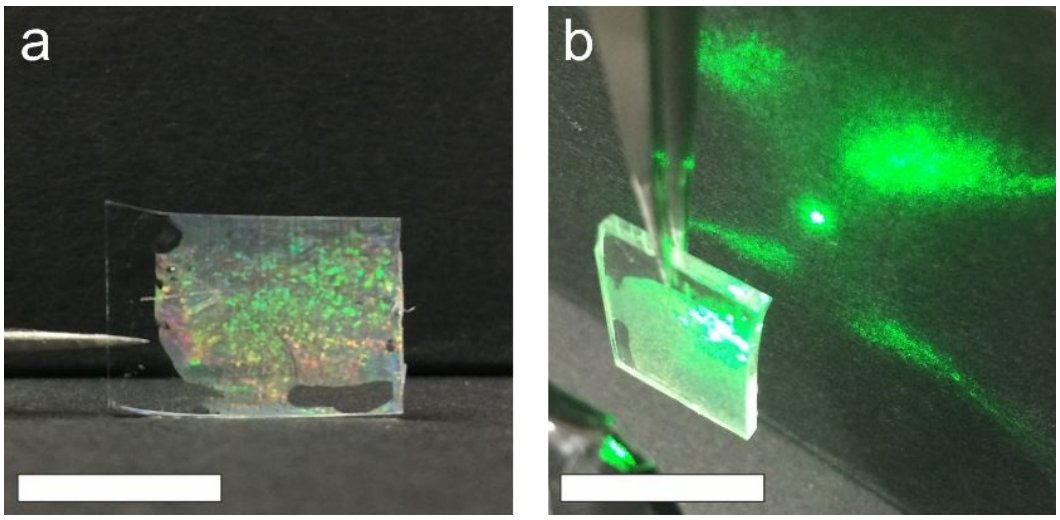


Fig. 10.8.: Immobilized particle monolayer. (a) Photograph of an immobilized particle monolayer on a hydrophobic substrate. The iridescent color preserved after drying of the monolayer indicate an intact positional arrangement of the particles. (b) Laser diffraction of an immobilized particle monolayer. The fourfold symmetric peaks confirm the presence of an extended particle domain with square symmetry. Scale bars are 1 cm.

Calculation of δ, α and S

Experimentally found structures can be assigned to specific stretching parameters δ, α and S. For this the stretched vectors $\vec{a}'$ and $\vec{b}'$ with the coordinates x'_a, y'_a and x'_b, y'_b have to be extracted from SEM images. The coordinates are inserted into

the following two equations to calculate α and S. In order to determine the right parameters, the results of both $\vec{a}'$ and $\vec{b}'$ have to be screened with respect to δ.

$$\alpha = \operatorname{arccot}\left(\frac{y' - y(\delta)}{x' - x(\delta)}\right) \tag{10.10}$$

$$S = \frac{x' - x(\delta) \cdot \cos^2\alpha + y(\delta) \cdot \sin\alpha \cdot \cos\alpha}{x(\delta) \cdot \sin^2\alpha + y(\delta) \cdot \sin\alpha \cdot \cos\alpha} \tag{10.11}$$

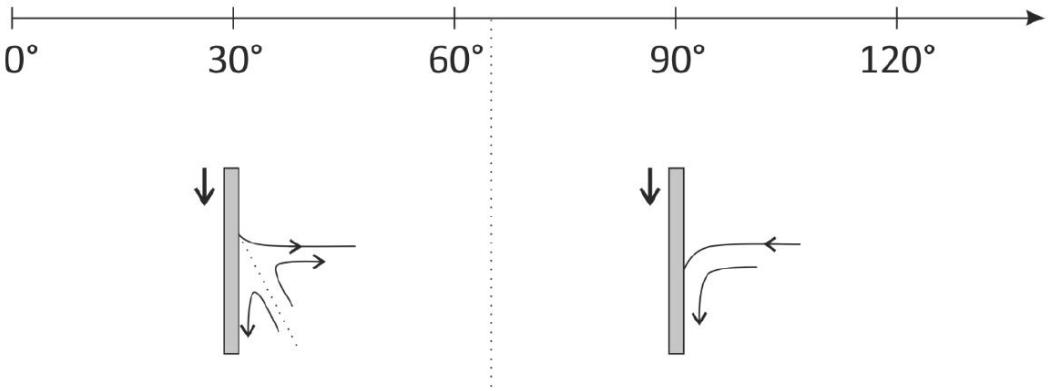


Fig. 10.9.: Streamline profiles upon immersion of a solid substrate at different contact angles.[146] The dotted line indicates the critical angle θ_c below which an X-type is impossible.

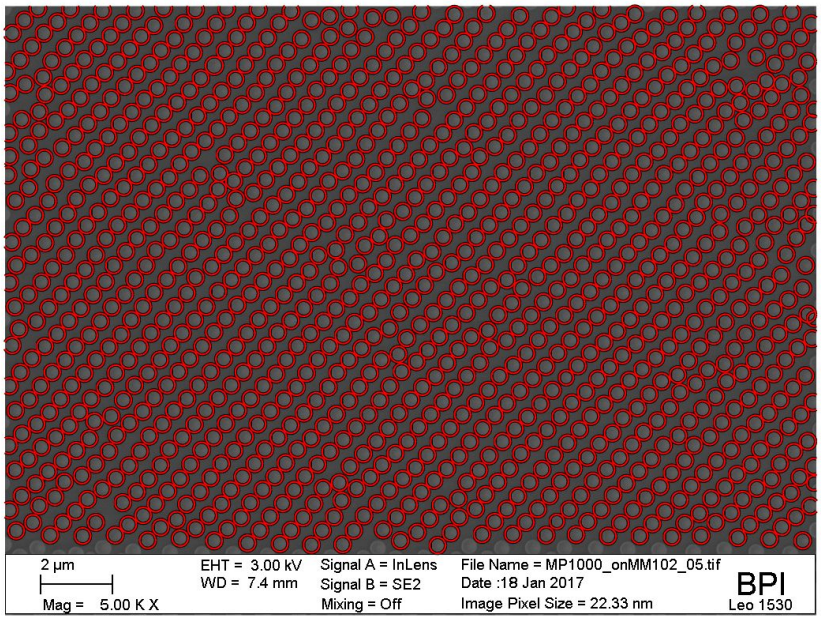


Fig. 10.10.: SEM image of a stretched monodomain. The red circles indicate the particles found by MATLAB's circle detection function after optimizing the search parameters manually.

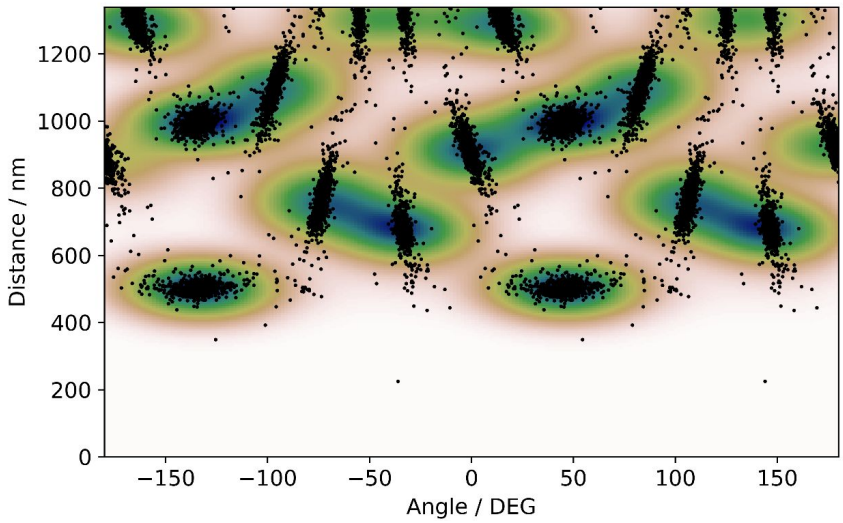


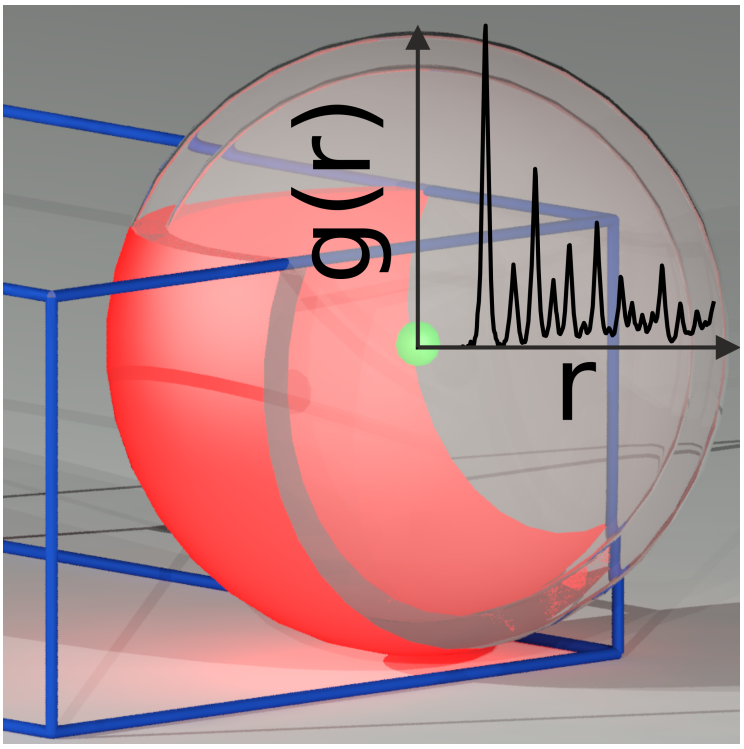
Fig. 10.11.: 2D histogram of the distance between particle pairs and the corresponding angle with the x-axis for the SEM image in Figure 10.10. Each black dot represents an inter-particle connection. A Gaussian kernel density estimator is used to calculate the smooth distribution function (color). Local maxima in the distribution, with small interparticle distances, represent possible candidates for the stretched basis vectors $\vec{a}'$ and $\vec{b}'$ in polar coordinates.

11

Computing the 3D Radial Distribution Function from Particle Positions: An Advanced Analytic Approach

Bernd A. F. Kopera and Markus Retsch

Department of Chemistry, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany



Published in *Analytical Chemistry*, **2018**, 90, 23, 13909-13914
Reproduced with permission from ACS Publications

11.1 Abstract

The radial distribution function, $g(r)$, is ubiquitously used to analyze the internal structure of particulate systems. However, experimentally derived particle coordinates are always confined to a finite sample volume. This poses a particular challenge on computing $g(r)$: Once the radial distance, r, extends beyond the sample boundaries in at least one dimension, substantial deviations from the true $g(r)$ function can occur. State of the art algorithms for $g(r)$ mitigate this issue for instance by using artificial periodic boundary conditions. However, ignoring the finite nature of the sample volume distorts $g(r)$ significantly. Here, we present a simple, analytic algorithm for the computation of $g(r)$ in finite samples. No additional assumptions about the sample are required. The key idea is to use an analytic solution for the intersection volume between a spherical shell and the sample volume. In addition, we discovered a natural upper bound for the radial distance that only depends on sample size and shape. This analytic approach will prove to be invaluable for the quantitative analysis of the increasing amount of experimentally derived tomography data.

11.2 Introduction

The radial distribution function, $g(r)$, is an important tool to analyze the internal structure of particulate systems. Examples include the analysis of colloid assemblies,[1] particle dynamics,[2]–[5] plasma physics,[6] sphere packings,[7],[8] and protein diffusion.[9] Real space particle positions become increasingly available with the advent of TEM tomography,[10],[11] positron emission tomography,[12] and microcomputed X-ray tomography.[13] The crystal structure, number of next neighbors, and interaction potentials between particles are accessible from $g(r)$.[14],[15] Thus, $g(r)$ is a very powerful tool to analyze complex 3D particle positions in a quantitative way. However, computing $g(r)$ is challenging for finite samples once the radial distance, r, exceeds the sample in at least one dimension. No particle positions are known outside the measurement boundaries. This has to be taken into account once the radial distance reaches beyond the sample volume. Figure 11.1 shows the effect of ignoring the finite sample volume.

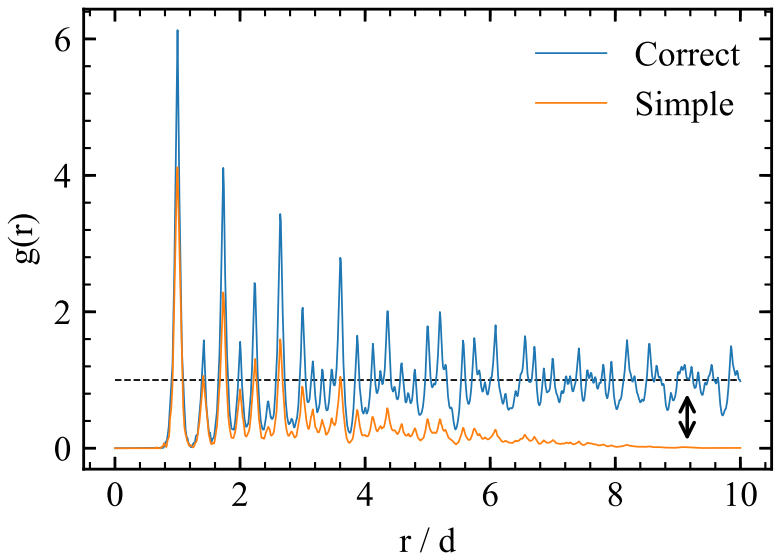


Fig. 11.1.: Radial distribution functions for a Gaussian disturbed FCC lattice of 665 particles. Note that $g(r)$ computed with the simple algorithm artificially decays to 0 at large radii. The correct $g(r)$ approaches 1. The artificial decay is caused by ignoring the boundaries of the finite sample volume.

The simple algorithm, which does not take the sample volume into account, has lower peak heights and artificially decays to 0 at large radii. We tested the established software packages OVITO[16] and MDAnalysis.[17] Both use this simple algorithm and normalize $g(r)$ incorrectly without periodic boundaries. This problem is not new: Current algorithms for the computation of $g(r)$ fall into three categories:

1. One can extend the finite sample to infinity by applying periodic boundaries.[15] In this case, the radial distance never exceeds the now infinitely large sample volume. However, this is an additional, unnecessary assumption about the sample. Periodic boundaries on all samples introduce artificial periodicities (see Supporting Information (SI) Section 11.8.2) and should therefore be avoided.

2. Another way to compute $g(r)$ is to average over a smaller subset of the sample and at the same time limit the radial distance. In this case, the correct $g(r)$ is obtained. However, this procedure creates an unnecessary trade-off between noise reduction and the maximal radial distance. A significant amount of information is thereby lost (see SI Section 11.14).

3. The third way is to estimate $g(r)$ with a Monte Carlo algorithm.[18] This method is robust and simple. However, all Monte Carlo algorithms feature an intrinsic trade-off between computational runtime and numeric precision.

Thus, there is currently no efficient method available to compute $g(r)$ from finite samples, i.e., from experimentally derived particle coordinates such as (high resolution) fluorescence microscopy or electron or X-ray tomography. Here, we present an efficient, analytic, algorithm for the correct computation of $g(r)$. No additional assumptions about the sample are required. Our key idea is to use an analytic solution for the intersection volume between the sample and a spherical shell. Using this intersection volume, we normalize $g(r)$ in finite samples correctly. At the same time, our algorithm takes all particles within the measured volume into account. Consequently, this results in good long-range correlations at a low noise. The maximal radial distance is only limited by the total sample dimensions.

11.3 Theory

Radial Distribution Function

We now present a brief description of the standard algorithm for computing $g(r)$. A good explanation of $g(r)$ is presented by Younge *et al.*[18] The radial distribution function is the normalized average radial particle number density histogram. Normalized means that the radial particle density is divided by the average particle density in the sample. The average particle density is given by the number of particles in the sample, $\#\mathrm{P}_{\mathrm{Sample}}$, divided by the sample volume, $\mathrm{V}_{\mathrm{Sample}}$. The local radial particle density is computed as follows. First, we create concentric spherical shells, with thickness, Δr, around a central particle. The volume of such a shell, $V_{\mathrm{Shell}}(r, \Delta r)$, is given by

$$V_{\mathrm{Shell}}(r, \delta r) = \frac{4}{3}\pi \left[\left(r + \frac{\delta r}{2} \right)^3 - \left(r - \frac{\delta r}{2} \right)^3 \right] \tag{11.1}$$

Each of these shells corresponds to a radial bin in the $g(r)$ histogram. Therefore, we get the local particle number density by dividing the particle number in each shell, $\#P(r, \Delta r)$, by the corresponding shell volume. Note that this procedure assumes that the entire shell is filled with particles. The local $g(r)$ around the central particle is then given by

$$g_{\mathrm{Local}}(r, \delta r) = \frac{\#P(r, \delta r)}{V_{\mathrm{Shell}}(r, \delta r)} \div \frac{\#P_{\mathrm{Sample}}}{V_{\mathrm{Sample}}} \tag{11.2}$$

We then compute the local $g(r)$ using every particle in our sample as the central particle. Averaging all local $g(r)$ reduces noise and yields the final $g(r)$. We call this procedure the simple algorithm when the finite sample volume is not taken into account. Further details for computing $g(r)$ are given in SI Section 11.8.1. The key problem with this procedure is that the shell volume is not calculated correctly at the edges of a finite set of particle coordinates. Consequently, $g(r)$ converges to 0 instead of 1 in the simple case for large distances.

Possible Shell-Sample Intersection Volumes

The key idea of this paper is to use the shell-sample intersection volume for normalizing $g(r)$. Three likely sample volumes are shown in Figure 11.2. The intersection volume between a sphere and a shell has been published by Freireich *et al.*[19] Spherical sample boundaries occur in colloidal superballs[20] and spherical nanoparticles.

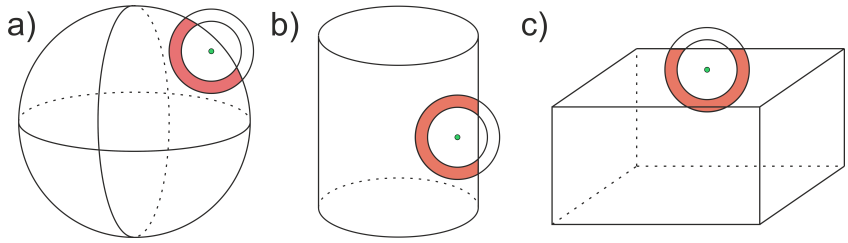


Fig. 11.2.: Possible sample boundary shapes: (a) sphere, (b) cylinder, (c) rectangular cuboid (box). Only the central particle is shown for clarity. The intersection volume between a spherical shell and the sample volume is shown in red. This intersection volume is essential for computing the correct $g(r)$.

Cylinders are the natural sample shape for confocal scanning microscopy and transmission electron microscopy. A lens creates a focal plane bounded by a circle. Scanning through different heights creates a cylindrically shaped volume. Unfortunately, the intersection volume between a sphere and a finite cylinder has no analytic solution. A good numeric approximation is presented by Mueller.[21] In practice, most sample boundaries are rectangular cuboids (boxes). They are created by rectangular detectors and scanning in a perpendicular manner through the sample. Therefore, we will focus on rectangular sample volumes in the following. The general case of the intersection between a sphere and an arbitrary polyhedron was recently solved by Strobl *et al.*[22] We adapt their approach to derive the intersection volume between a spherical shell and a box and apply it to the $g(r)$ normalization problem.

Shell-Box Intersection Volume

We now derive the intersection volume between a spherical shell and a rectangular box, $V_{\text{Shell-Box}}$. Every spherical shell is bounded by two concentric spheres. Both spheres have their center at the central particle. Figure 11.3 shows a cross section through a shell-box intersection with important intersection volumes highlighted. We set the central particle as the origin of our Cartesian coordinate system (see Figure 11.4). The axes of our coordinate system align with the edges of the rectangular box.

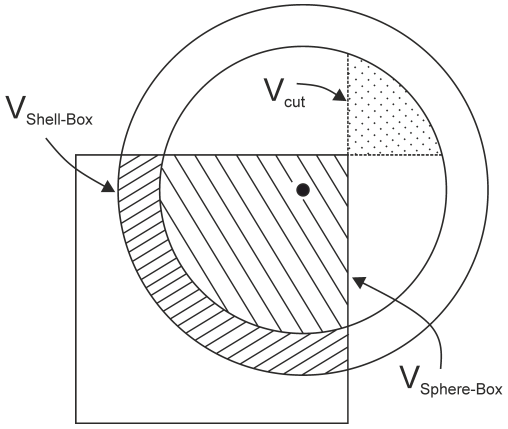


Fig. 11.3.: Cross section through a box intersected by a spherical shell. Three important intersection volumes are shown.

We compute the volume of a spherical shell by subtracting the inner boundary sphere from the outer boundary sphere like

$$V_{\text{Shell-Box}}(r, \delta r) = V_{\text{Sphere-Box}}(r + \frac{\delta r}{2}) - V_{\text{Shell-Box}}(r - \frac{\delta r}{2}). \tag{11.3}$$

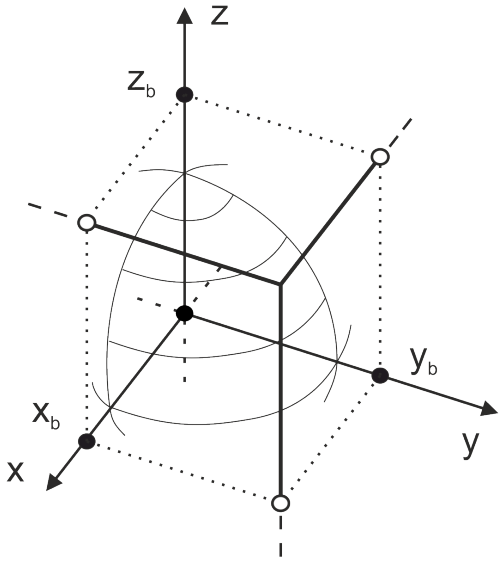


Fig. 11.4.: Cartesian coordinate system with the central particle at the origin. The faces of the rectangular box are defined by their intersection with the axes at x_b, y_b, and z_b. Only the corner in the all positive octant is shown.

Therefore, we have to find the intersection volume between a sphere of radius R and a box. This simplification is valid, because the intersection of the inner sphere and the box is always completely inside the intersection volume of the outer sphere and the box. To further simplify the intersection volume, we split each sphere into eight octants along the coordinate planes. Each sphere octant is intersected by at

most three of the six box faces. These three faces are adjacent to a single corner. This is true, because the center of the sphere and our coordinate system is always inside the box. Furthermore, we mirror all octants into the all positive octant (I). Mirroring preserves the volume. This allows us to treat all octants the same and in turn significantly reduce the number of possible cases. Figure 11.4 shows the corner of the box in the all positive (first) octant of our coordinate system. The boundary positions are defined by the intersections of the box faces with the coordinate axes, named x_b, y_b, and z_b. Overall, the sphere-box intersection volume, $V_\text{Sphere-Box}$, is the sum of all the octant-corner intersection volumes

$$V_\text{Sphere-Box} = \sum_{i=1}^{8} V_{\text{Octant},i\text{ - Corner},i} \tag{11.4}$$

There are a few cases to consider. First, the corner of the box can be fully inside the sphere octant (see Figure 11.20). In this case, the remaining intersection volume is only bounded by the box faces and is given by $(x_\mathrm{b} \cdot y_\mathrm{b} \cdot z_\mathrm{b})$. Next, we look at a single box face intersecting the sphere octant (see Figure 11.21). Here, a quarter of a spherical cap is removed from the sphere octant. This is true for every face, as long as there is no overlap between the spherical caps. Therefore, we test every face of the box corner for overlap with the sphere octant. If they overlap, we remove the respective quarter of a spherical cap. Finally, we have to test for an overlap of the spherical caps. We test this case by checking

$$a^2 + b^2 < R^2 \tag{11.5}$$

for all three possible box face pairs (a, b) $\forall\ a, b\ \in\ (x_\mathrm{b}, y_\mathrm{b}, z_\mathrm{b})$ $a \neq b$ If the spherical caps overlap, we add the overlap volume back to the remaining octant. This is necessary, because we removed the overlap volume twice in the previous step (see Figure 11.24). We call the overlap volume between two spherical caps a spherical cut. We derived the volume of a spherical cut in SI Section 4 as

$$\begin{aligned} V_\text{cut}(R_\mathrm{s}, a, b) = {} & \frac{R_\mathrm{s}^3}{6} \cdot \left[\pi - 2 \cdot \arctan\left(\frac{a \cdot b}{R_\mathrm{s} \cdot \sqrt{R_\mathrm{s}^2 - a^2 - b^2}}\right)\right] \\ & + \frac{1}{2} \cdot \left[\arctan\left(\frac{a}{\sqrt{R_\mathrm{s}^2 - a^2 - b^2}}\right) - \frac{\pi}{2}\right] \cdot \left[R_\mathrm{s}^2 \cdot b - \frac{b^3}{3}\right] \\ & + \frac{1}{2} \cdot \left[\arctan\left(\frac{b}{\sqrt{R_\mathrm{s}^2 - a^2 - b^2}}\right) - \frac{\pi}{2}\right] \cdot \left[R_\mathrm{s}^2 \cdot a - \frac{a^3}{3}\right] \\ & + \frac{a \cdot b \cdot \sqrt{R_\mathrm{s}^2 - a^2 - b^2}}{3}. \end{aligned} \tag{11.6}$$

Scheme 3 summarizes the procedure for the octant-corner intersection volume. Note, that this algorithm only uses analytic expressions for the respective volumes. No numeric integration is required.

Algorithm 2 Overlap volume between an octant and a corner

Ensure: $R, x_\mathrm{b}, y_\mathrm{b}, z_\mathrm{b} \geq 0$ ▷ Mirror into the all positive octant

```
procedure OctantVolume(R, x_b, y_b, z_b)
    if x_b^2 + y_b^2 + z_b^2 ≤ R^2 then
        return x_b · y_b · z_b                    ▷ The corner is inside the octant
    end if
    V_Octant ← 1/8 · 4/3 · π · R^3                ▷ Start with a complete octant
    for b ∈ [x_b, y_b, z_b] do
        if b < R then                             ▷ Remove the spherical caps
            V_Octant ← V_Octant − π/12 · (2 · R^3 − 3 · b · R^2 + b^3)
        end if
    end for
    for (a, b) ∈ [(x_b, y_b), (x_b, z_b), (y_b, z_b)] do
        if a^2 + b^2 < R^2 then                   ▷ Add the cap intersections
            V_Octant ← V_Octant + SphereCutVol(R, a, b)
        end if
    end for
    return V_Octant
end procedure
```

Analytic Algorithm for g(r)

We now use this analytic solution for the shell-box intersection volume, $V_{\text{Shell-Box}}$, derived in the last section, to normalize $g(r)$ correctly. To do so, we modify (11.2) by using the actual shell-box intersection volume instead of the simple shell volume. The equation for the local $g(r)$ becomes

$$g_{\text{Local}}(r, \delta r) = \frac{\#P(r, \Delta r)}{V_{\text{Shell-Box}}(r, \Delta r)} \div \frac{\#P_{\text{Sample}}}{V_{\text{Sample}}} \tag{11.7}$$

The local particle density in the shell, at radius r, is now correct even if the shell extends beyond the sample boundaries. We added a detailed description of the full algorithm in Pseudocode and an exemplary Python and C++ implementation in the Supporting Information.

11.4 Results

Improved Accuracy

We tested our new algorithm against an exemplary data set. The data set consists of Gaussian disturbed FCC particles in a cubic sample volume (see SI Section 11.10.1). We used a set of 665 particles to compute $g(r)$ with the simple and our new analytic algorithm. A second set, consisting of 29659 particles, allowed us to compute the correct $g(r)$ as a reference. We computed the correct $g(r)$ by averaging over a smaller subset (see SI section 11.13) of the 29659 particles and limiting the radial distance. Our results are summarized in Figure 11.5. The correct $g(r)$ (blue line) and the analytic $g(r)$(green line) overlap almost perfectly.

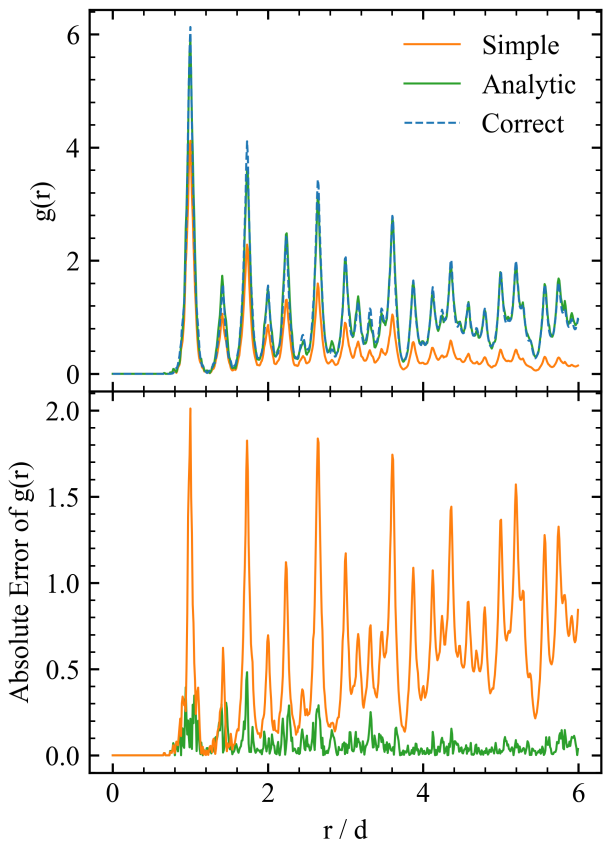


Fig. 11.5.: (a) Comparison of the correct $g(r)$ (dashed blue line) and $g(r)$ computed with the simple (orange line) and our new analytic algorithm (green line). The correct $g(r)$ almost perfectly matches the analytic $g(r)$ (dashed blue line and solid green line overlap). (b) Absolute error of $g(r)$ computed with the two algorithms. Note that our analytic algorithm has significantly less error than the simple algorithm.

This overlap also manifests itself in a small absolute error, shown in Figure 11.5b. In contrast, the simple $g(r)$ (orange line) is significantly different over the entire radial distance. Large absolute errors are observed, especially at the peak positions. Even the first maxima, at small radii, are affected. This proves the importance of normalizing $g(r)$ correctly in finite samples. Quantifying the error in $g(r)$ is not well understood and has hardly been discussed up to now. We expect that the main parameters influencing the error are the particle number, size, distance, surface-to-volume ratio, and the surface roughness. An accurate and quantitative error analysis of $g(r)$ in finite samples is now feasible with our analytical analysis. The unambiguous evaluation of these presumably interrelating parameters, however, requires rigorous experiments and is beyond the scope of our contribution.

Maximal Useful Radial Distance

Our algorithm extends the radial distance, for which $g(r)$ can be computed correctly, in finite samples. However, there is an intrinsic limit. This limit is created by the finite nature of the sample. Figure 11.6 shows arectangular sample volume and $g(r)$ computed from a set of FCC particles in this sample volume.

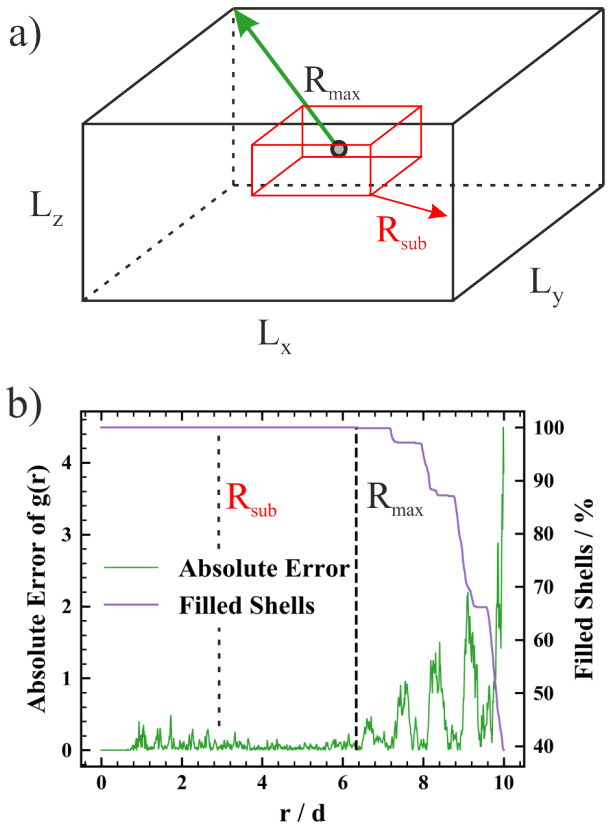


Fig. 11.6.: (a) Rectangular box with dimensions L_x, L_y, and L_z. This box is the boundary of our sample. Averaging over the red subvolume could be used to avoid the normalization problem. The radius of a circumscribed sphere, R_{max}, is the maximal useful radial distance for this sample. (b) Absolute error of $g(r)$ computed with our analytic algorithm. The error is small for radii smaller than R_{max}.

A subvolume and the corresponding limiting radius are shown in red. The subvolume approach has a severe trade-off between radial distance and noise reduction. But, more importantly, this limit is much higher than the limit when we avoid the normalization problem by averaging over a small subvolume (see SI Section 11.13). All particles outside the red volume are not available as central particles. Our approach removes this limitation and increases the maximal radial distance. The radius of the circumscribed sphere

$$R_{max} = \frac{1}{2} \cdot \sqrt{L_x^2 + L_y^2 + L_z^2} \tag{11.8}$$

is the maximal useful radial distance for a sample. The absolute error of g(r) increases significantly beyond R_{max}. This increase in error indicates a natural upper bound. We think that the increase in error is associated with the occurrence of empty shells. Assume we chose a particle at the center of the sample volume as our central particle. Every shell will contain particles up to the radial distance, R_{max}. Shells beyond R_{max}

are empty, because they are fully outside the sample volume. Furthermore, their intersection volume with the box is also empty. This leaves us with an undefined state, because we divide zero particles by zero volume. Removing these cases from the computation reduces the shells available for averaging (see Figure 11.5b). This in turn increases the noise and therefore the absolute error. Radial distances beyond $R_{\max}$ should therefore not be considered for the computation of $g(r)$.

Computational Performance

We tested the computational performance of our analytic algorithm by comparing it to the simple algorithm. We implemented both algorithms in C++ and measured the runtime as a function of the particle number, N. Figure 11.7a compares the runtime for the two algorithms.

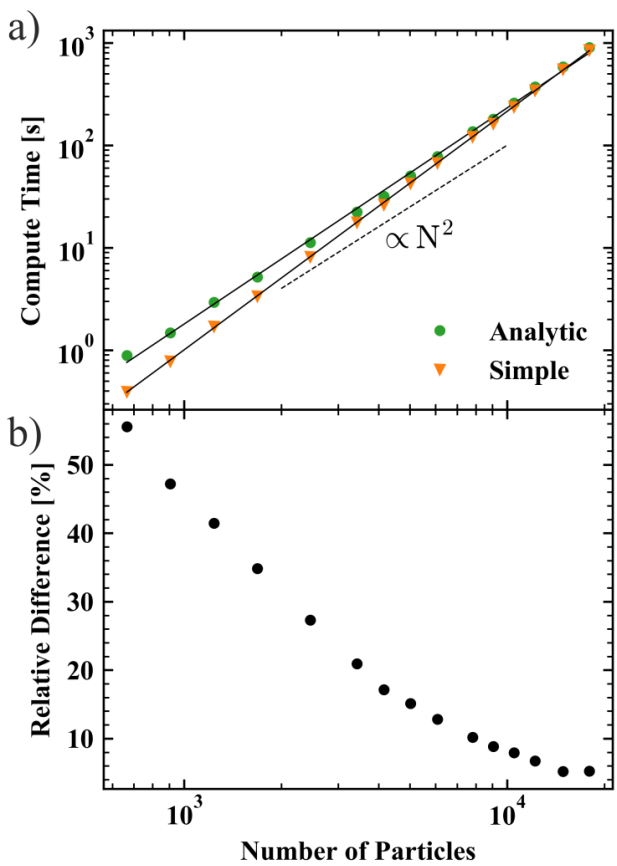


Fig. 11.7.: (a) Runtime comparison for the two $g(r)$ algorithms as a function of particle number. Both algorithms have a computational complexity close to $\mathcal{O}(N^2)$. (b) Relative difference in computation time between the simple and analytic algorithm. The increase in runtime for our analytic algorithm becomes negligible for large particle sets.

We benchmarked all programs on a Dell Latitude E7440 Notebook with an Intel Core i5-4300 clocked at 2.5 GHz and 8 GB RAM. Our implementation of the analytic algorithm in C++ needs 15 min to compute $g(r)$ from 17969 particle coordinates. Both algorithms have an approximated complexity of $\mathcal{O}(N^2)$). This result is expected, because the local $g(r)$ is computed for every particle (N times) in the set. Computing the local $g(r)$ requires a look at every neighbor (N times) of the central particle. Therefore, the computational complexity should scale with N^2. We can explain the slight divergence from N^2 as follows. Our implementation computes $g(r)$ up to $R_{\max}$. This distance increases with increasing particle number, because the sample volume becomes larger. Therefore, the number of shells increases with increasing radial distance. This leads to a slight divergence from the expected $\mathcal{O}(N^2)$ behavior. However, our implementation is not optimized for computational performance. For instance, the particle positions can be stored in a k-d tree data structure. This would allow an efficient nearest neighbor search around the central particle.[23] Furthermore, the computation of the individual local $g(r)$'s can be executed concurrently.

11.5 Conclusion

We presented an analytic algorithm for the computation of the correct 3D radial distribution function, $g(r)$, from real space particle coordinates. It is now possible to compute the radial distribution function accurately without further assumptions about the measured sample. This is possible through a normalization of $g(r)$ using the intersection volume between a spherical shell and the sample volume. This algorithm will allow for a more quantitative and detailed analysis of experimentally obtained particle packings and object spacings. With the advent of widespread experimental access to reconstructed 3D data, this work will improve the analysis of particulate systems and help uncover long-distance correlations in finite samples.

Associated Content

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.analchem.8b03157. Detailed derivation of the equations together with exemplary implementations of the simple and analytic $g(r)$ algorithm in Python 3, C++, and Pseudocode.

Acknowledgments

B.A.F.K. acknowledges support from the Elite Network of Bavaria. We thank Martin Dulle for fruitful discussions and help with OVITO. We thank Astrid Rauh for introducing us to the challenge of computing $g(r)$ in finite volumes.

11.6 References

[1] A. van Blaaderen and P. Wiltzius, „Real-space structure of colloidal hard-sphere glasses," *Science*, vol. 270, no. 5239, pp. 1177–1179, 1995.

[2] G. D. Scott, „Radial distribution of the random close packing of equal spheres," *Nature*, vol. 194, no. 4832, pp. 956–957, 1962.

[3] Z. Adamczyk, M. Zembala, B. Siwek, and P. Warszyński, „Structure and ordering in localized adsorption of particles," *Journal of Colloid and Interface Science*, vol. 140, no. 1, pp. 123–137, 1990.

[4] R. Jeanneret and D. Bartolo, „Geometrically protected reversibility in hydrodynamic loschmidt-echo experiments," *Nature Communications*, vol. 5, no. 1, 2014.

[5] C. Scholz and T. Pöschel, „Velocity distribution of a homogeneously driven two-dimensional granular gas," *Physical Review Letters*, vol. 118, p. 198 003, 19 2017.

[6] H. H. Hwang and M. J. Kushner, „Simulation of the formation of two-dimensional coulomb liquids and solids in dusty plasmas," *Journal of Applied Physics*, vol. 82, no. 5, pp. 2106–2114, 1997.

[7] S. Torquato and F. H. Stillinger, „Exactly solvable disordered sphere-packing model in arbitrary-dimensional euclidean spaces," *Physical Review E*, vol. 73, p. 031 106, 3 2006.

[8] G. Zhang and S. Torquato, „Precise algorithm to generate random sequential addition of hard hyperspheres at saturation," *Physical Review E*, vol. 88, p. 053 312, 5 2013.

[9] S. Bucciarelli, J. S. Myung, B. Farago, *et al.*, „Dramatic influence of patchy attractions on short-time protein diffusion under crowded conditions," *Science Advances*, vol. 2, no. 12, 2016.

[10] P. Ercius, O. Alaidi, M. J. Rames, and G. Ren, „Electron tomography: A three-dimensional analytic tool for hard and soft materials research," *Advanced Materials*, vol. 27, no. 38, pp. 5638–5663, 2015.

[11] J. Miao, P. Ercius, and S. J. L. Billinge, „Atomic electron tomography: 3d structures without crystals," *Science*, vol. 353, no. 6306, 2016.

[12] E. J. Keliher, Y.-X. Ye, G. R. Wojtkiewicz, *et al.*, „Polyglucose nanoparticles with renal elimination and macrophage avidity facilitate PET imaging in ischaemic heart disease," *Nature Communications*, vol. 8, no. 1, 2017.

[13] S. Jiang, H. Wang, Y. Wu, *et al.*, „Ultrastrong steel via minimal lattice misfit and high-density nanoprecipitation," *Nature*, vol. 544, no. 7651, pp. 460–464, 2017.

[14] J.-P. Hansen and I. R. McDonald, *Theory of Simple Liquids - with Applications to Soft Matter*, 4. Aufl. Amsterdam, Boston: Academic Press, 2013.

[15] D. Frenkel and B. Smit, *Understanding Molecular Simulation - From Algorithms to Applications*, 2. Aufl. Amsterdam: Elsevier, 2001.

[16] A. Stukowski, „Visualization and analysis of atomistic simulation data with OVITO–the open visualization tool," *Modelling and Simulation in Materials Science and Engineering*, vol. 18, no. 1, p. 015 012, 2009.

[17] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, „Mdanalysis: A toolkit for the analysis of molecular dynamics simulations," *Journal of Computational Chemistry*, vol. 32, no. 10, pp. 2319–2327, 2011.

[18] K. Younge, C. Christenson, A. Bohara, J. Crnkovic, and P. Saulnier, „A model system for examining the radial distribution function," *American Journal of Physics*, vol. 72, no. 9, pp. 1247–1250, 2004.

[19] B. Freireich, M. Kodam, and C. Wassgren, „An exact method for determining local solid fractions in discrete element method simulations," *AIChE Journal*, vol. 56, no. 12, pp. 3036–3048, 2010.

[20] N. Vogel, S. Utech, G. T. England, *et al.*, „Color from hierarchy: Diverse optical properties of micron-sized spherical colloidal assemblies," *Proceedings of the National Academy of Sciences*, vol. 112, no. 35, pp. 10 845–10 850, 2015.

[21] G. E. Mueller, „Radial void fraction distributions in randomly packed fixed beds of uniformly sized spheres in cylindrical containers," *Powder Technology*, vol. 72, no. 3, pp. 269–275, 1992.

[22] S. Strobl, A. Formella, and T. Pöschel, „Exact calculation of the overlap volume of spheres and mesh elements," *Journal of Computational Physics*, vol. 311, pp. 158–172, 2016.

[23] J. L. Bentley, „Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

11.7 Supporting Information

11.8 Theory

11.8.1 How to Compute $g(r)$ from Particle Coordinates

This section describes how the radial distribution function, $g(r)$, is computed from particle coordinates. First, we look at the relation between $g(r)$ and the local particle density

$$\rho(r) = \rho_{\text{avg}} \cdot g(r) \quad \Leftrightarrow \quad g(r) = \frac{1}{\rho_{\text{avg}}} \cdot \rho(r). \tag{11.9}$$

The average density, ρ_{avg}, is given by the number of all particles, N, divided by the known sample volume, V_{Sample}, like

$$\rho_{\text{avg}} = \frac{N}{V_{\text{Sample}}}. \tag{11.10}$$

The actual local particle density, $\rho_{\text{real}}(r)$, at one specific radius, r, is hard to obtain. Technically, $\rho_{\text{real}}(r)$, is defined by

$$\rho_{\text{real}}(r) = \lim_{\Delta r \to 0} \frac{\#\text{P}(\text{r} - \frac{\Delta \text{r}}{2}, \text{r} + \frac{\Delta \text{r}}{2})}{V_{\text{Shell}}(r - \frac{\Delta r}{2}, r + \frac{\Delta r}{2})}. \tag{11.11}$$

Here, $\#P$ denotes the number of particles in a spherical shell with the given inner and outer radii. This definition has some issues when it comes to actually computing $\rho(r)$ and $g(r)$. Both the volume of a spherical shell and the number of particles in the shell approach zero when Δr approaches zero. In practice, this issue is solved by using a finite Δr. The equation for the estimated local particle density is therefore

$$\rho_{\text{calc}}(r, \Delta r) = \frac{\#\text{P}(\text{r} - \frac{\Delta \text{r}}{2}, \text{r} + \frac{\Delta \text{r}}{2})}{V_{\text{Shell}}(r - \frac{\Delta r}{2}, r + \frac{\Delta r}{2})}. \tag{11.12}$$

Before we analyze the implications of using a finite, Δr, we take a look at the shell volume. The volume of a spherical shell is given by the difference of the inner and outer bounding sphere. Therefore, we have

$$V_{\text{Shell}}(r, \Delta r) = V_{\text{sphere}}(r + \frac{\Delta r}{2}) - V_{\text{sphere}}(r - \frac{\Delta r}{2}) = \frac{4}{3}\pi \left[\left(r + \frac{\Delta r}{2} \right)^3 - \left(r - \frac{\Delta r}{2} \right)^3 \right] \tag{11.13}$$

We simplify this equation by expanding the cube roots into

$$\frac{4}{3}\pi \left[r^3 + 3r^2 \cdot \frac{\Delta r}{2} + 3r \cdot \frac{\Delta r^2}{4} + \frac{\Delta r^3}{8} - \left[r^3 - 3r^2 \cdot \frac{\Delta r}{2} + 3r \cdot \frac{\Delta r^2}{4} - \frac{\Delta r^3}{8} \right] \right]. \tag{11.14}$$

Next, we add all terms with equal power, which leaves us with

$$\boxed{V_{\text{Shell}}(r, \Delta r) = \frac{4}{3} \cdot \pi \cdot \left[3 \cdot r^2 \cdot \Delta r + \frac{\Delta r^3}{4}\right] = 4\pi \cdot r^2 \cdot \Delta r + \pi \cdot \frac{\Delta r^3}{3}}. \tag{11.15}$$

This equation gives us a simple way to calculate the volume of a spherical shell. Equation (11.12) becomes

$$\rho_{\text{calc}}(r, \Delta r) = \frac{\#\text{P}(\text{r} - \frac{\Delta \text{r}}{2}, \text{r} + \frac{\Delta \text{r}}{2})}{4\pi r^2 \Delta r + \pi \frac{\Delta r^3}{3}}. \tag{11.16}$$

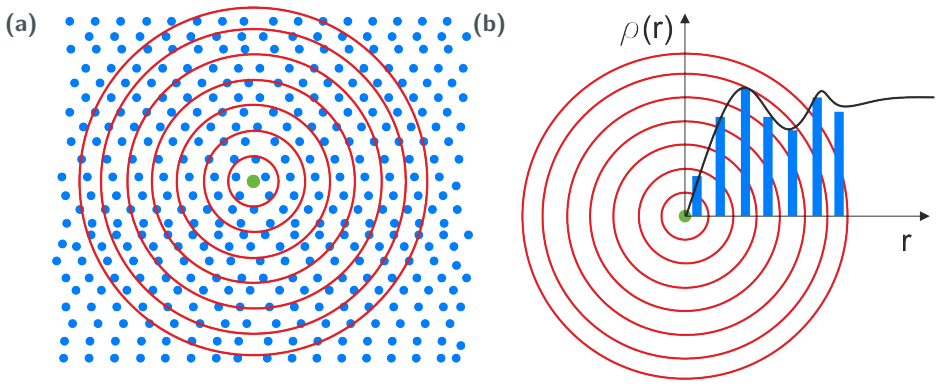


Fig. 11.8.: (a) Slightly distorted cubic grid of particles. The red circles create spherical shells around the central green particle. Each shell has a certain number of blue particles in it. (b) Histogram showing the particle density, $\rho(r)$, as a function of radial distance.

Figure 11.8 illustrates the procedure defined in equation (11.12). We move the center of our radial coordinate system onto a single particle. The red shells have a thickness of Δr. The average density in each shell is the number of particles divided by the shell volume (see equation (11.10)). The radial distribution function around a single particle, i, is therefore given by

$$g_i(r, \Delta r) = \frac{1}{\rho_{\text{avg}}} \cdot \frac{\#\text{P}(\text{r} - \frac{\Delta \text{r}}{2}, \text{r} + \frac{\Delta \text{r}}{2})}{4\pi r^2 \Delta r + \pi \frac{\Delta r^3}{3}}. \tag{11.17}$$

We now average over all possible particles

$$g_{\text{calc}}(r, \Delta r) = \frac{1}{N} \cdot \sum_{i=1}^{N} g_i(r, \Delta r), \tag{11.18}$$

to get better statistics. Note, that we assumed a homogeneous particle distribution.

11.8.2 The Effects of Artificial Periodic Boundaries

In this section, we look at the effecs of applying artificial periodic boundaries to our sample. Figure 11.9 shows a small sample volume with periodic boundaries. The spherical shells for the $g(r)$ computation are now always inside the sample volume.

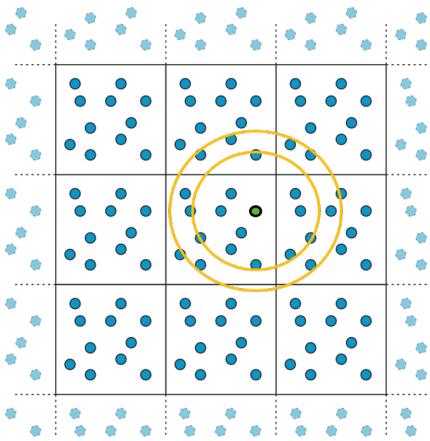


Fig. 11.9.: Small sample volume extended to infinity by periodic boundary conditions. Note the creation of artificial periodicities.

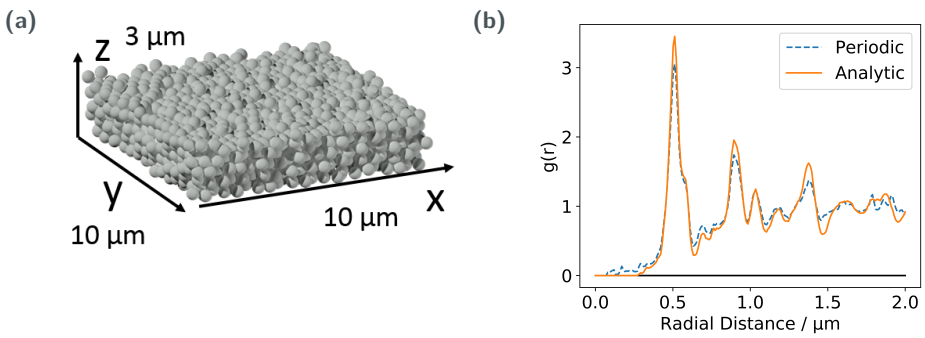


Fig. 11.10.: (a) Colloidal assemble measured with confocal scanning microscopy and analyzed with TrackPy. (b) Radial distribution function for the particle assembly in a. Periodic boundaries distort the peak shape and position.

The effects of periodic boundaries are most pronounced for small samples. Larger finite sample volumes are less affected by periodic boundaries. The main effects are:

- Stitching errors from unknown sample boundary positions
- Artificial periodicities, especially in small samples
- Rough boundaries from missing particles distort $g(r)$

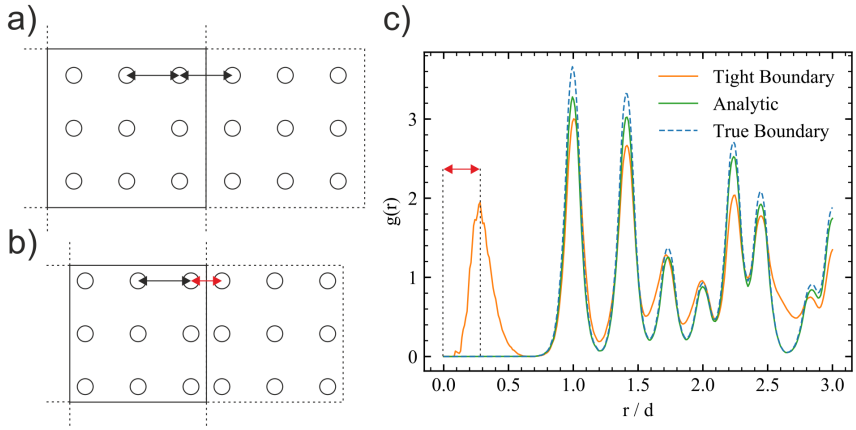


Fig. 11.11.: a) Simple cubic lattice with periodic boundaries a half next neighbor distance away from the outer most particles. No stitching errors (artificial periodicities) are created. b) Same lattice as in a) but with a tighter boundary. The red distance is artificial. c) $g(r)$ of a simple cubic lattice with 2197 particles arranged in a cubic sample volume. The unit cell has size 1 and the particle positions are distorted with a 3D Gaussian blur ($\sigma = 0.07$). The graph labeled true boundary corresponds to a) and the tight boundary to b). Our analytic solution uses a tight boundary but reproduces the true boundary case well. Note the large peak at $r < 1$ created by the stitching error when a tight periodic boundary is used. The size of this peak strongly depends on the particle number in the sample and the sample shape. The position of the peak is at $r > 1$ when the boundary is more then a half next neighbor distance away from the outer most particle.

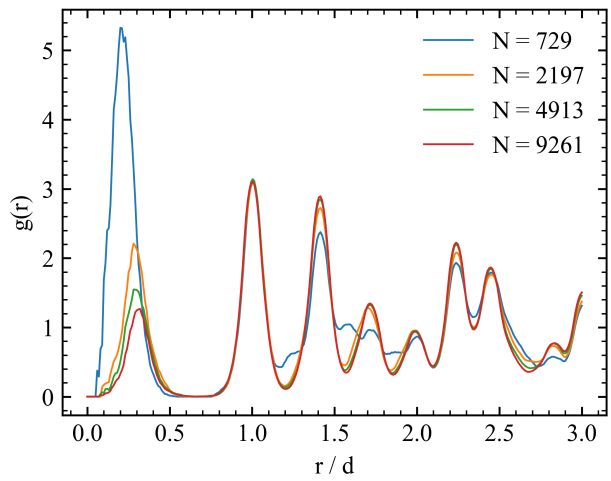


Fig. 11.12.: $g(r)$ of a simple cubic lattice arranged in a cubic sample volume similar to Figure 11.11 c). We increase the number of particles in the sample volume from 729 to 9261. The unit cell is 1. The artificial peak at $r < 1$ is reduced when the particle number is increased. We attribute this decrease to the decreasing surface to volume ratio.

11.8.3 The Effects of Averaging Over a Small Subvolume

In this section we look at a way of avoiding the normalization problem. We avoid it by confining the averaging over the local $g(r)$ to a small sub-volume. At the same time, we limit the radial distance as shown in Figure 11.13. The spherical bins now never reach beyond the finite sample volume.

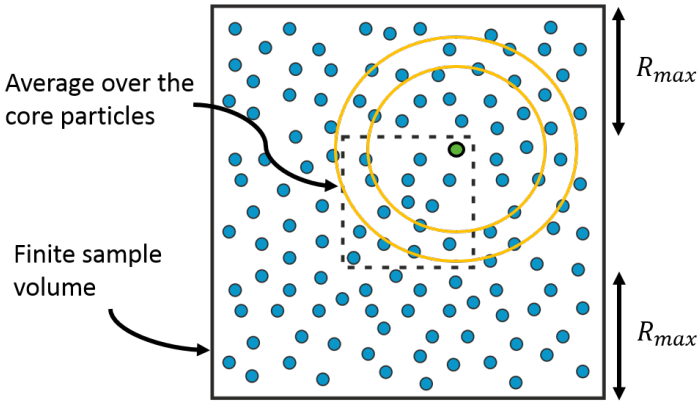


Fig. 11.13.: Finite sample volume with a smaller subvolume (dashed lines). The radial distance is limited to $R_{\max}$. Averaging is only perfomed over the particles in the subvolume. The spherical bins (yellow) never reach beyond the sample boundaries.

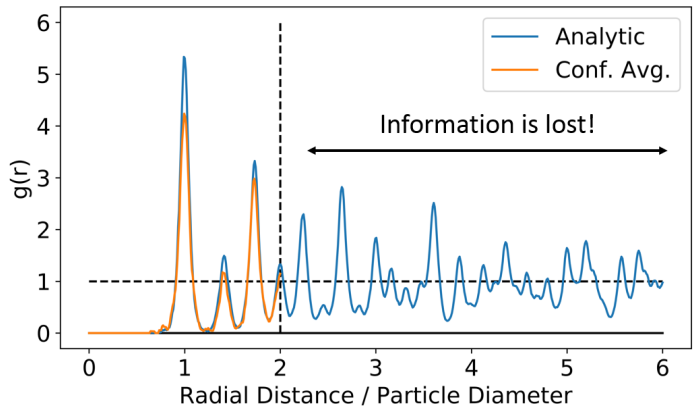


Fig. 11.14.: Radial distribution function for our FCC test data set. The confined average algorithm produces the same $g(r)$ as the analytic algorithm. However, radial distances beyond $R_{\max}$ are not accessible and information is lost.

The main effect of this method is a limited radial distance. There is also a trade-off between radial distance and noise reduction. Less particles are available for averaging, increasing the noise.

11.8.4 The Intersection Volume Between a Shell and a Box

In this section, we derive the intersection volume between a spherical shell and a rectangular box. During the calculation of $g(r)$, the volume of a spherical shell, occupied by particles, is needed. In practice, the box where we have information about the particle positions is finite. Therefore, we need the intersection volume between the spherical shell and the box to properly normalize $g(r)$. We assume that the center of the spherical shell is always inside the box. This assumption is valid since the center of our shell is a known particle. Furthermore, we assume that the walls of the box are perpendicular to each other.

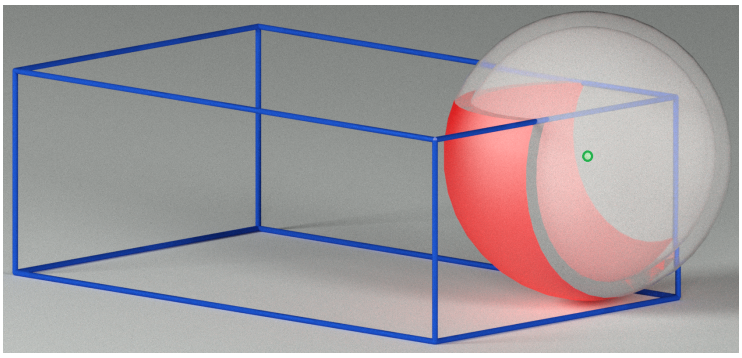


Fig. 11.15.: Intersection volume (red) between a spherical shell and a rectangular box (blue). The green particle is at the center of the spherical shell. The red volume is needed to normalize $g(r)$.

An example rendering of a spherical shell, protruding outside of a box, is shown in Figure 11.15. The red volume is the intersection volume between the spherical shell and the blue box.

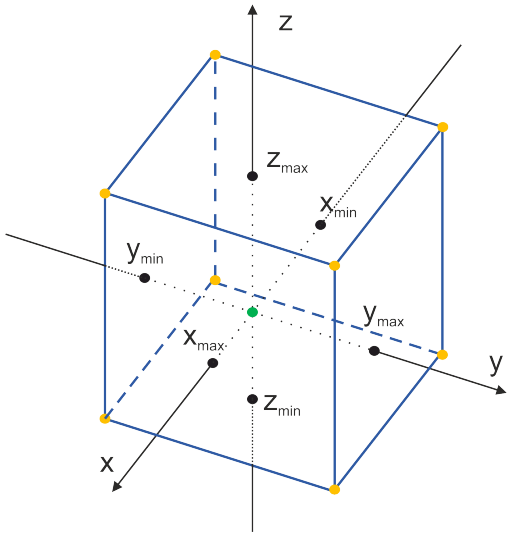


Fig. 11.16.: Definition of the box boundaries. Without loss of generality, we assume that the center of this coordinate system is at the center of the current particle. We can always shift the center of the COSY because the coordinates of the central particle are known.

The boundaries of the box are defined in Figure 11.16. The center of our Cartesian coordinate system is at the center of the special shell. To calculate the red intersection volume we proceed as follows. First, we dived the problem into smaller subproblems that are easier to solve. We then solve the subproblems and reassemble their solutions into the final solution.

We note that the volume of a spherical shell is given by the volume of the inner sphere minus the outer sphere like

$$V_{\text{shell}} = V_{\text{outer sphere}} - V_{\text{inner sphere}}. \tag{11.19}$$

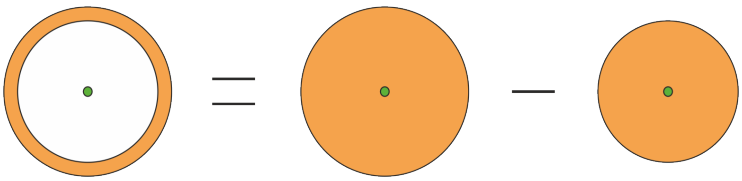


Fig. 11.17.: The volume of a spherical shell is given by the difference of the inner and outer sphere. This is true as long as the inner sphere is fully inside the outer sphere.

A graphical proof of equation (11.19) is given in Figure 11.17. The intersection volume between the smaller sphere and the box is always completely inside the intersection volume of the larger sphere and the box. Therefore, equation (11.19) also holds when both spheres are intersected by a box. We reduced our original problem to the problem of finding the intersection volume of a sphere and a box.

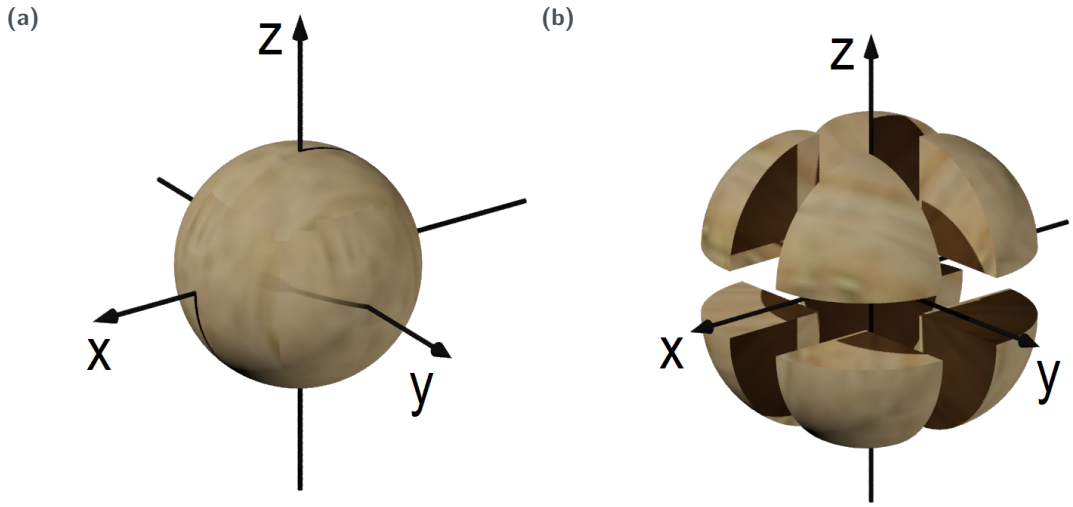


Fig. 11.18.: Separation of a sphere (a) into eight octants (b) along the planes of the Cartesian coordinate system. The octants are moved away from the center for clarity.

Next, we use the intrinsic symmetries present in a sphere and a box. To do so, we split the sphere into eight octants as shown in Figure 11.18. Note, that the walls of our rectangular box are perpendicular to the axes of the Cartesian coordinate system. Three walls of our box cut into each octant at most, since the center of the sphere is always inside the box. Therefore, we reduced the original problem to finding the remaining volume of an octant intersected by three perpendicular boundaries. Overall, the sphere volume is then given by the sum of the remaining octant - box intersection volumes like

$$V_{\text{sphere}} = \sum_{i=1}^{8} V_{\text{Octant}}(i). \tag{11.20}$$

For simplicity, we mirror all octants into the first octant bounded by three positive axes. This mirroring reduces the number of different cases significantly. Mirroring preserves the volume. The labels for the new boundaries are shown in Figure 11.19.

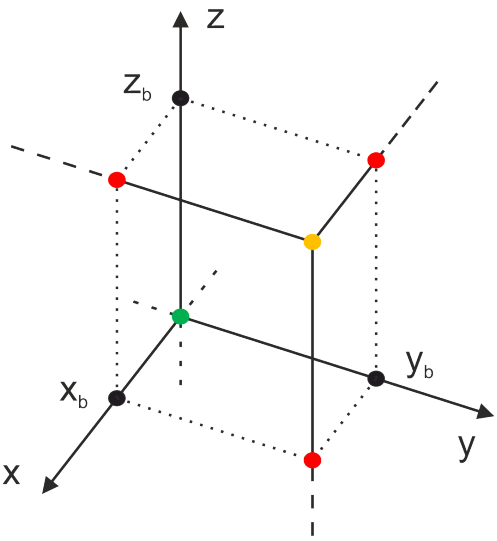


Fig. 11.19.: The three boundaries: x_b, y_b, and z_b, defining the dimensions of the corner. Note that we can always mirror a corner into the all positive octant of the COSY. This does not affect the intersection volume between the corner and the sphere octant.

We are looking for a simple algorithm to calculate the intersection volume between a sphere and a box. Using the above definitions and simplifications we arrive at the following function:

```
def SphereVolume(Rs, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax):

    VSphere = 0
    # abs() mirrors the boundaries into the first octant
    for xb in [abs(Xmin), abs(Xmax)]:
        for yb in [abs(Ymin), abs(Ymax)]:
            for zb in [abs(Zmin), abs(Zmax)]:
                VSphere += OctVolume(Rs, xb, yb, zb)
    return VSphere
```

This leaves us with the challenge of finding a simple solution for the intersection volume of a box corner with a sphere octant (OctVolume(R_s, x_b, y_b, z_b)). Again we look for patterns that might simplify this task. First we look at the case that the intersection volume is only bounded by the box and not by the sphere. This is the case when

$$x_b^2 + y_b^2 + z_b^2 \leq R_s^2. \tag{11.21}$$

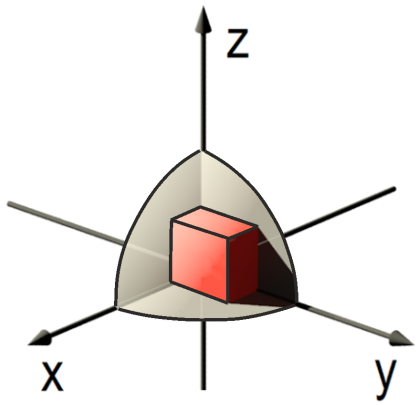


Fig. 11.20.: Intersection volume (red) between a box corner and an octant. The box corner is fully inside the octant.

Figure 11.20 shows the case of a box corner inside the octant. In this case, the orange box corner in Figure 11.19 is inside or on the sphere. Therefore, the intersection volume is given by:

$$\text{OctVolume}(R_\text{s}, x_\text{b}, y_\text{b}, z_\text{b}) = x_\text{b} \cdot y_\text{b} \cdot z_\text{b}. \tag{11.22}$$

This is a special case and we have to treat it as such. Next, we look at a single plane intersecting the octant. This situation is shown in Figure 11.21.

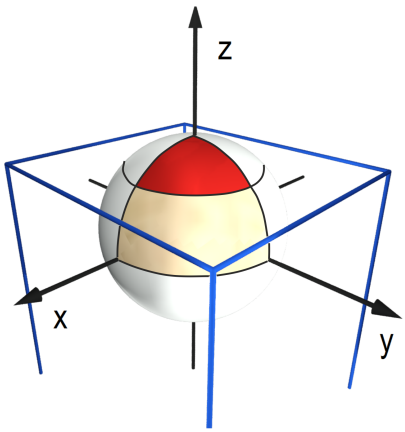


Fig. 11.21.: Single boundary intersecting the sphere octant. A quarter of a spherical cap (red) is removed from the octant by the plane (blue).

A single plane removes a quarter of a spherical cap (red) from the octant. For now, we assume that the three planes do not intersect themselves. A boundary plane, B, intersects the octant only when it is closer to the origin than the sphere radius:

$$B < R_\text{s}. \tag{11.23}$$

In this case, we subtract a quarter of a spherical cap, derived in appendix 11.8.7,

$$\frac{1}{4} \cdot V_{\text{cap}}(R_{\text{s}}, B) = \frac{\pi}{4} \cdot \left[\frac{2}{3} \cdot R_{\text{s}}^3 - R_{\text{s}}^2 \cdot B + \frac{1}{3} \cdot B^3\right] \tag{11.24}$$

from the octant. The last thing left to consider is the overlap between two spherical caps. Two boundary planes, a, and b, intersect each other inside the octant, when

$$a^2 + b^2 < R_{\text{s}}^2. \tag{11.25}$$

This case is shown in Figure 11.22.

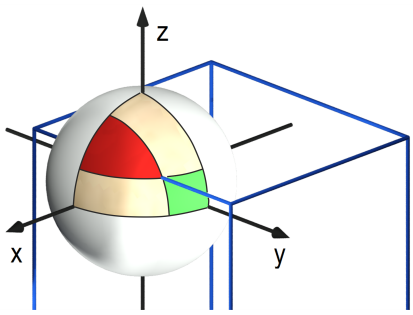


Fig. 11.22.: Two boundaries intersecting the octant. The boundaries also intersect themselves inside the octant. During the first step, we subtracted the red volume twice from the octant. Therefore, we have to add the intersection volume between the two spherical caps (red).

For simplicity, we name the intersection volume of two spherical caps a spherical cut. The volume of a spherical cut is derived in section 11.8.5. For every pair of boundaries, we have to check if they intersect. If they do, we need to add the spherical cut volume, since we removed it twice when we removed the spherical caps. Overall, we obtain the following algorithm for the volume of a sphere octant intersected by up to three planes:

```
def OctVolume(Rs, xb, yb, zb):

    # if all boundaries intersect inside the octant
    if xb**2 + yb**2 + zb**2 < Rs**2:
        return xb * yb * zb
    # if they dont intersect we start with a full octant
    VOctant = 1/8 * 4/3 * pi * Rs**3
    # then,  we remove the spherical caps
    for B in [xb, yb, zb]:
        if B < Rs:
            VOctant -= pi/4*(2/3*Rs**3-B*Rs**2+1/3*B**3)
    # finally, we add the intersections of the caps
    for (a, b) in [(xb, yb), (xb, zb), (yb, zb)]:
        if a**2 + b**2 < Rs**2:
            VOctant += VSphereCut(Rs, a, b)
    return VOctant
```

With these two algorithms and equation (11.19) we are able to efficiently calculate the intersection volume between a spherical shell and a rectangular box.

11.8.5 The Volume of a Spherical Cut

In this section, we derive the volume of a spherical cut. Without loss of generality, we limit ourselves to the X- and Z-boundary planes. Figure 11.23 shows a 3D rendering of the spherical cut in red. The red volume is created by slicing a sphere twice, once perpendicular to the X-, and once perpendicular to the Z-axis.

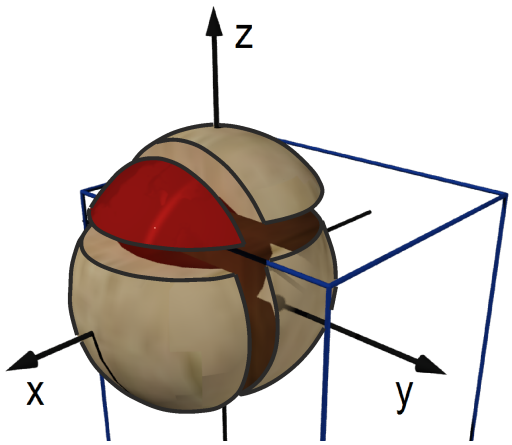


Fig. 11.23.: Spherical cut (red) created by slicing a sphere with two planes perpendicular to the X and Z axis. The parts created by the cuts are exploded for better visualization. Note that we defined the cut volume to be half of the red volume.

The integral to obtain the red volume directly is hard to solve. Therefore, we look for a decomposition into easier integrals. We choose the decomposition shown in Figure 11.24.

Fig. 11.24.: Decomposition used to calculate the volume of a spherical cut (red) by adding and subtracting different volumes from the spherical part. Note that the volumes are rendered from two octants for easier understanding. Only the internal volume (green) is hard to obtain.

Using this decomposition, the volume of a spherical cut is given by

$$V_{\text{cut}}(R_{\text{s}}, x_{\text{b}}, z_{\text{b}}) = \frac{1}{8} \cdot V_{\text{sphere}}(R_{\text{s}}) - \frac{1}{4} \cdot V_{\text{seg}}(R_{\text{s}}, x_{\text{b}}) - \frac{1}{4} \cdot V_{\text{seg}}(R_{\text{s}}, z_{\text{b}}) + V_{\text{int}}(R_{\text{s}}, x_{\text{b}}, z_{\text{b}}). \tag{11.26}$$

All volumes, except for the green, internal volume, V_{int} are known. We insert the volume of the spherical segment derived in section 11.8.6 and get

$$V_{\text{cut}}(R_{\text{s}}, x_{\text{b}}, z_{\text{b}}) = \frac{1}{8} \cdot \frac{4}{3}\pi R_{\text{s}}^3 - \frac{\pi}{4} \cdot \left[R_{\text{s}}^2 \cdot x_{\text{b}} - \frac{x_{\text{b}}^3}{3}\right] - \frac{\pi}{4} \cdot \left[R_{\text{s}}^2 \cdot z_{\text{b}} - \frac{z_{\text{b}}^3}{3}\right] + V_{\text{int}}(R_{\text{s}}, x_{\text{b}}, z_{\text{b}}). \tag{11.27}$$

The analytic solution for the internal volume, V_{int}, is given in appendix 11.8.8. We insert the solution for V_{int} into equation (11.27) and obtain

$$\begin{aligned} V_{\text{cut}}(R_{\text{s}}, x_{\text{b}}, z_{\text{b}}) = {} & \frac{1}{8} \cdot \frac{4}{3}\pi R_{\text{s}}^3 - \frac{\pi}{4} \cdot \left[R_{\text{s}}^2 \cdot x_{\text{b}} - \frac{x_{\text{b}}^3}{3}\right] - \frac{\pi}{4} \cdot \left[R_{\text{s}}^2 \cdot z_{\text{b}} - \frac{z_{\text{b}}^3}{3}\right] + \\ & \frac{z_{\text{b}} \cdot x_{\text{b}} \cdot \sqrt{R_{\text{s}}^2 - x_{\text{b}}^2 - z_{\text{b}}^2}}{3} + \frac{1}{2} \cdot \arctan\left(\frac{x_{\text{b}}}{\sqrt{R_{\text{s}}^2 - x_{\text{b}}^2 - z_{\text{b}}^2}}\right) \cdot \left[R_{\text{s}}^2 \cdot z_{\text{b}} - \frac{z_{\text{b}}^3}{3}\right] + \\ & \frac{1}{2} \cdot \arctan\left(\frac{z_{\text{b}}}{\sqrt{R_{\text{s}}^2 - x_{\text{b}}^2 - z_{\text{b}}^2}}\right) \cdot \left[R_{\text{s}}^2 \cdot x_{\text{b}} - \frac{x_{\text{b}}^3}{3}\right] - \frac{R_{\text{s}}^3}{3} \cdot \arctan\left(\frac{z_{\text{b}} \cdot x_{\text{b}}}{R_{\text{s}} \cdot \sqrt{R_{\text{s}}^2 - x_{\text{b}}^2 - z_{\text{b}}^2}}\right). \end{aligned} \tag{11.28}$$

Next, we factor out the $R_{\text{s}}^3/6$ terms in the square brackets to simplify the equation into

$$\begin{aligned} V_{\text{cut}}(R_{\text{s}}, x_{\text{b}}, z_{\text{b}}) = {} & \frac{R_{\text{s}}^3}{6} \cdot \left[\pi - 2 \cdot \arctan\left(\frac{z_{\text{b}} \cdot x_{\text{b}}}{R_{\text{s}} \cdot \sqrt{R_{\text{s}}^2 - x_{\text{b}}^2 - z_{\text{b}}^2}}\right)\right] \\ & + \frac{1}{2} \cdot \left[\arctan\left(\frac{z_{\text{b}}}{\sqrt{R_{\text{s}}^2 - x_{\text{b}}^2 - z_{\text{b}}^2}}\right) - \frac{\pi}{2}\right] \cdot \left[R_{\text{s}}^2 \cdot x_{\text{b}} - \frac{x_{\text{b}}^3}{3}\right] \\ & + \frac{1}{2} \cdot \left[\arctan\left(\frac{x_{\text{b}}}{\sqrt{R_{\text{s}}^2 - x_{\text{b}}^2 - z_{\text{b}}^2}}\right) - \frac{\pi}{2}\right] \cdot \left[R_{\text{s}}^2 \cdot z_{\text{b}} - \frac{z_{\text{b}}^3}{3}\right] \\ & + \frac{z_{\text{b}} \cdot x_{\text{b}} \cdot \sqrt{R_{\text{s}}^2 - x_{\text{b}}^2 - z_{\text{b}}^2}}{3}. \end{aligned} \tag{11.29}$$

This is the final analytic solution for the volume of the spherical cut, V_{cut}. In general, a boundary pair, (a, b), for which

$$a^2 + b^2 < R_{\text{s}}^2 \tag{11.30}$$

holds, creates the following spherical cut volume:

$$
\begin{aligned}
V_{\mathrm{cut}}(R_{\mathrm{s}}, a, b) = {} & \frac{R_{\mathrm{s}}^3}{6} \cdot \left[\pi - 2 \cdot \arctan\left(\frac{a \cdot b}{R_{\mathrm{s}} \cdot \sqrt{R_{\mathrm{s}}^2 - a^2 - b^2}}\right)\right] \\
& + \frac{1}{2} \cdot \left[\arctan\left(\frac{a}{\sqrt{R_{\mathrm{s}}^2 - a^2 - b^2}}\right) - \frac{\pi}{2}\right] \cdot \left[R_{\mathrm{s}}^2 \cdot b - \frac{b^3}{3}\right] \\
& + \frac{1}{2} \cdot \left[\arctan\left(\frac{b}{\sqrt{R_{\mathrm{s}}^2 - a^2 - b^2}}\right) - \frac{\pi}{2}\right] \cdot \left[R_{\mathrm{s}}^2 \cdot a - \frac{a^3}{3}\right] \\
& + \frac{a \cdot b \cdot \sqrt{R_{\mathrm{s}}^2 - a^2 - b^2}}{3}.
\end{aligned}
\tag{11.31}
$$

11.8.6 The Volume of a Spherical Segment

In this section, we derive the volume of a spherical segment. A spherical segment of height, B, is shown in Figure 11.25.

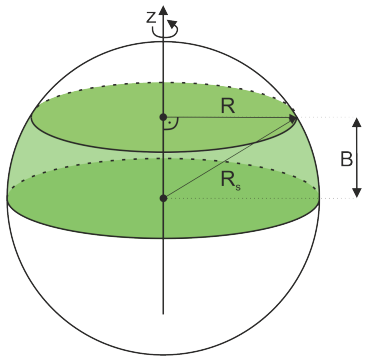


Fig. 11.25.: Spherical segment (green) of height, B, inside a sphere of radius, R_s.

A spherical segment can be viewed as a rotation body. The rotation axis is vertical (z) and the radius, R, depends on the height. Using Pythagoreans theorem, we see that

$$R^2 + z^2 = R_\mathrm{s}^2 \quad \Leftrightarrow \quad R^2 = R_\mathrm{s}^2 - z^2. \tag{11.32}$$

To get the volume of the spherical segment, V_seg, we integrate over tiny discs along the z-axis like

$$V_\mathrm{seg}(R_\mathrm{s}, B) = \int_0^B \pi \cdot R^2 \,\mathrm{d}z = \int_0^B \pi \cdot \left[R_\mathrm{s}^2 - z^2\right] \,\mathrm{d}z. \tag{11.33}$$

Then, we split the integral at the minus sign and factor out the sphere radius, R_s, to get

$$\pi \cdot \left[\int_0^B R_\mathrm{s}^2 \; dz - \int_0^B z^2 \,\mathrm{d}z\right] = \pi \cdot \left[R_\mathrm{s}^2 \cdot \int_0^B 1 \,\mathrm{d}z - \int_0^B z^2 \,\mathrm{d}z\right]. \tag{11.34}$$

Finally, we solve the integrals by

$$\pi \cdot \left[R_\mathrm{s}^2 \cdot [z]_0^B - \left[\frac{1}{3} \cdot z^3\right]_0^B\right] = \pi \cdot \left[R_\mathrm{s}^2 \cdot [B - 0] - \left[\frac{1}{3} \cdot B^3 - \frac{1}{3} \cdot 0^3\right]\right], \tag{11.35}$$

which leaves us with

$$\boxed{V_\mathrm{seg}(R_\mathrm{s}, B) = \pi \cdot \left[R_\mathrm{s}^2 \cdot B - \frac{1}{3} \cdot B^3\right]}. \tag{11.36}$$

This is the final equation for the volume of a spherical segment as a function of the boundary position, B, and the sphere radius, R_s.

11.8.7 The Volume of a Spherical Cap

In this section, we derive the volume of a spherical cap. A spherical cap is shown in Figure 11.26. This volume is very similar to the volume of a spherical segment.

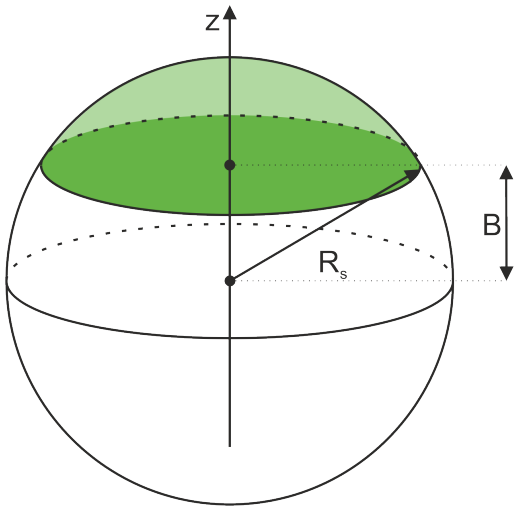


Fig. 11.26.: Spherical cap (green) with radius, R_s, and height, $R_\mathrm{s} - B$.

First, we note that a spherical cap is just a hemi-sphere minus a spherical segment:

$$V_\mathrm{cap}(R_\mathrm{s}, B) = \frac{1}{2} \cdot \frac{4}{3}\pi R_\mathrm{s}^3 - V_\mathrm{seg}(R_\mathrm{s}, B) \tag{11.37}$$

We insert the equation for a spherical segment, derived in appendix 11.8.6, and simplify the first term to get

$$V_\mathrm{cap}(R_\mathrm{s}, B) = \frac{2}{3}\pi R_\mathrm{s}^3 - \pi \cdot \left[R_\mathrm{s}^2 \cdot B - \frac{1}{3} \cdot B^3\right]. \tag{11.38}$$

Finally, we factor out π and obtain

$$\boxed{V_\mathrm{cap}(R_\mathrm{s}, B) = \pi \cdot \left[\frac{2}{3} \cdot R_\mathrm{s}^3 - R_\mathrm{s}^2 \cdot B + \frac{1}{3} \cdot B^3\right]}. \tag{11.39}$$

This is the final equation for the volume of a spherical cap as a function of the boundary position, B, from the sphere center.

11.8.8 The Internal Volume V_{int}

In this section, we derive the volume of the internal element, V_{int}. Figure 11.27 shows a rotated drawing of V_{int} inside a Cartesian coordinate system.

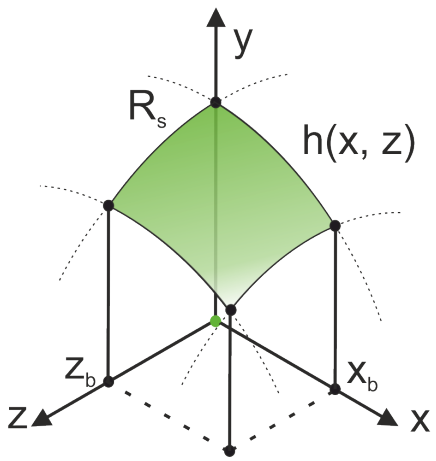


Fig. 11.27.: Integration in Cartesian coordinates yields the volume under a 3D surface. In our case, the 3D surface (green) is part of the surface of a sphere.

To obtain V_{int}, we integrate a section of the surface of a sphere. The section is bounded by perpendicular planes through x_{b} and z_{b}. These planes are also orthogonal to their respective coordinate axis. A sphere, with its center at the origin, is defined by

$$x^2 + y^2 + z^2 = R_{\text{s}}^2. \tag{11.40}$$

The height, $h(x, z) = y$, is therefore

$$h(x, z) = \sqrt{R_{\text{s}}^2 - z^2 - x^2}. \tag{11.41}$$

To obtain the volume under the green surface, we integrate along the x and z coordinate. The integration boundaries are from 0 to the values x_{b} and z_{b} respectively, like

$$V_{\text{int}}(R_{\text{s}}, x_{\text{b}}, z_{\text{b}}) = \int_0^{x_{\text{b}}} \int_0^{z_{\text{b}}} h(x, z)\,\mathrm{d}z\,\mathrm{d}x = \int_0^{x_{\text{b}}} \int_0^{z_{\text{b}}} \sqrt{R_{\text{s}}^2 - z^2 - x^2}\,\mathrm{d}z\,\mathrm{d}x. \tag{11.42}$$

We solve the two nested integrals one after another. First we integrate over the z-coordinate. Applying the anti-derivative, given in appendix 11.8.9, we get

$$V_{\text{int}} = \int_0^{x_{\text{b}}} \left[\frac{1}{2} \cdot \left[z \cdot \sqrt{R_{\text{s}}^2 - z^2 - x^2} + \left(R_{\text{s}}^2 - x^2\right) \cdot \arctan\left(\frac{z}{\sqrt{R_{\text{s}}^2 - z^2 - x^2}}\right)\right]\right]_0^{z_{\text{b}}} \mathrm{d}x. \tag{11.43}$$

We insert the upper and lower bound, yielding

$$V_{\text{int}} = \int_0^{x_\text{b}} \frac{1}{2} \cdot \left[z_\text{b} \cdot \sqrt{R_\text{s}^2 - z_\text{b}^2 - x^2} + \left(R_\text{s}^2 - x^2\right) \cdot \arctan\left(\frac{z_\text{b}}{\sqrt{R_\text{s}^2 - z_\text{b}^2 - x^2}}\right)\right] \tag{11.44}$$
$$- \frac{1}{2} \cdot \left[0 \cdot \sqrt{R_\text{s}^2 - 0^2 - x^2} + \left(R_\text{s}^2 - x^2\right) \cdot \arctan\left(\frac{0}{\sqrt{R_\text{s}^2 - 0^2 - x^2}}\right)\right] \mathrm{d}x.$$

Since $\arctan(0) = 0$, we are left with

$$V_{\text{int}} = \int_0^{x_\text{b}} \frac{1}{2} \cdot \left[z_\text{b} \cdot \sqrt{R_\text{s}^2 - z_\text{b}^2 - x^2} + \left(R_\text{s}^2 - x^2\right) \cdot \arctan\left(\frac{z_\text{b}}{\sqrt{R_\text{s}^2 - z_\text{b}^2 - x^2}}\right)\right] \mathrm{d}x. \tag{11.45}$$

Next, we split the remaining integral into three parts. We also bring the factor of 2 onto the side of V_{int}. This gives us

$$\begin{aligned} 2 \cdot V_{\text{int}} = z_\text{b} \cdot & \int_0^{x_\text{b}} \sqrt{R_\text{s}^2 - z_\text{b}^2 - x^2}\, \mathrm{d}x \\ + R_\text{s}^2 \cdot & \int_0^{x_\text{b}} \arctan\left(\frac{z_\text{b}}{\sqrt{R_\text{s}^2 - z_\text{b}^2 - x^2}}\right) \mathrm{d}x \\ - & \int_0^{x_\text{b}} x^2 \cdot \arctan\left(\frac{z_\text{b}}{\sqrt{R_\text{s}^2 - z_\text{b}^2 - x^2}}\right) \mathrm{d}x. \end{aligned} \tag{11.46}$$

We now have three individual integrals to solve. We label them according to

$$2 \cdot V_{\text{int}} = z_\text{b} \cdot I_1 + R_\text{s}^2 \cdot I_2 - I_3. \tag{11.47}$$

For simplicity, we define the terms

$$A = R_\text{s}^2 - z_\text{b}^2 \quad \text{and} \quad S = \sqrt{R_\text{s}^2 - z_\text{b}^2 - x_\text{b}^2} = \sqrt{A - x_\text{b}^2}, \tag{11.48}$$

that are always greater than 0. The anti-derivative of the first integral,

$$I_1 = \int_0^{x_\text{b}} \sqrt{R_\text{s}^2 - z_\text{b}^2 - x^2}\, \mathrm{d}x = \int_0^{x_\text{b}} \sqrt{A - x^2}\, \mathrm{d}x \tag{11.49}$$

is solved in appendix 11.8.9 and yields

$$I_1 = \left[\frac{1}{2} \cdot \left[x \cdot \sqrt{A - x^2} + A \cdot \arctan\left(\frac{x}{\sqrt{A - x^2}}\right)\right]\right]_0^{x_\text{b}}. \tag{11.50}$$

Inserting the upper and lower boundaries gives us

$$I_1 = \frac{1}{2}\cdot\left[x_\mathrm{b}\cdot\sqrt{A-x_\mathrm{b}^2} + A\cdot\arctan\left(\frac{x_\mathrm{b}}{\sqrt{A-x_\mathrm{b}^2}}\right)\right] \tag{11.51}$$
$$-\frac{1}{2}\cdot\left[0\cdot\sqrt{A-0^2} + A\cdot\arctan\left(\frac{0}{\sqrt{A-0^2}}\right)\right].$$

Since $\arctan(0) = 0$, we are left with

$$\boxed{I_1 = \frac{1}{2}\cdot\left[x_\mathrm{b}\cdot S + A\cdot\arctan\left(\frac{x_\mathrm{b}}{S}\right)\right]}. \tag{11.52}$$

The second integral,

$$I_2 = \int_0^{x_\mathrm{b}} \arctan\left(\frac{z_\mathrm{b}}{\sqrt{R_\mathrm{s}^2 - z_\mathrm{b}^2 - x^2}}\right)\,\mathrm{d}x = \int_0^{x_\mathrm{b}} \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)\,\mathrm{d}x, \tag{11.53}$$

is solved in appendix 11.8.10 and yields

$$I_2 = \left[z_\mathrm{b}\cdot\arctan\left(\frac{x}{\sqrt{A-x^2}}\right)\right. \tag{11.54}$$
$$+x\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)$$
$$\left.-R_\mathrm{s}\cdot\arctan\left(\frac{x\cdot z_\mathrm{b}}{R_\mathrm{s}\cdot\sqrt{A-x^2}}\right)\right]_0^{x_\mathrm{b}}.$$

Inserting the upper and lower boundaries gives us

$$I_2 = z_\mathrm{b}\cdot\arctan\left(\frac{x_\mathrm{b}}{\sqrt{A-x_\mathrm{b}^2}}\right) \tag{11.55}$$
$$+x_\mathrm{b}\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x_\mathrm{b}^2}}\right)$$
$$-R_\mathrm{s}\cdot\arctan\left(\frac{x_\mathrm{b}\cdot z_\mathrm{b}}{R_\mathrm{s}\cdot\sqrt{A-x_\mathrm{b}^2}}\right)$$
$$-\left[z_\mathrm{b}\cdot\arctan\left(\frac{0}{\sqrt{A-0^2}}\right)\right.$$
$$+0\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-0^2}}\right)$$
$$\left.-R_\mathrm{s}\cdot\arctan\left(\frac{0\cdot z_\mathrm{b}}{R_\mathrm{s}\cdot\sqrt{A-0^2}}\right)\right]$$

Since $\arctan(0) = 0$, we are left with

$$\boxed{I_2 = z_\mathrm{b} \cdot \arctan\left(\frac{x_\mathrm{b}}{S}\right) + x_\mathrm{b} \cdot \arctan\left(\frac{z_\mathrm{b}}{S}\right) - R_\mathrm{s} \cdot \arctan\left(\frac{x_\mathrm{b} \cdot z_\mathrm{b}}{R_\mathrm{s} \cdot S}\right)}. \tag{11.56}$$

The third integral,

$$I_3 = \int_0^{x_\mathrm{b}} x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{R_\mathrm{s}^2 - z_\mathrm{b}^2 - x^2}}\right) \mathrm{d}x = \int_0^{x_\mathrm{b}} x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) \mathrm{d}x, \tag{11.57}$$

is solved in the appendix 11.8.11, like

$$\begin{aligned} &\Bigg[-\frac{z_\mathrm{b}}{6} \cdot x \cdot \sqrt{A - x^2} - \frac{z_\mathrm{b}}{6} \cdot \left(z_\mathrm{b}^2 - 3 \cdot R_\mathrm{s}^2\right) \cdot \arctan\left(\frac{x}{\sqrt{A - x^2}}\right) \\ &+ \frac{x^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) - \frac{R_\mathrm{s}^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b} \cdot x}{R_\mathrm{s} \cdot \sqrt{A - x^2}}\right) \Bigg]_0^{x_\mathrm{b}}. \end{aligned} \tag{11.58}$$

Inserting the upper and lower boundaries gives us

$$\begin{aligned} &-\frac{z_\mathrm{b}}{6} \cdot x_\mathrm{b} \cdot \sqrt{A - x_\mathrm{b}^2} - \frac{z_\mathrm{b}}{6} \cdot \left(z_\mathrm{b}^2 - 3 \cdot R_\mathrm{s}^2\right) \cdot \arctan\left(\frac{x_\mathrm{b}}{\sqrt{A - x_\mathrm{b}^2}}\right) \\ &+ \frac{x_\mathrm{b}^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x_\mathrm{b}^2}}\right) - \frac{R_\mathrm{s}^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b} \cdot x_\mathrm{b}}{R_\mathrm{s} \cdot \sqrt{A - x_\mathrm{b}^2}}\right) \\ &-\Bigg[-\frac{z_\mathrm{b}}{6} \cdot 0 \cdot \sqrt{A - 0^2} - \frac{z_\mathrm{b}}{6} \cdot \left(z_\mathrm{b}^2 - 3 \cdot R_\mathrm{s}^2\right) \cdot \arctan\left(\frac{0}{\sqrt{A - 0^2}}\right) \\ &+ \frac{0^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - 0^2}}\right) - \frac{R_\mathrm{s}^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b} \cdot 0}{R_\mathrm{s} \cdot \sqrt{A - 0^2}}\right) \Bigg]. \end{aligned} \tag{11.59}$$

Since $\arctan(0) = 0$, we are left with

$$I_3 = \frac{-x_\mathrm{b} z_\mathrm{b} S}{6} - \frac{z_\mathrm{b}}{6}\left(z_\mathrm{b}^2 - 3R_\mathrm{s}^2\right) \arctan\left(\frac{x_\mathrm{b}}{S}\right) + \tag{11.60}$$

$$\frac{x_\mathrm{b}^3}{3} \arctan\left(\frac{z_\mathrm{b}}{S}\right) - \frac{R_\mathrm{s}^3}{3} \arctan\left(\frac{z_\mathrm{b} x_\mathrm{b}}{R_\mathrm{s} S}\right). \tag{11.61}$$

Now, we combine the solutions for the three integrals and get:

$$\begin{aligned} 2 \cdot V_\mathrm{int} = z_\mathrm{b} \cdot &\left[\frac{1}{2} \cdot \left[x_\mathrm{b} \cdot S + \left(R_\mathrm{s}^2 - z_\mathrm{b}^2\right) \cdot \arctan\left(\frac{x_\mathrm{b}}{S}\right)\right]\right] \\ +R_\mathrm{s}^2 \cdot &\left[z_\mathrm{b} \cdot \arctan\left(\frac{x_\mathrm{b}}{S}\right) + x_\mathrm{b} \cdot \arctan\left(\frac{z_\mathrm{b}}{S}\right) - R_\mathrm{s} \cdot \arctan\left(\frac{z_\mathrm{b} \cdot x_\mathrm{b}}{R_\mathrm{s} \cdot S}\right)\right] \\ -[&\frac{-x_\mathrm{b} \cdot z_\mathrm{b} \cdot S}{6} - \frac{z_\mathrm{b}}{6} \cdot \left(z_\mathrm{b}^2 - 3R_\mathrm{s}^2\right) \cdot \arctan\left(\frac{x_\mathrm{b}}{S}\right) + \\ &\frac{x_\mathrm{b}^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b}}{S}\right) - \frac{R_\mathrm{s}^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b} \cdot x_\mathrm{b}}{R_\mathrm{s} \cdot S}\right)]. \end{aligned} \tag{11.62}$$

Next, we expand all products to remove the brackets

$$
\begin{aligned}
2 \cdot V_{\text{int}} = & \frac{z_{\text{b}} \cdot x_{\text{b}} \cdot S}{2} + \frac{z_{\text{b}} \cdot (R_{\text{s}}^2 - z_{\text{b}}^2)}{2} \cdot \arctan\left(\frac{x_{\text{b}}}{S}\right) \\
& + R_{\text{s}}^2 \cdot z_{\text{b}} \cdot \arctan\left(\frac{x_{\text{b}}}{S}\right) + R_{\text{s}}^2 \cdot x_{\text{b}} \cdot \arctan\left(\frac{z_{\text{b}}}{S}\right) - R_{\text{s}}^3 \cdot \arctan\left(\frac{z_{\text{b}} \cdot x_{\text{b}}}{R_{\text{s}} \cdot S}\right) \\
& + \frac{x_{\text{b}} \cdot z_{\text{b}} \cdot S}{6} + \frac{z_{\text{b}}}{6} \cdot \left(z_{\text{b}}^2 - 3R_{\text{s}}^2\right) \cdot \arctan\left(\frac{x_{\text{b}}}{S}\right) - \frac{x_{\text{b}}^3}{3} \cdot \arctan\left(\frac{z_{\text{b}}}{S}\right) \\
& + \frac{R_{\text{s}}^3}{3} \cdot \arctan\left(\frac{z_{\text{b}} \cdot x_{\text{b}}}{R_{\text{s}} \cdot S}\right).
\end{aligned} \tag{11.63}
$$

We then factor out the arcus tangens functions

$$
\begin{aligned}
2 \cdot V_{\text{int}} = & \frac{z_{\text{b}} \cdot x_{\text{b}} \cdot S}{2} + \frac{x_{\text{b}} \cdot z_{\text{b}} \cdot S}{6} \\
& + \arctan\left(\frac{x_{\text{b}}}{S}\right) \cdot \left[\frac{z_{\text{b}} \cdot (R_{\text{s}}^2 - z_{\text{b}}^2)}{2} + R_{\text{s}}^2 \cdot z_{\text{b}} + \frac{z_{\text{b}}}{6} \cdot \left(z_{\text{b}}^2 - 3R_{\text{s}}^2\right)\right] \\
& + \arctan\left(\frac{z_{\text{b}}}{S}\right) \cdot \left[R_{\text{s}}^2 \cdot x_{\text{b}} - \frac{x_{\text{b}}^3}{3}\right] + \arctan\left(\frac{z_{\text{b}} \cdot x_{\text{b}}}{R_{\text{s}} \cdot S}\right) \cdot \left[\frac{R_{\text{s}}^3}{3} - R_{\text{s}}^3\right].
\end{aligned} \tag{11.64}
$$

Next, we take a closer look at the term in the square brackets containing z_{b} and see that

$$
\begin{aligned}
\frac{z_{\text{b}} \cdot (R_{\text{s}}^2 - z_{\text{b}}^2)}{2} + R_{\text{s}}^2 \cdot z_{\text{b}} + \frac{z_{\text{b}}}{6} \cdot \left(z_{\text{b}}^2 - 3R_{\text{s}}^2\right) = & \\
\frac{z_{\text{b}} \cdot R_{\text{s}}^2}{2} - \frac{z_{\text{b}}^3}{2} + R_{\text{s}}^2 \cdot z_{\text{b}} + \frac{z_{\text{b}}^3}{6} - \frac{z_{\text{b}} \cdot R_{\text{s}}^2}{2} = & \\
R_{\text{s}}^2 \cdot z_{\text{b}} - \frac{z_{\text{b}}^3}{3} &
\end{aligned} \tag{11.65}
$$

Using equation (11.65), we simplify the terms into

$$
\begin{aligned}
2 \cdot V_{\text{int}} = & \frac{2 \cdot z_{\text{b}} \cdot x_{\text{b}} \cdot S}{3} + \arctan\left(\frac{x_{\text{b}}}{S}\right) \cdot \left[R_{\text{s}}^2 \cdot z_{\text{b}} - \frac{z_{\text{b}}^3}{3}\right] \\
& + \arctan\left(\frac{z_{\text{b}}}{S}\right) \cdot \left[R_{\text{s}}^2 \cdot x_{\text{b}} - \frac{x_{\text{b}}^3}{3}\right] + \arctan\left(\frac{z_{\text{b}} \cdot x_{\text{b}}}{R_{\text{s}} \cdot S}\right) \cdot R_{\text{s}}^3 \cdot \left[-\frac{2}{3}\right].
\end{aligned} \tag{11.66}
$$

Finally, we bring the factor of 2 on the other side

$$
\begin{aligned}
V_{\text{int}} = & \frac{z_{\text{b}} \cdot x_{\text{b}} \cdot S}{3} + \frac{1}{2} \cdot \arctan\left(\frac{x_{\text{b}}}{S}\right) \cdot \left[R_{\text{s}}^2 \cdot z_{\text{b}} - \frac{z_{\text{b}}^3}{3}\right] \\
& + \frac{1}{2} \cdot \arctan\left(\frac{z_{\text{b}}}{S}\right) \cdot \left[R_{\text{s}}^2 \cdot x_{\text{b}} - \frac{x_{\text{b}}^3}{3}\right] - \frac{R_{\text{s}}^3}{3} \cdot \arctan\left(\frac{z_{\text{b}} \cdot x_{\text{b}}}{R_{\text{s}} \cdot S}\right),
\end{aligned} \tag{11.67}
$$

and substitute $S = \sqrt{R_\mathrm{s}^2 - x_\mathrm{b}^2 - z_\mathrm{b}^2}$ back into the equation

$$V_\mathrm{int} = \frac{z_\mathrm{b} \cdot x_\mathrm{b} \cdot \sqrt{R_\mathrm{s}^2 - x_\mathrm{b}^2 - z_\mathrm{b}^2}}{3} + \frac{1}{2} \cdot \arctan\left(\frac{x_\mathrm{b}}{\sqrt{R_\mathrm{s}^2 - x_\mathrm{b}^2 - z_\mathrm{b}^2}}\right) \cdot \left[R_\mathrm{s}^2 \cdot z_\mathrm{b} - \frac{z_\mathrm{b}^3}{3}\right] \tag{11.68}$$

$$+ \frac{1}{2} \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{R_\mathrm{s}^2 - x_\mathrm{b}^2 - z_\mathrm{b}^2}}\right) \cdot \left[R_\mathrm{s}^2 \cdot x_\mathrm{b} - \frac{x_\mathrm{b}^3}{3}\right] - \frac{R_\mathrm{s}^3}{3} \cdot \arctan\left(\frac{z_\mathrm{b} \cdot x_\mathrm{b}}{R_\mathrm{s} \cdot \sqrt{R_\mathrm{s}^2 - x_\mathrm{b}^2 - z_\mathrm{b}^2}}\right).$$

This is the final equation for the internal volume, V_int, as a function of the octant radius, R_s, and the boundary positions, x_b and z_b.

11.8.9 Proof for the Integral I_1

In this section, we proof that the integral

$$I_1 = \int \sqrt{A - x^2}\,\mathrm{d}x, \quad (A - x^2) > 0 \tag{11.69}$$

is solved by

$$\frac{1}{2} \cdot \left[x \cdot \sqrt{A - x^2} + A \cdot \arctan\left(\frac{x}{\sqrt{A - x^2}}\right)\right] + \mathcal{C}. \tag{11.70}$$

To do so, we take the derivative of equation (11.70) with respect to x. Note that

$$\frac{\mathrm{d}}{\mathrm{d}x} \arctan(x) = \frac{1}{1 + x^2}. \tag{11.71}$$

The derivative is then given by

$$\frac{\mathrm{d}}{\mathrm{d}x}\frac{1}{2} \cdot \left[x \cdot \sqrt{A - x^2} + A \cdot \arctan\left(\frac{x}{\sqrt{A - x^2}}\right)\right] + \mathcal{C} = \tag{11.72}$$
$$\frac{1}{2} \cdot [1 \cdot \sqrt{A - x^2} + x \cdot \frac{1}{2\sqrt{A - x^2}} \cdot (-2x) +$$
$$A \cdot \frac{1}{1 + \frac{x^2}{A - x^2}} \cdot \frac{\sqrt{A - x^2} - x \cdot \frac{1}{2\sqrt{A - x^2}} \cdot (-2x)}{A - x^2}].$$

We combine all fractions to get

$$\frac{1}{2} \cdot \left[\sqrt{A - x^2} - \frac{x^2}{\sqrt{A - x^2}} + \frac{A}{A - x^2 + x^2} \cdot \left[\sqrt{A - x^2} + \frac{x^2}{\sqrt{A - x^2}}\right]\right]. \tag{11.73}$$

Finally, we simplify the fractions and see that

$$\frac{1}{2} \cdot \left[\sqrt{A - x^2} - \frac{x^2}{\sqrt{A - x^2}} + \sqrt{A - x^2} + \frac{x^2}{\sqrt{A - x^2}}\right] = \sqrt{A - x^2}, \tag{11.74}$$

which is the integrand we were looking for. Therefore, the integral, I_1, is solved by the anti-derivative given in equation (11.70).

11.8.10 Proof for the Integral I_2

In this section, we prove that the integral

$$I_2 = \int \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right) \mathrm{d}x, \quad (A-x^2) > 0 \tag{11.75}$$

is solved by the following anti-derivative

$$z_\mathrm{b} \cdot \arctan\left(\frac{x}{\sqrt{A-x^2}}\right) + x \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right) - R_\mathrm{s} \cdot \arctan\left(\frac{x \cdot z_\mathrm{b}}{R_\mathrm{s} \cdot \sqrt{A-x^2}}\right) + \mathcal{C}. \tag{11.76}$$

To prove that, we calculate the derivative with respect to x like

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}x}\, z_\mathrm{b} \cdot \arctan\left(\frac{x}{\sqrt{A-x^2}}\right) \\ &+\frac{\mathrm{d}}{\mathrm{d}x}\, x \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right) \\ &-\frac{\mathrm{d}}{\mathrm{d}x}\, R_\mathrm{s} \cdot \arctan\left(\frac{x \cdot z_\mathrm{b}}{R_\mathrm{s} \cdot \sqrt{A-x^2}}\right) + \mathcal{C} \end{aligned} \tag{11.77}$$

We calculate the three derivatives individually. The first derivative is given by

$$\frac{\mathrm{d}}{\mathrm{d}x}\, z_\mathrm{b} \cdot \arctan\left(\frac{x}{\sqrt{A-x^2}}\right) = z_\mathrm{b} \cdot \frac{1}{1+\frac{x^2}{A-x^2}} \cdot \frac{1 \cdot \sqrt{A-x^2} - x \cdot \frac{1}{2 \cdot \sqrt{A-x^2}} \cdot (-2x)}{A-x^2} \tag{11.78}$$

we simplify the fractions and get

$$\frac{z_\mathrm{b}}{A-x^2+x^2} \cdot \left[\sqrt{A-x^2} + \frac{x^2}{\sqrt{A-x^2}}\right]. \tag{11.79}$$

The x^2 cancel each other out, leaving us with

$$\frac{z_\mathrm{b}}{A} \cdot \left[\sqrt{A-x^2} + \frac{x^2}{\sqrt{A-x^2}}\right]. \tag{11.80}$$

Next, we create equal denominators in the square brackets

$$\frac{z_\mathrm{b}}{A} \cdot \left[\frac{A-x^2}{\sqrt{A-x^2}} + \frac{x^2}{\sqrt{A-x^2}}\right], \tag{11.81}$$

which leaves us with

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}x}\, z_\mathrm{b} \cdot \arctan\left(\frac{x}{\sqrt{A-x^2}}\right) = \frac{z_\mathrm{b}}{\sqrt{A-x^2}}}. \tag{11.82}$$

The second derivative is

$$\frac{\mathrm{d}}{\mathrm{d}x}\, x\cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right) = \tag{11.83}$$
$$1\cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right) + x\cdot \frac{1}{1+\frac{z_\mathrm{b}^2}{A-x^2}}\cdot \frac{\sqrt{A-x^2}\cdot 0 - z_\mathrm{b}\cdot \frac{1}{2\sqrt{A-x^2}}\cdot(-2x)}{A-x^2}.$$

We combine the main fractions and get

$$\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right) + \frac{x}{A-x^2+z_\mathrm{b}^2}\cdot\frac{z_\mathrm{b}\cdot x}{\sqrt{A-x^2}}. \tag{11.84}$$

Using our definition $A = R_\mathrm{s}^2 - z_\mathrm{b}^2$ we have

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}x}\, x\cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right) = \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right) + \frac{z_\mathrm{b}\cdot x^2}{(R_\mathrm{s}^2-x^2)\cdot\sqrt{A-x^2}}}. \tag{11.85}$$

The third derivative is given by

$$\frac{\mathrm{d}}{\mathrm{d}x}\, R_\mathrm{s}\cdot \arctan\left(\frac{z_\mathrm{b}\cdot x}{R_\mathrm{s}\cdot\sqrt{A-x^2}}\right) = \tag{11.86}$$
$$R_\mathrm{s}\cdot\frac{1}{1+\frac{z_\mathrm{b}^2\cdot x^2}{R_\mathrm{s}^2\cdot(A-x^2)}}\cdot\frac{R_\mathrm{s}\cdot\sqrt{A-x^2}\cdot z_\mathrm{b} - z_\mathrm{b}\cdot x\cdot R_\mathrm{s}\cdot\frac{1}{2\sqrt{A-x^2}}\cdot(-2x)}{R_\mathrm{s}^2\cdot(A-x^2)}.$$

We combine the main fraction while factoring out $z_\mathrm{b}\cdot R_\mathrm{s}$ from the numerator, to get

$$\frac{R_\mathrm{s}^2\cdot z_\mathrm{b}}{R_\mathrm{s}^2\cdot(A-x^2)+z_\mathrm{b}^2\cdot x^2}\cdot\left[\sqrt{A-x^2}+\frac{x^2}{\sqrt{A-x^2}}\right] \tag{11.87}$$

Next, we create identical denominators in the square brackets

$$\frac{R_\mathrm{s}^2\cdot z_\mathrm{b}}{R_\mathrm{s}^2\cdot(A-x^2)+z_\mathrm{b}^2\cdot x^2}\cdot\left[\frac{A-x^2}{\sqrt{A-x^2}}+\frac{x^2}{\sqrt{A-x^2}}\right], \tag{11.88}$$

which leaves us with

$$\frac{R_\mathrm{s}^2\cdot z_\mathrm{b}}{R_\mathrm{s}^2\cdot(A-x^2)+z_\mathrm{b}^2\cdot x^2}\cdot\frac{A}{\sqrt{A-x^2}}, \tag{11.89}$$

Before we proceed, we take a closer look at the largest denominator and see that

$$R_\mathrm{s}^2\cdot\left(A-x^2\right)+z_\mathrm{b}^2\cdot x^2 = \tag{11.90}$$
$$R_\mathrm{s}^2\cdot A - R_\mathrm{s}^2\cdot x^2 + z_\mathrm{b}^2\cdot x^2 =$$
$$R_\mathrm{s}^2\cdot A + x^2\cdot\left(z_\mathrm{b}^2-R_\mathrm{s}^2\right).$$

Since

$$A = R_\mathrm{s}^2 - z_\mathrm{b}^2 \Leftrightarrow -A = (z_\mathrm{b}^2 - R_\mathrm{s}^2), \tag{11.91}$$

we have

$$R_\mathrm{s}^2 \cdot A - A \cdot x^2 = A \cdot \left(R_\mathrm{s}^2 - x^2\right). \tag{11.92}$$

Using this identity in equation (11.89), we get

$$\frac{R_\mathrm{s}^2 \cdot z_\mathrm{b}}{A \cdot (R_\mathrm{s}^2 - x^2)} \cdot \frac{A}{\sqrt{A - x^2}}, \tag{11.93}$$

which leaves us with

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}x}\, R_\mathrm{s} \cdot \arctan\left(\frac{z_\mathrm{b} \cdot x}{R_\mathrm{s} \cdot \sqrt{A - x^2}}\right) = \frac{R_\mathrm{s}^2 \cdot z_\mathrm{b}}{(R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}}}. \tag{11.94}$$

Finally, we combine all the separate derivatives like

$$\frac{z_\mathrm{b}}{\sqrt{A - x^2}} + \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) + \frac{z_\mathrm{b} \cdot x^2}{(R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}} - \frac{R_\mathrm{s}^2 \cdot z_\mathrm{b}}{(R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}}. \tag{11.95}$$

We expand the first fraction to create equal denominators like

$$\frac{(R_\mathrm{s}^2 - x^2) \cdot z_\mathrm{b}}{(R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}} + \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) + \frac{z_\mathrm{b} \cdot x^2}{(R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}} - \frac{R_\mathrm{s}^2 \cdot z_\mathrm{b}}{(R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}}. \tag{11.96}$$

Then, we add all fractions

$$\frac{z_\mathrm{b} \cdot (R_\mathrm{s}^2 - x^2) + z_\mathrm{b} \cdot x^2 - R_\mathrm{s}^2 \cdot z_\mathrm{b}}{(R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}} + \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right), \tag{11.97}$$

and expand the round brackets into

$$\frac{z_\mathrm{b} \cdot R_\mathrm{s}^2 - z_\mathrm{b} \cdot x^2 + z_\mathrm{b} \cdot x^2 - R_\mathrm{s}^2 \cdot z_\mathrm{b}}{(R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}} + \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right). \tag{11.98}$$

We see, that all terms in the numerator cancel each other out, leaving us with

$$\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right), \tag{11.99}$$

which is the integrand we were looking for. Therefore, the integral, I_2, is solved by the anti-derivative given in equation (11.76).

11.8.11 Proof for the Integral I_3

In this section, we proof that the integral

$$I_3 = \int x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) \mathrm{d}x, \quad (A - x^2) > 0 \tag{11.100}$$

is solved by

$$-\frac{z_\mathrm{b}}{6}\cdot x\cdot\sqrt{A-x^2}-\frac{z_\mathrm{b}}{6}\cdot\left(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2\right)\cdot\arctan\left(\frac{x}{\sqrt{A-x^2}}\right) \tag{11.101}$$
$$+\frac{x^3}{3}\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)-\frac{R_\mathrm{s}^3}{3}\cdot\arctan\left(\frac{z_\mathrm{b}\cdot x}{R_\mathrm{s}\cdot\sqrt{A-x^2}}\right)+\mathcal{C}.$$

To do so, we take the derivative of this solution with respect to x like

$$\begin{aligned}-\frac{z_\mathrm{b}}{6}\cdot\frac{\mathrm{d}}{\mathrm{d}x}x\cdot\sqrt{A-x^2}\\ -\frac{z_\mathrm{b}}{6}\cdot\left(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2\right)\cdot\frac{\mathrm{d}}{\mathrm{d}x}\arctan\left(\frac{x}{\sqrt{A-x^2}}\right)\\ +\frac{\mathrm{d}}{\mathrm{d}x}\frac{x^3}{3}\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)\\ -\frac{R_\mathrm{s}^3}{3}\cdot\frac{\mathrm{d}}{\mathrm{d}x}\arctan\left(\frac{z_\mathrm{b}\cdot x}{R_\mathrm{s}\cdot\sqrt{A-x^2}}\right).\end{aligned} \tag{11.102}$$

We calculate the derivatives separately. The first derivative is given by

$$-\frac{z_\mathrm{b}}{6}\cdot\frac{\mathrm{d}}{\mathrm{d}x}x\cdot\sqrt{A-x^2}=-\frac{z_\mathrm{b}}{6}\cdot\left[1\cdot\sqrt{A-x^2}+x\cdot\frac{1}{2\sqrt{A-x^2}}\cdot(-2x)\right]. \tag{11.103}$$

We simplify the terms in the square brackets and get

$$\frac{z_\mathrm{b}}{6}\cdot\left[\frac{x^2}{\sqrt{A-x^2}}-\sqrt{A-x^2}\right]. \tag{11.104}$$

Then, we expand the second term in the square bracket

$$\frac{z_\mathrm{b}}{6}\cdot\left[\frac{x^2}{\sqrt{A-x^2}}-\frac{A-x^2}{\sqrt{A-x^2}}\right], \tag{11.105}$$

and simplify, which leaves us with

$$\boxed{-\frac{z_\mathrm{b}}{6}\cdot\frac{\mathrm{d}}{\mathrm{d}x}x\cdot\sqrt{A-x^2}=\frac{z_\mathrm{b}}{6}\cdot\frac{2\cdot x^2-A}{\sqrt{A-x^2}}}. \tag{11.106}$$

The second derivative is given by

$$\begin{aligned}-\frac{z_\mathrm{b}}{6}\cdot\left(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2\right)\cdot\frac{\mathrm{d}}{\mathrm{d}x}\arctan\left(\frac{x}{\sqrt{A-x^2}}\right)=\\ -\frac{z_\mathrm{b}}{6}\cdot\left(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2\right)\cdot\frac{1}{1+\frac{x^2}{A-x^2}}\cdot\frac{1\cdot\sqrt{A-x^2}-x\cdot\frac{1}{2\sqrt{A-x^2}}\cdot(-2x)}{A-x^2}.\end{aligned} \tag{11.107}$$

We multiply the fractions

$$-\frac{z_\mathrm{b}}{6}\cdot\left(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2\right)\cdot\frac{1}{A-x^2+x^2}\left[\sqrt{A-x^2}+\frac{x^2}{\sqrt{A-x^2}}\right], \tag{11.108}$$

and simplify to get

$$-\frac{z_\mathrm{b}\cdot(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2)}{6\cdot A}\cdot\left[\sqrt{A-x^2}+\frac{x^2}{\sqrt{A-x^2}}\right]. \tag{11.109}$$

Then, we expand the terms in the square brackets

$$-\frac{z_\mathrm{b}\cdot(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2)}{6\cdot A}\cdot\left[\frac{A-x^2}{\sqrt{A-x^2}}+\frac{x^2}{\sqrt{A-x^2}}\right], \tag{11.110}$$

which leaves us with

$$\boxed{-\frac{z_\mathrm{b}}{6}\cdot\left(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2\right)\cdot\frac{\mathrm{d}}{\mathrm{d}x}\arctan\left(\frac{x}{\sqrt{A-x^2}}\right)=-\frac{z_\mathrm{b}\cdot(z_\mathrm{b}^2-3\cdot R_\mathrm{s}^2)}{6\cdot\sqrt{A-x^2}}}. \tag{11.111}$$

The third derivative is given by

$$\frac{\mathrm{d}}{\mathrm{d}x}\frac{x^3}{3}\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)= \tag{11.112}$$

$$x^2\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)+\frac{x^3}{3}\cdot\frac{1}{1+\frac{z_\mathrm{b}^2}{A-x^2}}\cdot\frac{\sqrt{A-x^2}\cdot 0-z_\mathrm{b}\cdot\frac{1}{2\sqrt{A-x^2}}\cdot(-2x)}{A-x^2}.$$

We combine the fractions

$$x^2\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)+\frac{x^3}{3}\cdot\frac{1}{A-x^2+z_\mathrm{b}^2}\cdot\frac{x\cdot z_\mathrm{b}}{\sqrt{A-x^2}}, \tag{11.113}$$

and simplify the denominator, using $A=R_\mathrm{s}^2-z_\mathrm{b}^2$, yielding

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}x}\frac{x^3}{3}\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)=x^2\cdot\arctan\left(\frac{z_\mathrm{b}}{\sqrt{A-x^2}}\right)+\frac{x^4\cdot z_\mathrm{b}}{3\cdot(R_\mathrm{s}^2-x^2)\cdot\sqrt{A-x^2}}}. \tag{11.114}$$

The last derivative is given by

$$-\frac{R_\mathrm{s}^3}{3}\cdot\frac{\mathrm{d}}{\mathrm{d}x}\arctan\left(\frac{z_\mathrm{b}\cdot x}{R_\mathrm{s}\cdot\sqrt{A-x^2}}\right)= \tag{11.115}$$

$$-\frac{R_\mathrm{s}^3}{3}\cdot\frac{1}{1+\frac{z_\mathrm{b}^2\cdot x^2}{R_\mathrm{s}^2\cdot(A-x^2)}}\cdot\frac{R_\mathrm{s}\cdot\sqrt{A-x^2}\cdot z_\mathrm{b}-z_\mathrm{b}\cdot x\cdot R_\mathrm{s}\cdot\frac{1}{2\sqrt{A-x^2}}\cdot(-2x)}{R_\mathrm{s}^2\cdot(A-x^2)}.$$

We combine the fractions, leaving us with

$$-\frac{z_\mathrm{b}\cdot R_\mathrm{s}^4}{3}\cdot\frac{1}{R_\mathrm{s}^2\cdot(A-x^2)+z_\mathrm{b}^2\cdot x^2}\cdot\left[\sqrt{A-x^2}+\frac{x^2}{\sqrt{A-x^2}}\right]. \tag{11.116}$$

Before we proceed, we take a closer look at the largest denominator and see that

$$\begin{aligned} R_\mathrm{s}^2 \cdot \left(A - x^2\right) + z_\mathrm{b}^2 \cdot x^2 = \\ R_\mathrm{s}^2 \cdot A - R_\mathrm{s}^2 \cdot x^2 + z_\mathrm{b}^2 \cdot x^2 = \\ R_\mathrm{s}^2 \cdot A + x^2 \cdot \left(z_\mathrm{b}^2 - R_\mathrm{s}^2\right). \end{aligned} \tag{11.117}$$

Since $A = R_\mathrm{s}^2 - z_\mathrm{b}^2 \Leftrightarrow -A = z_\mathrm{b}^2 - R_\mathrm{s}^2$ we get

$$R_\mathrm{s}^2 \cdot A - A \cdot x^2 = A \cdot \left(R_\mathrm{s}^2 - x^2\right) \tag{11.118}$$

Using this identity, we combine the fractions to get

$$-\frac{R_\mathrm{s}^4 \cdot z_\mathrm{b}}{3 \cdot A \cdot (R_\mathrm{s}^2 - x^2)} \cdot \left[\sqrt{A - x^2} + \frac{x^2}{\sqrt{A - x^2}}\right]. \tag{11.119}$$

Next, we create equal denominator in the square brackets

$$-\frac{R_\mathrm{s}^4 \cdot z_\mathrm{b}}{3 \cdot A \cdot (R_\mathrm{s}^2 - x^2)} \cdot \left[\frac{A - x^2}{\sqrt{A - x^2}} + \frac{x^2}{\sqrt{A - x^2}}\right], \tag{11.120}$$

which leaves us with

$$\boxed{-\frac{R_\mathrm{s}^3}{3} \cdot \frac{\mathrm{d}}{\mathrm{d}x} \arctan\left(\frac{z_\mathrm{b} \cdot x}{R_\mathrm{s} \cdot \sqrt{A - x^2}}\right) = -\frac{R_\mathrm{s}^4 \cdot z_\mathrm{b}}{3 \cdot (R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}}}. \tag{11.121}$$

Finally, we sum up all the individual derivatives and get

$$\begin{aligned} \frac{z_\mathrm{b}}{6} \cdot \frac{2 \cdot x^2 - A}{\sqrt{A - x^2}} - \frac{z_\mathrm{b} \cdot (z_\mathrm{b}^2 - 3 \cdot R_\mathrm{s}^2)}{6 \cdot \sqrt{A - x^2}} + x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) \\ + \frac{x^4 \cdot z_\mathrm{b}}{3 \cdot (R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}} - \frac{R_\mathrm{s}^4 \cdot z_\mathrm{b}}{3 \cdot (R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}}. \end{aligned} \tag{11.122}$$

We add the fractions like

$$\begin{aligned} \frac{2 \cdot x^2 \cdot z_\mathrm{b} - A \cdot z_\mathrm{b} - z_\mathrm{b}^3 + 3 \cdot R_\mathrm{s}^2 \cdot z_\mathrm{b}}{6 \cdot \sqrt{A - x^2}} + x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) \\ + \frac{z_\mathrm{b} \cdot (x^4 - R_\mathrm{s}^4)}{3 \cdot (R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}}. \end{aligned} \tag{11.123}$$

We simplify the second fraction with the third binomic equation:

$$(a + b) \cdot (a - b) = a^2 - b^2. \tag{11.124}$$

This leaves us with

$$\frac{2 \cdot x^2 \cdot z_\mathrm{b} - A \cdot z_\mathrm{b} - z_\mathrm{b}^3 + 3 \cdot R_\mathrm{s}^2 \cdot z_\mathrm{b}}{6 \cdot \sqrt{A - x^2}} + x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) + \frac{z_\mathrm{b} \cdot (x^2 - R_\mathrm{s}^2) \cdot (x^2 + R_\mathrm{s}^2)}{3 \cdot (R_\mathrm{s}^2 - x^2) \cdot \sqrt{A - x^2}} \tag{11.125}$$

We now cancel the $R_\mathrm{s}^2 - x^2 = -(x^2 - R_\mathrm{s}^2)$ terms and are left with

$$\frac{2 \cdot x^2 \cdot z_\mathrm{b} - A \cdot z_\mathrm{b} - z_\mathrm{b}^3 + 3 \cdot R_\mathrm{s}^2 \cdot z_\mathrm{b}}{6 \cdot \sqrt{A - x^2}} + x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) - \frac{2 \cdot z_\mathrm{b} \cdot (x^2 + R_\mathrm{s}^2)}{6 \cdot \sqrt{A - x^2}}. \tag{11.126}$$

Again, we add the fractions with identical denominators. We also use $A = R_\mathrm{s}^2 - z_\mathrm{b}^2$ in the numerator, giving us

$$\frac{2 \cdot x^2 \cdot z_\mathrm{b} - (R_\mathrm{s}^2 - z_\mathrm{b}^2) \cdot z_\mathrm{b} - z_\mathrm{b}^3 + 3 \cdot R_\mathrm{s}^2 \cdot z_\mathrm{b} - 2 \cdot z_\mathrm{b} \cdot (x^2 + R_\mathrm{s}^2)}{6 \cdot \sqrt{A - x^2}} + x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right) \tag{11.127}$$

We expand the terms in the round brackets

$$\frac{2 \cdot x^2 \cdot z_\mathrm{b} - R_\mathrm{s}^2 \cdot z_\mathrm{b} + z_\mathrm{b}^3 - z_\mathrm{b}^3 + 3 \cdot R_\mathrm{s}^2 \cdot z_\mathrm{b} - 2 \cdot z_\mathrm{b} \cdot x^2 - 2 \cdot z_\mathrm{b} \cdot R_\mathrm{s}^2}{6 \cdot \sqrt{A - x^2}} + x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right). \tag{11.128}$$

All terms in the numerator cancel each other out, which leaves us with

$$\boxed{x^2 \cdot \arctan\left(\frac{z_\mathrm{b}}{\sqrt{A - x^2}}\right)}. \tag{11.129}$$

Therefore, the integral, I_3, is solved by the anti-derivative given in equation (11.101).

11.9 Pseudo code

Algorithm 3 Overlap volume between an octant and a corner

Ensure: $R, x_\mathrm{b}, y_\mathrm{b}, z_\mathrm{b} \geq 0$ ▷ Mirror into the all positive octant

procedure OCTANTVOLUME$(R, x_\mathrm{b}, y_\mathrm{b}, z_\mathrm{b})$
  if $x_\mathrm{b}^2 + y_\mathrm{b}^2 + z_\mathrm{b}^2 \leq R^2$ **then**
    return $x_\mathrm{b} \cdot y_\mathrm{b} \cdot z_\mathrm{b}$ ▷ The corner is inside the octant
  end if
  $V_{Octant} \leftarrow \frac{1}{8} \cdot \frac{4}{3} \cdot \pi \cdot R^3$ ▷ Start with a complete octant
  for $b \in [x_\mathrm{b}, y_\mathrm{b}, z_\mathrm{b}]$ **do**
    if $b < R$ **then** ▷ Remove the spherical caps
      $V_{Octant} \leftarrow V_{Octant} - \frac{\pi}{12} \cdot (2 \cdot R^3 - 3 \cdot b \cdot R^2 + b^3)$
    end if
  end for
  for $(a, b) \in [(x_\mathrm{b}, y_\mathrm{b}), (x_\mathrm{b}, z_\mathrm{b}), (y_\mathrm{b}, z_\mathrm{b})]$ **do**
    if $a^2 + b^2 < R^2$ **then** ▷ Add the cap intersections
      $V_{Octant} \leftarrow V_{Octant} + \text{SPHERECUTVOL}(R, a, b)$
    end if
  end for
  return V_{Octant}
end procedure

Algorithm 4 Compute the volume of a spherical cut.

Ensure: $R, A, B \geq 0,\ A^2 + B^2 < R^2$

procedure SPHERECUTVOL(R, A, B)
  $Root \leftarrow \sqrt{R^2 - A^2 - B^2}$
  $V_{cut} \leftarrow \frac{R^3}{6} \cdot \left(\pi - 2 \cdot \arctan\left(\frac{A \cdot B}{R \cdot Root}\right)\right)$
  $V_{cut} \leftarrow V_{cut} + \frac{1}{2} \cdot \left(\arctan\left(\frac{A}{Root}\right) - \frac{\pi}{2}\right) \cdot \left(R^2 \cdot B - \frac{B^3}{3}\right)$
  $V_{cut} \leftarrow V_{cut} + \frac{1}{2} \cdot \left(\arctan\left(\frac{B}{Root}\right) - \frac{\pi}{2}\right) \cdot \left(R^2 \cdot A - \frac{A^3}{3}\right)$
  $V_{cut} \leftarrow V_{cut} + \frac{1}{3} \cdot A \cdot B \cdot Root$
  return V_{cut}
end procedure

Algorithm 5 Compute the shell-box intersection volume.

Ensure: $R_{min} \geq 0,\ R_{max} > 0,\ R_{min} < R_{max},\ x_{min} < x_{max},\ y_{min} < y_{max},\ z_{min} < z_{max}$

procedure SHELLVOLUME$(R_{min}, R_{max}, x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})$
  $InnerShell \leftarrow \text{SPHEREVOLUME}(R_{min}, x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})$
  $OuterShell \leftarrow \text{SPHEREVOLUME}(R_{max}, x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})$
  return $OuterShell - InnerShell$
end procedure

Algorithm 6 Compute the sphere-box intersection volume.

Ensure: $R \geq 0,\ x_{min} < x_{max},\ y_{min} < y_{max},\ z_{min} < z_{max}$

```
procedure SphereVolume(R, Boundaries)
    V_Sphere ← 0                                   ▷ Start with an empty volume
    for x_b ∈ [|x_min|, |x_max|] do
        for y_b ∈ [|y_min|, |y_max|] do
            for z_b ∈ [|z_min|, |z_max|] do
                V_Sphere ← V_Sphere + OctantVolume(R, x_b, y_b, z_b)
            end for
        end for
    end for
    return V_Sphere
end procedure
```

Algorithm 7 Compute the shell-sphere intersection volume.

Ensure: $R_{min} \geq 0,\ R_{max} > 0,\ R_{min} < R_{max}$

```
procedure ShellSphereVolume(R_min, R_max, R_boundary, x_c, y_c, z_c, CentralP)
    dx ← x_c − CentralP.XPos
    dy ← y_c − CentralP.YPos
    dz ← z_c − CentralP.ZPos
    PartD ← √(dx² + dy² + dz²)   ▷ Distance between the sphere and the shell centers
    InnerSphere ← SphereSphereVolume(R_min, R_boundary, PartD)
    OuterSphere ← SphereSphereVolume(R_max, R_boundary, PartD)
    return OuterSphere − InnerSphere
end procedure
```

Algorithm 8 Compute the sphere-sphere intersection volume.

Ensure: $R_{sphere} \geq 0,\ R_{boundary} > 0, Dist > 0$

```
procedure SphereSphereVolume(R_sphere, R_boundary, Dist)
    if PartD > R_sphere + R_boundary then          ▷ Check if the spheres intersect
        return 0
    end if
    A ← R_boundary − (R_boundary² − R_sphere² + Dist²) / (2·Dist)
    B ← (R_boundary² − R_sphere² + PartD²) / (2·Dist) − (Dist − R_sphere)
    InnerCap ← A · (3 · R_boundary − A)
    OuterCap ← B · (3 · R_sphere − B)
    IntersectVol ← (π/3) · (OuterCap + InnerCap)
    return IntersectVol
end procedure
```

Algorithm 9 Compute $g(r)$ from a particle set bounded by a rectangular box.

Ensure: $\Delta r > 0, \textit{Rectangular box boundary.}$ ▷ Positive radial bin size

procedure RDF($Particles, \Delta r$)
  $Bounds \leftarrow$ FindBoundary($Particles$) ▷ Determine the sample boundaries
  $V_{Sample} \leftarrow Bounds.L_x \cdot Bounds.L_y \cdot Bounds.L_z$
  $RhoMean \leftarrow \frac{Particles.size()}{V_{Sample}}$ ▷ Mean particle density
  $Rmax \leftarrow \frac{1}{2} \cdot \sqrt{Bounds.L_x^2 + Bounds.L_y^2 + Bounds.L_z^2}$ ▷ Maximal radial distance
  $BinNum \leftarrow \lfloor \frac{Rmax}{\Delta r} \rfloor$ ▷ Number of radial bins
  $r \leftarrow [\,]$ ▷ Empty list for the radial bins
  for $k = 0, k < BinNum$ **do**
    $r.append((k + 0.5) \cdot \Delta r)$ ▷ List with radial bin centers
    $k \leftarrow k + 1$
  end for
  $Gr \leftarrow [0...]$ ▷ Empty bins for $g(r)$
  $NonEmptyShells \leftarrow [0...]$ ▷ Keeps track of non empty shells
  for $CentralP \in Particles$ **do** ▷ Use every particle as the center
    $LocalGr \leftarrow [0...]$ ▷ Empty bins for the local $g(r)$
    for $Neighbor \in Particles \wedge Neighbor \neq CentralP$ **do**
      $D \leftarrow$ Distance($Neighbor, CentralP$) ▷ Inter particle distance
      $Idx \leftarrow \lfloor \frac{1}{\Delta r} \cdot D \rfloor$ ▷ Index into $LocalGr$
      $LocalGr[Idx] \leftarrow LocalGr[Idx] + 1$ ▷ Bin the particle
    end for
    $ShiftBounds \leftarrow$ ShiftCenter($Bounds, CentralP$) ▷ Shift the COSY center to $CentralP$
    for $k = 0, k < BinNum$ **do** ▷ Compute the local particle density
      $V_{shell} \leftarrow$ ShellVolume($r[k] - \frac{\Delta r}{2}, r[k] + \frac{\Delta r}{2}, ShiftBounds$)
      if $V_{shell} > 0$ **then**
        $LocalGr[k] \leftarrow LocalGr[k]/V_{shell}$ ▷ Use the shell-box intersection volume
        $NonEmptyShells[k] \leftarrow NonEmptyShells[k] + 1$
      end if ▷ Empty intersection volumes are not used
    end for
    $Gr \leftarrow Gr + LocalGr$ ▷ Add up all local $g(r)$ bins for averaging
  end for
  for $k = 0, k < BinNum$ **do**
    $Gr[k] \leftarrow Gr[k]/NonEmptyShells[k]$ ▷ Average over all non emtpy shells
  end for
  $Gr \leftarrow \frac{Gr}{RhoMean}$ ▷ Final normalization by the average particle density
  return r, Gr
end procedure

Algorithm 10 Compute $g(r)$ from a particle set bounded by a sphere.

Ensure: $\Delta r > 0, Rectangular\ box\ boundary.$ ▷ Positive radial bin size

procedure RDF($Particles, \Delta r$)
  $x_c \leftarrow$ MEAN($Particles.XPos$) ▷ Determine the boundary sphere center
  $y_c \leftarrow$ MEAN($Particles.YPos$)
  $z_c \leftarrow$ MEAN($Particles.ZPos$)
  $Radii \leftarrow$ DISTANCE($Particles, (x_{center}, y_{center}, z_{center})$) ▷ Radial particle distances from the center
  $R_{boundary} \leftarrow$ MAX($Radii$) ▷ Sphere surrounding all particles
  $V_{Sample} \leftarrow \frac{4}{3} \cdot \pi \cdot R^3_{boundary}$
  $RhoMean \leftarrow \frac{Particles.size()}{V_{Sample}}$ ▷ Mean particle density
  $BinNum \leftarrow \lfloor \frac{R_{boundary}}{\Delta r} \rfloor$ ▷ Number of radial bins
  $r \leftarrow [\,]$ ▷ Empty list for the radial bins
  for $k = 0, k < BinNum$ **do**
    $r.append((k + 0.5) \cdot \Delta r)$ ▷ List with radial bin centers
    $k \leftarrow k + 1$
  end for
  $Gr \leftarrow [0...]$ ▷ Empty bins for $g(r)$
  for $CentralP \in Particles$ **do** ▷ Use every particle as the center
    $LocalGr \leftarrow [0...]$ ▷ Empty bins for the local $g(r)$
    for $Neighbor \in Particles \wedge Neighbor \neq CentralP$ **do**
      $dx \leftarrow Neighbor.XPos - CentralP.XPos$ ▷ Inter particle distance
      $dy \leftarrow Neighbor.YPos - CentralP.YPos$
      $dz \leftarrow Neighbor.ZPos - CentralP.ZPos$
      $Idx \leftarrow \lfloor \frac{1}{\Delta r} \cdot \sqrt{dx^2 + dy^2 + dz^2} \rfloor$ ▷ Index into $LocalGr$
      $LocalGr[Idx] \leftarrow LocalGr[Idx] + 1$ ▷ Bin the particle
    end for
    for $k = 0, k < BinNum$ **do** ▷ Compute the local particle density
      $V_{shell} \leftarrow$ SHELLSPHEREVOLUME($r[k] - \frac{\Delta r}{2}, r[k] + \frac{\Delta r}{2}, R_{boundary}, x_c, y_c, z_c, CentralP$)
      if $V_{shell} > 0$ **then**
        $LocalGr[k] \leftarrow LocalGr[k]/V_{shell}$ ▷ Use the shell-box intersection volume
      end if
    end for
    $Gr \leftarrow Gr + LocalGr$ ▷ Add up all local $g(r)$ bins for averaging
  end for
  $Gr \leftarrow \frac{Gr}{Particles.size()}$ ▷ Average over all particles
  $Gr \leftarrow \frac{Gr}{RhoMean}$ ▷ Final normalization by the average particle density
  return r, Gr
end procedure

11.10 Source code

11.10.1 FCC Coordinate Generation

```
from numpy import sqrt
from random import random
from numpy.random import normal
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D

"""
This program calculates FCC coordinates.
A 3D Gaussian disturbance can be added.
Random Sphere Point Picking Algorithm from:
G. Marsaglia, The Annals of Mathematical Statistics,
1972, Vol. 43, No. 2, 845-646
"""

### DASHBOARD
SimuName = "FCC_Blurr_Cube"  # Name of the Experiment and the Files
GitterKonst = sqrt(2)        # Cell constant for the FCC unit cell
PartDiam = 1                 # Diameter of the Particles
Blurr = True                 # add a 3D gaussian blurr?
sigma = 0.07                 # standard deviation of the blurr
Lx = 4                       # Distance in x-direction from -Lx to Lx
Ly = 4                       # Distance in y-direction from -Ly to Ly
Lz = 4                       # Distance in z-direction from -Lz to Lz

def calcFCCCoor(GitterKonst, Lx, Ly, Lz):

    CoorList = []
    XLim = int(Lx/GitterKonst*5)
    YLim = int(Ly/GitterKonst*5)
    ZLim = int(Lz/GitterKonst*5)
    L = GitterKonst/2
    A = [0, L, L]
    B = [L, 0, L]
    C = [L, L, 0]
    for h in range(-ZLim*2, ZLim*2, 1):
        for k in range(-YLim*2, YLim*2, 1):
            for l in range(-XLim*2, XLim*2, 1):
                NewX = h * A[0] + k * B[0] + l * C[0]
                NewY = h * A[1] + k * B[1] + l * C[1]
                NewZ = h * A[2] + k * B[2] + l * C[2]
                if abs(NewX) < Lx and\
                   abs(NewY) < Ly and\
                   abs(NewZ) < Lz:

                    CoorList.append([NewX, NewY, NewZ])

    return CoorList

def getRandOffset(sigma):

    x1 = 2
    x2 = 2
    while x1**2 + x2**2  >= 1:
        x1 = random()*2-1
        x2 = random()*2-1
    R = normal(0, sigma)
    Wurzel = sqrt(1 - x1**2 - x2**2)
    x = 2 * x1 * Wurzel
    y = 2 * x2 * Wurzel
    z = 1 - 2 * (x1**2 + x2**2)
```

```python

    return [x * R, y * R, z * R]

def addGaussianBlurr(CoorList, sigma):

    BlurrList = []
    for P in CoorList:
        Offset = getRandOffset(sigma)
        NewX = P[0] + Offset[0]
        NewY = P[1] + Offset[1]
        NewZ = P[2] + Offset[2]
        BlurrList.append([NewX, NewY, NewZ])

    return BlurrList

def plotCorr(CoorList, SimuName):

    X = [0] * len(CoorList)
    Y = [0] * len(CoorList)
    Z = [0] * len(CoorList)
    for P in range(0, len(CoorList), 1):
        X[P] = CoorList[P][0]
        Y[P] = CoorList[P][1]
        Z[P] = CoorList[P][2]

    fig = figure()
    ax = fig.add_subplot(111, projection = "3d", aspect = "equal")
    ax.scatter(X, Y, Z)
    ax.set_xlabel("X")
    ax.set_ylabel("Y")
    ax.set_zlabel("Z")
    savefig(SimuName + "_3D.png", dpi = 600)
    show()

    return True

def saveData(CoorList, SimuName):

    DataFile = open(SimuName + "_rawCoor.txt", "w")
    for Particle in CoorList:
        DataFile.write(str(Particle[0]) + "\t" +\
                       str(Particle[1]) + "\t" +\
                       str(Particle[2]) + "\n")
    DataFile.close()

    return True

def createAutoCADFile(CoorList, PartDiam, SimuName):

    CADFile = open(SimuName + "_CAD.scr", "w")
    for Particle in CoorList:
        CADFile.write("SPHERE\n")  # SPHERE command
        CADFile.write(str(round(Particle[0], 5)) + ",") # X-Position
        CADFile.write(str(round(Particle[1], 5)) + ",") # Y-Position
        CADFile.write(str(round(Particle[2], 5)) + "\n")# Z-Position
        CADFile.write(str(PartDiam/2) + "\n") # Radius

    CADFile.write("\n")  # Blank line at the end
    CADFile.close()

    return True

###MAIN
print("Calculating FCC Coordinates.")
FCC = calcFCCCoor(GitterKonst, Lx, Ly, Lz)
print("Particle Number: " + str(len(FCC)))
if Blurr == True:
    print("Adding a Gaussian blurr with Sigma = " + str(sigma))
```

```
    FCC = addGaussianBlurr(FCC, sigma)

print("Saveing the Coordinats to a .txt file.")
saveData(FCC, SimuName)
print("Creating a AutoCAD file.")
createAutoCADFile(FCC, PartDiam, SimuName)
print("Plotting the Coordinates.")
plotCorr(FCC, SimuName)
```

11.10.2 Simple $g(r)$ Implementation in Python 3

```
from scipy import spatial
from numpy import sqrt, pi, zeros

"""
This function calculates the pair correlation function g(2)(r)
for a dense set of particles in a rectengular box. The particle
coordinates are supplied as [[x, y, z],...] lists.
All particles are used as central particles.
The spherical shell might extend beyond the known region.
A KDTree data structure is used for efficiency.
"""

def RDF_Simple(Particles, r, dr):

    # preallocate list for Gr
    Gr = zeros(len(r))
    # maximal radial distance
    MaxDist = r[-1] + dr/2
    # sort all Particles in a KDTree
    ParticleTree = spatial.KDTree(Particles)
    # use every particle as the center once
    k = 0
    for CentralP in Particles:
        # these are the indices for the particles at most MaxDist away:
        NNIndices = ParticleTree.query_ball_point(CentralP,\
                                                  MaxDist, p = 2, eps = 0)
        # look at every other particle at most MaxDist away:
        for Neighbour in NNIndices:
            if CentralP != Particles[Neighbour]:
                # calc the distance to the neighbour
                dx = CentralP[0] - Particles[Neighbour][0]
                dy = CentralP[1] - Particles[Neighbour][1]
                dz = CentralP[2] - Particles[Neighbour][2]
                d = sqrt(dx**2 + dy**2 + dz**2)
                # what bins is the particle in?
                IdxList = [k for k in range(0, len(r), 1)\
                           if abs(r[k] - d) <= dr]
                # add one to every bin the particle is in
                for Pos in IdxList:
                    # count the particle
                    Gr[Pos] += 1
        k += 1
        print("Finished Particle " + str(k) + " of " + str(len(Particles)))
    # final normalization
    Gr[0] = 0
    BoxVol = (max(Particles[0]) - min(Particles[0]))*\
             (max(Particles[1]) - min(Particles[1]))*\
             (max(Particles[2]) - min(Particles[2]))
    for i in range(1, len(r), 1):
        Gr[i] /= len(Particles) # average over all central particles
        # the most inner shells are spheres!
        if r[i]- dr/2 > 0:
            # by the shell volume
            Gr[i] /= 4/3 *pi * ((r[i] + dr/2)**3 - (r[i]- dr/2)**3)
```

```
        else:
            Gr[i] /= 4/3 *pi * (r[i] + dr/2)**3 # by the shell volume
        Gr[i] *= BoxVol / len(Particles) # by the number density

    return Gr
```

11.10.3 Analytic $g(r)$ Implementation in Python 3

```
from numpy import sqrt, pi, zeros, ones, arctan
from random import random, seed
from time import time
seed(time())

"""
This function calculates the pair correlation function g(2)(r)
for a dense set of particles in a rectengular box. The particle
coordinates are supplied as [[x, y, z],...] lists.
All particles are used as central particles.
The spherical shell might extend beyond the known region.
An empty sourrounding is assumed.
"""

def SphereCutVol(Rs, A, B):

    Root = sqrt(Rs**2-A**2-B**2)
    Vcut = 1/6*Rs**3*(pi - 2*arctan(A*B/(Rs*Root)))
    Vcut += 1/2 * (arctan(A/Root) - pi/2)*(Rs**2*B-1/3*B**3)
    Vcut += 1/2 * (arctan(B/Root) - pi/2)*(Rs**2*A-1/3*A**3)
    Vcut += 1/3 * A * B * Root

    return Vcut

def OctVolume(Rs, xb, yb, zb):

    # if all boundaries are fully in the octant
    if xb**2 + yb**2 + zb**2 < Rs**2:
        return xb * yb * zb
    # if no boundary intersects we start with
    VOctant = 1/8 * 4/3 * pi * Rs**3
    # remove the spherical caps
    for B in [xb, yb, zb]:
        if B < Rs:
            VOctant -= pi/4*(2/3*Rs**3-B*Rs**2+1/3*B**3)
    # add the intersections of the caps
    for (a, b) in [(xb, yb), (xb, zb), (yb, zb)]:
        if a**2 + b**2 < Rs**2:
            VOctant += SphereCutVol(Rs, a, b)

    return VOctant

def SphereVolume(Rs, BoxBounds):

    [Xmin, Xmax, Ymin, Ymax, Zmin, Zmax] = BoxBounds
    VSphere = 0
    # abs() mirrors the boundaries into the first octant
    for xb in [Xmin, Xmax]:
        for yb in [Ymin, Ymax]:
            for zb in [Zmin, Zmax]:
                VSphere += OctVolume(Rs, abs(xb), abs(yb), abs(zb))

    return VSphere

def ShellVolume(Rmin, Rmax, BoxBounds):

    # check for negative Rmin values
```

```
    Rmin = max([Rmin, 0])
    InnerShell = SphereVolume(Rmin, BoxBounds)
    OuterShell = SphereVolume(Rmax, BoxBounds)
    Volume = OuterShell - InnerShell

    return Volume

def RDF_AnalyticNorm(Particles, r, dr):

    # Gr averaged over all particles
    Global_Gr = zeros(len(r))
    # Keep track of the usefull shell volumes
    NonEmptyShells = zeros(len(r))
    # maximal radial distance
    MaxDist = r[-1] + dr/2
    # Box boundaries
    XList = [Particles[k][0] for k in range(0, len(Particles), 1)]
    YList = [Particles[k][1] for k in range(0, len(Particles), 1)]
    ZList = [Particles[k][2] for k in range(0, len(Particles), 1)]
    BoxBounds = [min(XList), max(XList),\
                 min(YList), max(YList),\
                 min(ZList), max(ZList)]
    # box size
    Lx = BoxBounds[1] - BoxBounds[0]
    Ly = BoxBounds[3] - BoxBounds[2]
    Lz = BoxBounds[5] - BoxBounds[4]
    print("Lx = " + str(Lx))
    print("Ly = " + str(Ly))
    print("Lz = " + str(Lz))
    MeanDensity = len(Particles) / (Lx * Ly * Lz)
    # use every particle as the center once
    for CentralP in range(0, len(Particles), 1):
        # local Gr around the current particle
        Local_Gr = zeros(len(r))
        # look at every other particle at most MaxDist away:
        for Neighbour in range(0, len(Particles), 1):
            if CentralP != Neighbour:
                # calc the distance to the neighbour
                dx = Particles[CentralP][0] - Particles[Neighbour][0]
                dy = Particles[CentralP][1] - Particles[Neighbour][1]
                dz = Particles[CentralP][2] - Particles[Neighbour][2]
                d = sqrt(dx**2 + dy**2 + dz**2)
                # what bins is the particle in?
                IdxList = [k for k in range(0, len(r), 1) if abs(r[k] - d) <= dr/2]
                # add one to every bin the particle is in
                for Pos in IdxList:
                    # count the particle
                    Local_Gr[Pos] += 1
        # shift the center of box cosy
        LocalBox = [BoxBounds[0] - Particles[CentralP][0],\
                    BoxBounds[1] - Particles[CentralP][0],\
                    BoxBounds[2] - Particles[CentralP][1],\
                    BoxBounds[3] - Particles[CentralP][1],\
                    BoxBounds[4] - Particles[CentralP][2],\
                    BoxBounds[5] - Particles[CentralP][2]]
        # normalize with the shell volume
        for RIdx in range(0, len(r), 1):
            SVolume = ShellVolume(r[RIdx]-dr/2, r[RIdx]+dr/2, LocalBox)
            if SVolume > 0.0:
                Local_Gr[RIdx] /= SVolume
                NonEmptyShells[RIdx] += 1
        # normalize by the mean particle density
        Local_Gr = Local_Gr / MeanDensity
        # save in the global gr for the average over particles
        Global_Gr = Global_Gr + Local_Gr
        print("Finished Particle " + str(CentralP) + " of " + str(len(Particles)))

    # final normalization considering the non empty shell volumes
```

```
    for k in range(0, len(Global_Gr), 1):
        if NonEmptyShells[k] != 0:
            Global_Gr[k] /= NonEmptyShells[k]
        else:
            print("All Shells at R =  " + str(r[k]) + " are Empty!")

    return Global_Gr
```

11.10.4 Analytic - Internal Volume

```
def VInt_Analytic(Rs, xb, zb):

    S = sqrt(Rs**2-xb**2-zb**2)
    Vint = 1/3 * (xb * zb * S)
    Vint += 1/2 * arctan(xb/S)*(Rs**2*zb-(1/3)*zb**3)
    Vint += 1/2 * arctan(zb/S)*(Rs**2*xb-(1/3)*xb**3)
    Vint -= 1/3 * Rs**3 * arctan(xb*zb/(Rs * S))

    return Vint
```

11.10.5 Monte Carlo - Internal Volume

```
def VInt_MC(Rs, xb, zb, NPoints):

    InsidePoints = 0
    for k in range(0, NPoints, 1):
        # random point in the first (Rs,Rs,Rs) octant
        XRand = random() * Rs
        YRand = random() * Rs
        ZRand = random() * Rs
        if XRand**2 + YRand**2 + ZRand**2 < Rs**2 and\
           XRand < xb and ZRand < zb:
            InsidePoints += 1

    return InsidePoints / NPoints * Rs**3
```

11.10.6 Numeric Integration - Internal Volume

```
from numpy import sqrt, pi, arctan, linspace, array
from scipy.integrate import quad

def VInt_Num(Rs, xb, zb):

    def height(x, z, Rs):
        return sqrt(Rs**2 - x**2 - z**2)

    def VInt(zb, Rs):

        return quad(height, 0, xb, args = (zb, Rs))[0]
    Volume = quad(VInt, 0, zb, args = (Rs))[0]

    return Volume
```

11.10.7 Analytic - Spherical Cut Volume

```
def SphereCutVol(Rs, A, B):

    Root = sqrt(Rs**2-A**2-B**2)
    Vcut = 1/6*Rs**3*(pi - 2*arctan(A*B/(Rs*Root)))
    Vcut += 1/2 * (arctan(A/Root) - pi/2)*(Rs**2*B-1/3*B**3)
    Vcut += 1/2 * (arctan(B/Root) - pi/2)*(Rs**2*A-1/3*A**3)
    Vcut += 1/3 * A * B * Root

    return Vcut
```

11.10.8 The Octant Sum

```
def SphereVolume(Rs, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax):

    VSphere = 0
    # abs() mirrors the boundaries into the first octant
    for xb in [abs(Xmin), abs(Xmax)]:
        for yb in [abs(Ymin), abs(Ymax)]:
            for zb in [abs(Zmin), abs(Zmax)]:
                VSphere += OctVolume(Rs, xb, yb, zb)
    return VSphere
```

11.10.9 The Octant Volume

```
def OctVolume(Rs, xb, yb, zb):

    # if all boundaries intersect inside the octant
    if xb**2 + yb**2 + zb**2 < Rs**2:
        return xb * yb * zb
    # if they dont intersect we start with a full octant
    VOctant = 1/8 * 4/3 * pi * Rs**3
    # then,  we remove the spherical caps
    for B in [xb, yb, zb]:
        if B < Rs:
            VOctant -= pi/4*(2/3*Rs**3-B*Rs**2+1/3*B**3)
    # finally, we add the intersections of the caps
    for (a, b) in [(xb, yb), (xb, zb), (yb, zb)]:
        if a**2 + b**2 < Rs**2:
            VOctant += VSphereCut(Rs, a, b)
    return VOctant
```

11.10.10 Monte Carlo - Spherical Cut Volume

```
def SphereCutVol_MC(Rs, A, B, NPoints):

    InsidePoints = 0
    for k in range(0, NPoints, 1):
        # random point in the first (Rs,Rs,Rs) octant
        XRand = random() * Rs
        YRand = random() * Rs
        ZRand = random() * Rs
        if XRand**2 + YRand**2 + ZRand**2 < Rs**2 and\
           XRand > A and YRand > B:
            InsidePoints += 1
```

```

    return InsidePoints / NPoints * Rs**3
```

11.10.11 Monte Carlo - Sphere Volume

```
def SphereVolume_MC(Rs, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax, NPoints):

    InsidePoints = 0
    for k in range(0, NPoints, 1):
        # random point in the box
        XRand = random() * (Xmax-Xmin) + Xmin
        YRand = random() * (Ymax-Ymin) + Ymin
        ZRand = random() * (Zmax-Zmin) + Zmin
        # if the point is inside the sphere
        if XRand**2 + YRand**2 + ZRand**2 < Rs**2:
            InsidePoints += 1

    return InsidePoints / NPoints*(Xmax-Xmin)*(Ymax-Ymin)*(Zmax-Zmin)
```

11.10.12 Simple $g(r)$ in C++

```
#include <iostream>
#include <sstream>
#include <chrono>
#include <cmath>
#include <cstdlib>
#include <string>
#include <fstream>
#include <istream>
#include <vector>
#include <streambuf>

long double pi = 3.14159265358979323846264 33;

struct Box
{
    long double Xmin = 0;
    long double Xmax = 0;
    long double Ymin = 0;
    long double Ymax = 0;
    long double Zmin = 0;
    long double Zmax = 0;
    long int PartNum = 0;
};

struct Particle
{
    long double XPos = 0;
    long double YPos = 0;
    long double ZPos = 0;
};

std::vector<std::string> split(const std::string& s, char delimiter)
{
   std::vector<std::string> tokens;
   std::string token;
   std::istringstream tokenStream(s);
   while (std::getline(tokenStream, token, delimiter))
   {
      tokens.push_back(token);
   }
```

```
    return tokens;
};

inline std::vector<Particle> readPartCoor(const std::string FileName)
{
    // open the raw data file
    std::ifstream DataFile(FileName + ".txt");
    std::string TxtData;
    // get length of file:
    DataFile.seekg(0, std::ios::end);   // go to the end of the file
    TxtData.reserve(DataFile.tellg());  // allocate enough space
    DataFile.seekg(0, std::ios::beg);   // go to the beginning of the file
    TxtData.assign((std::istreambuf_iterator<char>(DataFile)),
                   std::istreambuf_iterator<char>());
    DataFile.close();
    // split the data into lines
    std::vector<std::string> datalines = split(TxtData, '\n');
    Particle P;  // single particle struct
    std::vector<Particle> Particles; // list of particles
    std::cout << "The first lines in the raw data file are:" << std::endl;
    std::cout << datalines.at(0) << std::endl;
    std::cout << datalines.at(1) << std::endl;
    std::cout << datalines.at(2) << std::endl;
    std::cout << datalines.at(3) << std::endl;
    std::cout << "The RawDataFile has " << datalines.size() <<
    " lines." << std::endl;
    for(long int k = 0; k < datalines.size() ; k++)
    {
        std::vector<std::string> results;
        // split the individual lines (X, Y, Z)
        results = split(datalines.at(k), '\t');
        // save the single particle positions
        P.XPos = std::stold(results.at(0));
        P.YPos = std::stold(results.at(1));
        P.ZPos = std::stold(results.at(2));
        Particles.push_back(P);
    }
    return Particles;
};

inline Box FindBoxBounds(const std::vector<Particle> Particles)
{
    // Box boundaries
    Box BoxBounds;
    // initialize the boundaries with the first particle
    BoxBounds.Xmin = Particles.at(0).XPos;
    BoxBounds.Xmax = Particles.at(0).XPos;
    BoxBounds.Ymin = Particles.at(0).YPos;
    BoxBounds.Ymax = Particles.at(0).YPos;
    BoxBounds.Zmin = Particles.at(0).ZPos;
    BoxBounds.Zmax = Particles.at(0).ZPos;
    BoxBounds.PartNum = Particles.size();
    // loop over all other particles and find minima and maxima
    for(long int Pos = 1; Pos < Particles.size(); Pos++)
    {
        if(Particles.at(Pos).XPos < BoxBounds.Xmin)
            BoxBounds.Xmin = Particles.at(Pos).XPos;
        if(Particles.at(Pos).XPos > BoxBounds.Xmax)
            BoxBounds.Xmax = Particles.at(Pos).XPos;
        if(Particles.at(Pos).YPos < BoxBounds.Ymin)
            BoxBounds.Ymin = Particles.at(Pos).YPos;
        if(Particles.at(Pos).YPos > BoxBounds.Ymax)
            BoxBounds.Ymax = Particles.at(Pos).YPos;
        if(Particles.at(Pos).ZPos < BoxBounds.Zmin)
            BoxBounds.Zmin = Particles.at(Pos).ZPos;
        if(Particles.at(Pos).ZPos > BoxBounds.Zmax)
            BoxBounds.Zmax = Particles.at(Pos).ZPos;
    };
```

```
    return BoxBounds;
};

inline std::vector<long double> RDF_AnalyticNorm(
const std::vector<Particle> Particles,
const std::vector<long double> r,
const long double dr, const Box BoxBounds)
{
    // Gr averaged over all particles
    std::vector<long double> Global_Gr(r.size(), 0);
    // maximal radial distance
    long double MaxDist = r.at(r.size()-1) + dr/2.0;
    // box size
    long double Lx = BoxBounds.Xmax - BoxBounds.Xmin;
    long double Ly = BoxBounds.Ymax - BoxBounds.Ymin;
    long double Lz = BoxBounds.Zmax - BoxBounds.Zmin;
    // Show this to the user
    std::cout << "The Sample Volume has the following size:" << std::endl;
    std::cout << "Lx = " << Lx << std::endl;
    std::cout << "Ly = " << Ly << std::endl;
    std::cout << "Lz = " << Lz << std::endl;
    long double MeanDensity = Particles.size() / (Lx * Ly * Lz);
    // use every particle as the center once
    for(long int CentralP = 0; CentralP < Particles.size(); CentralP++)
    {
        // local Gr around the current particle
        std::vector<long double> Local_Gr(r.size(), 0);
        // look at every other particle at most MaxDist away:
        for(long int Neighbour = 0; Neighbour < Particles.size(); Neighbour++)
        {
            if(CentralP != Neighbour)
            {
                // calc the distance to the neighbour
                long double dx = Particles.at(CentralP).XPos -
                                 Particles.at(Neighbour).XPos;
                long double dy = Particles.at(CentralP).YPos -
                                 Particles.at(Neighbour).YPos;
                long double dz = Particles.at(CentralP).ZPos -
                                 Particles.at(Neighbour).ZPos;
                long double d = std::sqrt(dx*dx + dy*dy + dz*dz);
                // what bins is the particle in?
                for(long int RPos = 0; RPos < r.size(); RPos++)
                {
                    if(std::abs(r.at(RPos) - d) <= dr/2.0)
                    {
                        Local_Gr.at(RPos) += 1;
                    }
                }
            }
        }
        // local box for the shell box intersection volume
        Box LocalBox = BoxBounds;
        // shift the center of the box cosy
        LocalBox.Xmin = BoxBounds.Xmin - Particles.at(CentralP).XPos;
        LocalBox.Xmax = BoxBounds.Xmax - Particles.at(CentralP).XPos;
        LocalBox.Ymin = BoxBounds.Ymin - Particles.at(CentralP).YPos;
        LocalBox.Ymax = BoxBounds.Ymax - Particles.at(CentralP).YPos;
        LocalBox.Zmin = BoxBounds.Zmin - Particles.at(CentralP).ZPos;
        LocalBox.Zmax = BoxBounds.Zmax - Particles.at(CentralP).ZPos;
        // normalize with the shell volume
        for(long int RIdx = 0; RIdx < r.size(); RIdx++)
        {
            // calculate the shell box intersection volume
            long double SVolume = 4.0/3.0 * pi *
            (std::pow(r.at(RIdx)+dr/2.0, 3) - std::pow(r.at(RIdx)-dr/2.0, 3));
            Local_Gr.at(RIdx) /= SVolume;
            // normalize by the mean particle density
            Local_Gr.at(RIdx) /= MeanDensity;
```

```
            // save in the global gr for the average over particles
            Global_Gr.at(RIdx) += Local_Gr.at(RIdx);
        }

        if(CentralP % 1000 == 0)
            std::cout << "Finished Particle " << CentralP <<
             " of " << Particles.size() << std::endl;
    }
    // final average over all particles
    for(long int k = 0; k < Global_Gr.size(); k++)
    {
        Global_Gr.at(k) /= Particles.size();
    }

    return Global_Gr;
};

void SaveData(const std::string SimuName, const std::string FileName,
              const std::vector<long double> Gr,
              const std::vector<long double> r,
              const long double dr, const long double ComputeTime,
              Box BoxBounds)
{
    // box size
    long double Lx = BoxBounds.Xmax - BoxBounds.Xmin;
    long double Ly = BoxBounds.Ymax - BoxBounds.Ymin;
    long double Lz = BoxBounds.Zmax - BoxBounds.Zmin;
    long double RmaxNatural = std::sqrt(Lx*Lx + Ly*Ly + Lz*Lz)/2.0;
    std::ofstream DataFile;
    DataFile.open(SimuName + ".txt", std::ios::out);
    DataFile.precision(25);
    DataFile << "Results for the radial distribution function computation." <<
    std::endl;
    DataFile << "Particle File Name: " << FileName << std::endl;
    DataFile << "Computation time for g(r): " << ComputeTime <<
    " nanoseconds." << std::endl;
    DataFile << "Radial Bin Width dr: " << dr << std::endl;
    DataFile << "Number of radial Bins: " << r.size() << std::endl;
    DataFile << "Number of Particles in the sample: " <<
    BoxBounds.PartNum << std::endl;
    DataFile << "Average Particle Number Density: " <<
    BoxBounds.PartNum / (Lx * Ly * Lz) << std::endl;
    DataFile << "Maximal useful radial distance: " << RmaxNatural << std::endl;
    DataFile << "Box Boundary Xmin: " << BoxBounds.Xmin << std::endl;
    DataFile << "Box Boundary Xmax: " << BoxBounds.Xmax << std::endl;
    DataFile << "Box Boundary Ymin: " << BoxBounds.Ymin << std::endl;
    DataFile << "Box Boundary Ymax: " << BoxBounds.Ymax << std::endl;
    DataFile << "Box Boundary Zmin: " << BoxBounds.Zmin << std::endl;
    DataFile << "Box Boundary Zmax: " << BoxBounds.Zmax << std::endl;
    DataFile << "Box Boundary X-length: " << Lx << std::endl;
    DataFile << "Box Boundary Y-length: " << Ly << std::endl;
    DataFile << "Box Boundary Z-length: " << Lz << std::endl;
    DataFile << "Box Volume: " << Lx * Ly * Lz << std::endl;
    DataFile << "# r \t g(r)" << std::endl;
    for(long int l = 0; l < r.size(); l++)
    {
        DataFile << r.at(l) << "\t";
        DataFile << Gr.at(l) << std::endl;
    }
    DataFile.close();
};

int main(void)
{
    // Dashboard
    std::string SimuName = "";     // Name of the simulation
    std::string FileName = "";     // Name of the simulation
    long int Bins = 1;                      // number of sample calculations
```

```
    long double dr = 1;                    // bin size
    // User Input
    std::cout << "Please enter the following information:" << std::endl;
    std::cout << "Name for the simulation: ";
    std::cin >> SimuName;
    std::cout << "Particle File Name: ";
    std::cin >> FileName;
    std::cout << "Bin size (dr): ";
    std::cin >> dr;
    std::cout << "Starting g(r) computation." << std::endl;
    // output precision
    std::cout.precision(20);
    // read in the raw particle coordinates
    std::vector<Particle> Particles = readPartCoor(FileName);
    // benchmark the g(r) calculation
    auto start = std::chrono::steady_clock::now();
    // find the Box Boundaries
    Box BoxBounds = FindBoxBounds(Particles);
    // box size
    long double Lx = BoxBounds.Xmax - BoxBounds.Xmin;
    long double Ly = BoxBounds.Ymax - BoxBounds.Ymin;
    long double Lz = BoxBounds.Zmax - BoxBounds.Zmin;
    // maximal useful radial distance
    long double RMAX = std::sqrt(Lx*Lx + Ly*Ly + Lz*Lz)/2.0;
    // show the maximal usefull radius to the user
    std::cout << "Maximal usefull radius: " << RMAX << std::endl;
    // Number of Bins
    Bins = static_cast<long int>(RMAX / dr);
    // the radial position and g(r) vector
    std::vector<long double> r(Bins, 0);
    std::vector<long double> Gr(Bins, 0);
    // create the radial distance vector
    for(long int k = 0; k < Bins; k++)
    {
        r.at(k) = k * RMAX/Bins;
    }
    // compute the radial distribution function
    Gr = RDF_AnalyticNorm(Particles, r, dr, BoxBounds);
    auto end = std::chrono::steady_clock::now();
    auto diff = end - start;
    double ComputeTime = std::chrono::duration <double, std::nano> (diff).count();
    // information for the user
    std::cout << "Time for the g(r) computation: ";
    std::cout << ComputeTime;
    std::cout << " nanoseconds" << std::endl;
    // save all the data into a file
    SaveData(SimuName, FileName, Gr, r, dr, ComputeTime, BoxBounds);
    std::cout << "Program Finished!" << std::endl;
};
```

11.10.13 Analytic $g(r)$ in C++

```
#include <iostream>
#include <sstream>
#include <chrono>
#include <cmath>
#include <cstdlib>
#include <string>
#include <fstream>
#include <istream>
#include <vector>
#include <streambuf>

long double pi = 3.14159265358979323846264 33;

```

```
struct Box
{
    long double Xmin = 0;
    long double Xmax = 0;
    long double Ymin = 0;
    long double Ymax = 0;
    long double Zmin = 0;
    long double Zmax = 0;
    long int PartNum = 0;
};

struct Particle
{
    long double XPos = 0;
    long double YPos = 0;
    long double ZPos = 0;
};

std::vector<std::string> split(const std::string& s, char delimiter)
{
   std::vector<std::string> tokens;
   std::string token;
   std::istringstream tokenStream(s);
   while (std::getline(tokenStream, token, delimiter))
   {
      tokens.push_back(token);
   }
   return tokens;
};

inline std::vector<Particle> readPartCoor(const std::string FileName)
{
    // open the raw data file
    std::ifstream DataFile(FileName + ".txt");
    std::string TxtData;
    // get length of file:
    DataFile.seekg(0, std::ios::end);   // go to the end of the file
    TxtData.reserve(DataFile.tellg());  // allocate enough space
    DataFile.seekg(0, std::ios::beg);   // go to the beginning of the file
    TxtData.assign((std::istreambuf_iterator<char>(DataFile)),
                   std::istreambuf_iterator<char>());
    DataFile.close();
    // split the data into lines
    std::vector<std::string> datalines = split(TxtData, '\n');
    Particle P;  // single particle struct
    std::vector<Particle> Particles; // list of particles
    std::cout << "The first lines in the raw data file are:" << std::endl;
    std::cout << datalines.at(0) << std::endl;
    std::cout << datalines.at(1) << std::endl;
    std::cout << datalines.at(2) << std::endl;
    std::cout << datalines.at(3) << std::endl;
    std::cout << "The RawDataFile has " << datalines.size() <<
    " lines." << std::endl;
    for(long int k = 0; k < datalines.size() ; k++)
    {
        std::vector<std::string> results;
        // split the individual lines (X, Y, Z)
        results = split(datalines.at(k), '\t');
        // save the single particle positions
        P.XPos = std::stold(results.at(0));
        P.YPos = std::stold(results.at(1));
        P.ZPos = std::stold(results.at(2));
        Particles.push_back(P);
    }
    return Particles;
};

inline long double SphereCutVol(const long double Rs,
```

```
                                const long double A,
                                const long double B)
{
    long double Root = std::sqrt(Rs*Rs-A*A-B*B);
    long double Vcut = Rs*Rs*Rs*(pi - 2.0*std::atan(A*B/(Rs*Root))) / 6.0;
    Vcut +=  (std::atan(A/Root) - pi/2.0)*(Rs*Rs*B-1.0/3.0*B*B*B) / 2.0;
    Vcut += (std::atan(B/Root) - pi/2.0)*(Rs*Rs*A-1.0/3.0*A*A*A) / 2.0;
    Vcut +=  A * B * Root / 3.0;

    return Vcut;
};

inline long double OctVolume(const long double Rs,
                             const long double xb,
                             const long double yb,
                             const long double zb)
{
    // if all boundaries are fully in the octant
    if(xb*xb + yb*yb + zb*zb < Rs*Rs)
    {
        return xb * yb * zb;
    }
    // if no boundary intersects we start with a full octant
    long double VOctant = pi * Rs*Rs*Rs / 6.0;
    // remove the spherical caps
    if(xb < Rs)
    {
        VOctant -= pi/4.0*(2.0/3.0*Rs*Rs*Rs-xb*Rs*Rs+1.0/3.0*xb*xb*xb);
    }
    if(yb < Rs)
    {
        VOctant -= pi/4.0*(2.0/3.0*Rs*Rs*Rs-yb*Rs*Rs+1.0/3.0*yb*yb*yb);
    }
    if(zb < Rs)
    {
        VOctant -= pi/4.0*(2.0/3.0*Rs*Rs*Rs-zb*Rs*Rs+1.0/3.0*zb*zb*zb);
    }
    // add the intersections of the caps
    if((xb*xb + yb*yb) < (Rs*Rs))
    {
        VOctant += SphereCutVol(Rs, xb, yb);
    }
    if((xb*xb + zb*zb) < (Rs*Rs))
    {
        VOctant += SphereCutVol(Rs, xb, zb);
    }
    if((yb*yb + zb*zb) < (Rs*Rs))
    {
        VOctant += SphereCutVol(Rs, yb, zb);
    }
    return VOctant;
};

inline long double SphereVolume(const long double Rs, const Box BoxSize)
{
    long double VSphere = 0.0;
    // std::abs() mirrors the boundaries into the first octant
    VSphere += OctVolume(Rs, std::abs(BoxSize.Xmin),
    std::abs(BoxSize.Ymin), std::abs(BoxSize.Zmin));
    VSphere += OctVolume(Rs, std::abs(BoxSize.Xmin),
    std::abs(BoxSize.Ymin), std::abs(BoxSize.Zmax));
    VSphere += OctVolume(Rs, std::abs(BoxSize.Xmin),
    std::abs(BoxSize.Ymax), std::abs(BoxSize.Zmin));
    VSphere += OctVolume(Rs, std::abs(BoxSize.Xmin),
    std::abs(BoxSize.Ymax), std::abs(BoxSize.Zmax));
    VSphere += OctVolume(Rs, std::abs(BoxSize.Xmax),
    std::abs(BoxSize.Ymin), std::abs(BoxSize.Zmin));
    VSphere += OctVolume(Rs, std::abs(BoxSize.Xmax),
```

```
    std::abs(BoxSize.Ymin), std::abs(BoxSize.Zmax));
    VSphere += OctVolume(Rs, std::abs(BoxSize.Xmax),
    std::abs(BoxSize.Ymax), std::abs(BoxSize.Zmin));
    VSphere += OctVolume(Rs, std::abs(BoxSize.Xmax),
    std::abs(BoxSize.Ymax), std::abs(BoxSize.Zmax));

    return VSphere;
};

inline long double ShellVolume(const long double Rmin,
                               const long double Rmax,
                               const Box BoxSize)
{
    long double InnerSphere = SphereVolume(Rmin, BoxSize);
    long double OuterSphere = SphereVolume(Rmax, BoxSize);
    long double ShellVol = OuterSphere - InnerSphere;

    return ShellVol;
};

inline Box FindBoxBounds(const std::vector<Particle> Particles)
{
    // Box boundaries
    Box BoxBounds;
    // initialize the boundaries with the first particle
    BoxBounds.Xmin = Particles.at(0).XPos;
    BoxBounds.Xmax = Particles.at(0).XPos;
    BoxBounds.Ymin = Particles.at(0).YPos;
    BoxBounds.Ymax = Particles.at(0).YPos;
    BoxBounds.Zmin = Particles.at(0).ZPos;
    BoxBounds.Zmax = Particles.at(0).ZPos;
    BoxBounds.PartNum = Particles.size();
    // loop over all other particles and find minima and maxima
    for(long int Pos = 1; Pos < Particles.size(); Pos++)
    {
        if(Particles.at(Pos).XPos < BoxBounds.Xmin)
            BoxBounds.Xmin = Particles.at(Pos).XPos;
        if(Particles.at(Pos).XPos > BoxBounds.Xmax)
            BoxBounds.Xmax = Particles.at(Pos).XPos;
        if(Particles.at(Pos).YPos < BoxBounds.Ymin)
            BoxBounds.Ymin = Particles.at(Pos).YPos;
        if(Particles.at(Pos).YPos > BoxBounds.Ymax)
            BoxBounds.Ymax = Particles.at(Pos).YPos;
        if(Particles.at(Pos).ZPos < BoxBounds.Zmin)
            BoxBounds.Zmin = Particles.at(Pos).ZPos;
        if(Particles.at(Pos).ZPos > BoxBounds.Zmax)
            BoxBounds.Zmax = Particles.at(Pos).ZPos;
    };
    return BoxBounds;
};

inline std::vector<long double> RDF_AnalyticNorm(
const std::vector<Particle> Particles,
const std::vector<long double> r,
const long double dr, const Box BoxBounds)
{
    // Gr averaged over all particles
    std::vector<long double> Global_Gr(r.size(), 0);
    // Keep track of the usefull (non-empty) shell volumes
    std::vector<long double> NonEmptyShells(r.size(), 0);
    // maximal radial distance
    long double MaxDist = r.at(r.size()-1) + dr/2.0;
    // box size
    long double Lx = BoxBounds.Xmax - BoxBounds.Xmin;
    long double Ly = BoxBounds.Ymax - BoxBounds.Ymin;
    long double Lz = BoxBounds.Zmax - BoxBounds.Zmin;
    // Show this to the user
    std::cout << "The Sample Volume has the following size:" << std::endl;
```

```
    std::cout << "Lx = " << Lx << std::endl;
    std::cout << "Ly = " << Ly << std::endl;
    std::cout << "Lz = " << Lz << std::endl;
    long double MeanDensity = Particles.size() / (Lx * Ly * Lz);
    // use every particle as the center once
    for(long int CentralP = 0; CentralP < Particles.size(); CentralP++)
    {
        // local Gr around the current particle
        std::vector<long double> Local_Gr(r.size(), 0);
        // look at every other particle at most MaxDist away:
        for(long int Neighbour = 0; Neighbour < Particles.size(); Neighbour++)
        {
            if(CentralP != Neighbour)
            {
                // calc the distance to the neighbour
                long double dx = Particles.at(CentralP).XPos -
                                 Particles.at(Neighbour).XPos;
                long double dy = Particles.at(CentralP).YPos -
                                 Particles.at(Neighbour).YPos;
                long double dz = Particles.at(CentralP).ZPos -
                                 Particles.at(Neighbour).ZPos;
                long double d = std::sqrt(dx*dx + dy*dy + dz*dz);
                // what bins is the particle in?
                for(long int RPos = 0; RPos < r.size(); RPos++)
                {
                    if(std::abs(r.at(RPos) - d) <= dr/2.0)
                    {
                        Local_Gr.at(RPos) += 1;
                    }
                }
            }
        }
        // local box for the shell box intersection volume
        Box LocalBox = BoxBounds;
        // shift the center of the box cosy
        LocalBox.Xmin = BoxBounds.Xmin - Particles.at(CentralP).XPos;
        LocalBox.Xmax = BoxBounds.Xmax - Particles.at(CentralP).XPos;
        LocalBox.Ymin = BoxBounds.Ymin - Particles.at(CentralP).YPos;
        LocalBox.Ymax = BoxBounds.Ymax - Particles.at(CentralP).YPos;
        LocalBox.Zmin = BoxBounds.Zmin - Particles.at(CentralP).ZPos;
        LocalBox.Zmax = BoxBounds.Zmax - Particles.at(CentralP).ZPos;
        // normalize with the shell volume
        for(long int RIdx = 0; RIdx < r.size(); RIdx++)
        {
            // calculate the shell box intersection volume
            long double SVolume = ShellVolume(r.at(RIdx)-dr/2.0,
            r.at(RIdx)+dr/2.0, LocalBox);
            if(SVolume > 0.0)
            {
                Local_Gr.at(RIdx) /= SVolume;
                NonEmptyShells.at(RIdx) += 1;
            }
            // normalize by the mean particle density
            Local_Gr.at(RIdx) /= MeanDensity;
            // save in the global gr for the average over particles
            Global_Gr.at(RIdx) += Local_Gr.at(RIdx);
        }
        if(CentralP % 1000 == 0)
            std::cout << "Finished Particle " << CentralP << " of "
        << Particles.size() << std::endl;
    }
    // final normalization considering the non empty shell volumes
    for(long int k = 0; k < Global_Gr.size(); k++)
    {
        if(NonEmptyShells.at(k) != 0)
        {
            Global_Gr.at(k) /= NonEmptyShells.at(k);
        }
```

```
        else
        {
            std::cout << "All Shells at R =  " << r.at(k) <<
            " are Empty!" << std::endl;
        }
    }
    return Global_Gr;
};

void SaveData(const std::string SimuName, const std::string FileName,
              const std::vector<long double> Gr,
              const std::vector<long double> r,
              const long double dr,
              const long double ComputeTime,
              Box BoxBounds)
{
    // box size
    long double Lx = BoxBounds.Xmax - BoxBounds.Xmin;
    long double Ly = BoxBounds.Ymax - BoxBounds.Ymin;
    long double Lz = BoxBounds.Zmax - BoxBounds.Zmin;
    long double RmaxNatural = std::sqrt(Lx*Lx + Ly*Ly + Lz*Lz)/2.0;
    std::ofstream DataFile;
    DataFile.open(SimuName + ".txt", std::ios::out);
    DataFile.precision(25);
    DataFile << "Results for the radial distribution function computation." <<
    std::endl;
    DataFile << "Particle File Name: " << FileName << std::endl;
    DataFile << "Computation time for g(r): " << ComputeTime <<
    " nanoseconds." << std::endl;
    DataFile << "Radial Bin Width dr: " << dr << std::endl;
    DataFile << "Number of radial Bins: " << r.size() << std::endl;
    DataFile << "Number of Particles in the sample: " <<
    BoxBounds.PartNum << std::endl;
    DataFile << "Average Particle Number Density: " <<
    BoxBounds.PartNum / (Lx * Ly * Lz) << std::endl;
    DataFile << "Maximal useful radial distance: " <<
    RmaxNatural << std::endl;
    DataFile << "Box Boundary Xmin: " << BoxBounds.Xmin << std::endl;
    DataFile << "Box Boundary Xmax: " << BoxBounds.Xmax << std::endl;
    DataFile << "Box Boundary Ymin: " << BoxBounds.Ymin << std::endl;
    DataFile << "Box Boundary Ymax: " << BoxBounds.Ymax << std::endl;
    DataFile << "Box Boundary Zmin: " << BoxBounds.Zmin << std::endl;
    DataFile << "Box Boundary Zmax: " << BoxBounds.Zmax << std::endl;
    DataFile << "Box Boundary X-length: " << Lx << std::endl;
    DataFile << "Box Boundary Y-length: " << Ly << std::endl;
    DataFile << "Box Boundary Z-length: " << Lz << std::endl;
    DataFile << "Box Volume: " << Lx * Ly * Lz << std::endl;
    DataFile << "# r \t g(r)" << std::endl;
    for(long int l = 0; l < r.size(); l++)
    {
        DataFile << r.at(l) << "\t";
        DataFile << Gr.at(l) << std::endl;
    }
    DataFile.close();
};

int main(void)
{
    // Dashboard
    std::string SimuName = "";     // Name of the simulation
    std::string FileName = "";     // Name of the simulation
    long int Bins = 1;             // number of sample calculations
    long double dr = 1;            // bin size
    // User Input
    std::cout << "Please enter the following information:" << std::endl;
    std::cout << "Name for the simulation: ";
    std::cin >> SimuName;
    std::cout << "Particle File Name: ";
```

```
    std::cin >> FileName;
    std::cout << "Bin size (dr): ";
    std::cin >> dr;
    std::cout << "Starting g(r) computation." << std::endl;
    // output precision
    std::cout.precision(20);
    // read in the raw particle coordinates
    std::vector<Particle> Particles = readPartCoor(FileName);
    // benchmark the g(r) calculation
    auto start = std::chrono::steady_clock::now();
    // find the Box Boundaries
    Box BoxBounds = FindBoxBounds(Particles);
    // box size
    long double Lx = BoxBounds.Xmax - BoxBounds.Xmin;
    long double Ly = BoxBounds.Ymax - BoxBounds.Ymin;
    long double Lz = BoxBounds.Zmax - BoxBounds.Zmin;
    // maximal useful radial distance
    long double RMAX = std::sqrt(Lx*Lx + Ly*Ly + Lz*Lz)/2.0;
    // show the maximal usefull radius to the user
    std::cout << "Maximal usefull radius: " << RMAX << std::endl;
    // Number of Bins
    Bins = static_cast<long int>(RMAX / dr);
    // the radial position and g(r) vector
    std::vector<long double> r(Bins, 0);
    std::vector<long double> Gr(Bins, 0);
    // create the radial distance vector
    for(long int k = 0; k < Bins; k++)
    {
        r.at(k) = k * RMAX/Bins;
    }
    // compute the radial distribution function
    Gr = RDF_AnalyticNorm(Particles, r, dr, BoxBounds);
    auto end = std::chrono::steady_clock::now();
    auto diff = end - start;
    double ComputeTime = std::chrono::duration <double, std::nano> (diff).count();
    // information for the user
    std::cout << "Time for the g(r) computation: ";
    std::cout << ComputeTime;
    std::cout << " nanoseconds" << std::endl;
    // save all the data into a file
    SaveData(SimuName, FileName, Gr, r, dr, ComputeTime, BoxBounds);
    std::cout << "Program Finished!" << std::endl;
};
```

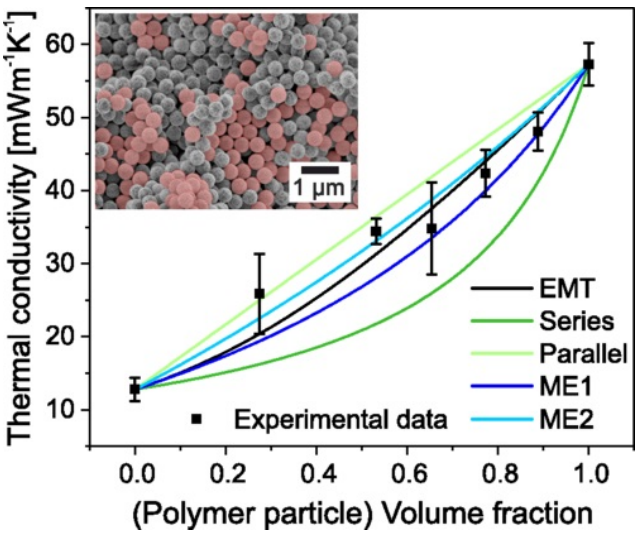
12

Thermal transport in binary colloidal glasses: Composition dependence and percolation assessment

Pia Ruckdeschel,[1] Alexandra Philipp,[1] Bernd A. F. Kopera,[1] Flora Bitterlich,[1] Martin Dulle,[2] Nelson W. Pech-May[1] and Markus Retsch[1]

[1] Department of Chemistry, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

[2] JCNS-1/ICS-1, Forschungszentrum Jülich, 52425 Jülich, Germany



Published in *Physical Review E*, **2018**, 97, 022612
Reproduced with permission from the American Physical Society

12.1 Abstract

The combination of various types of materials is often used to create superior composites that outperform the pure phase components. For any rational design, the thermal conductivity of the composite as a function of the volume fraction of the filler component needs to be known. When approaching the nanoscale, the homogeneous mixture of various components poses an additional challenge. Here, we investigate binary nanocomposite materials based on polymer latex beads and hollow silica nanoparticles. These form randomly mixed colloidal glasses on a sub-µm scale. We focus on the heat transport properties through such binary assembly structures. The thermal conductivity can be well described by the effective medium theory. However, film formation of the soft polymer component leads to phase segregation and a mismatch between existing mixing models. We confirm our experimental data by finite element modeling. This additionally allowed us to assess the onset of thermal transport percolation in such random particulate structures. Our study contributes to a better understanding of thermal transport through heterostructured particulate assemblies.

12.2 Introduction

Nanostructured materials have been the driving force for the tremendous development of many modern technologies, such as high-performance processors and data storage devices, sensors, or telecommunication equipment. (Opto) electronic devices have benefited from the ongoing miniaturization, which allowed device density to increase according to Moore's law. However, this miniaturization is also accompanied by an increased energy density and, consequently, by much higher, much more localized temperatures. Thus, it becomes increasingly important to control the way heat flows through these structures; one particular challenge is the dissipation of heat in electronic elements.[1] Therefore, electrically insulating heat spreading materials are used to prevent the formation of hot spots, which reduce the lifetime and reliability of electronic materials.[2],[3] These materials are commonly polymers filled with a highly thermally conductive micro- or nanomaterial.[4],[5] The thermal properties of these polymer composites depend mainly on the geometry of the filler (i.e., shape and size),[6] the volume fraction and distribution of the filler material in the matrix (lattice type),[7] the contact between the filler particles,[8] and the thermal properties of the components.[8] Advantages of polymer filled materials are easy processing and good mechanical properties.

For the application as a heat spreading material, a composite material needs to enhance the effective thermal conductivity greatly compared to that of its neat

matrix. The filler material is typically much more expensive and/or less available than the matrix component. Consequently, low loading fractions of the filler material are highly desirable. An important quantity to characterize the minimum amount of necessary filler is given by its percolation threshold. Percolation describes the formation of long-range connectivity in randomly ordered systems. Expectedly, the threshold shifts to lower filler volume fractions in composites by using small particles with a large aspect ratio (like rods or fibers),[9] a lattice type with a large number of neighbors (such as a face-centered cubic structure),[7] and a low contact resistance between the filler particles.[10]

Percolation theory analytically describes the composition dependence of a binary mixture of isotropic objects.[7] The relationship between the effective thermal conductivity of the composite κ_{eff} and the volume fraction of the filler V_{filler} is given by

$$\kappa_{\text{eff}} \propto \kappa_{\text{filler}} \left(V_{\text{filler}} - V_{\text{c}}\right)^n , \tag{12.1}$$

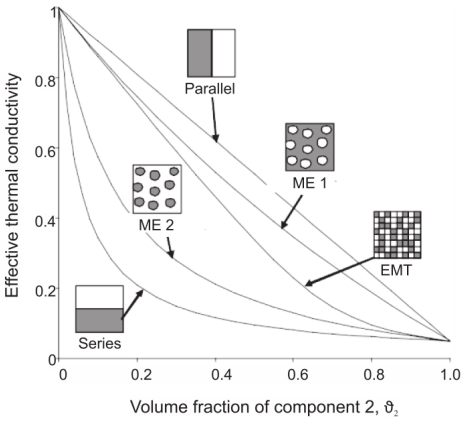
where κ_{filler} is the thermal conductivity of the filler, V_c the percolation threshold, and n the percolation exponent [typically 2 in a three-dimensional (3D) experiment]. Based on this equation, Mamunya *et al.*[11] developed an analytic expression for the electrical conductivity in composites, which can also be assigned to the effective thermal conductivity (κ_{eff})[8],

$$\kappa_{\text{eff}} = \left(\frac{\phi_1 - \phi_{\text{c}}}{F - \phi_c}\right)^t (\kappa_1 - \kappa_2) + \kappa_2, \tag{12.2}$$

where κ_1, κ_2 are the thermal conductivities of the two components 1 and 2 ($\kappa_1 \gg \kappa_2$), F is the maximum filling packing factor (here $F = 1$), ϕ_{c} is the percolation threshold ($\phi_{\text{c}} = \chi_{\text{c}} F$), which depends on the lattice structure (χ_{c} is the critical percolation probability, $\chi_{\text{c}} = 0.27$ for random close-packed structures),[12] ϕ_1 is the volume concentration of component 1, and t is a critical exponent which depends on the dimension of the structure (3D structures: $t = 2$).[7] However, the percolation theory cannot be applied to mixing ratios lower than the percolation threshold.

Therefore, the thermal conductivity of a composite material is often predicted by binary mixing models. These can be applied to the entire mixing phase diagram. The most common ones are the series and parallel model, the Maxwell-Eucken (ME) model 1 and 2 and the effective medium theory (EMT).[13],[14] These are summarized with their corresponding formulas in Fig. 12.1. The upper and lower bounds are given by the parallel and series model, respectively, in which the layers are aligned parallel or perpendicular to the heat flow direction. The ME model 1 and 2 assume separated inclusions in a dispersed, phase.[15] Thus, materials with percolating pathways cannot be described by the Maxwell-Eucken models. A better description is given by the EMT model. It considers a structure in which

two components are randomly distributed with neither phase being dispersed or continuous. Thus, continuous heat conduction pathways can be formed by either particle type depending on the respective volume fraction.



Series model	$\kappa_e = \frac{1}{\frac{1-\vartheta_2}{\kappa_1} + \frac{\vartheta_2}{\kappa_2}}$
Parallel model	$\kappa_e = (1-\vartheta_2)\kappa_1 + \vartheta_2\kappa_2$
Maxwell-Eucken 1	$\kappa_e = \kappa_1 \frac{2\kappa_1 + \kappa_2 - 2(\kappa_1 - \kappa_2)\vartheta_2}{2\kappa_1 + \kappa_2 + (\kappa_1 - \kappa_2)\vartheta_2}$
Maxwell-Eucken 2	$\kappa_e = \kappa_2 \frac{2\kappa_2 + \kappa_1 - 2(\kappa_2 - \kappa_1)(1-\vartheta_2)}{2\kappa_2 + \kappa_1 + (\kappa_2 - \kappa_1)(1-\vartheta_2)}$
Effective medium theory (EMT)	$\kappa_e = 1/4((3\vartheta_2 - 1)\kappa_2 + [3(1-\vartheta_2) - 1]\kappa_1 + \sqrt{[(3\vartheta_2 - 1)\kappa_1 + (3\{1-\vartheta_2\} - 1)\kappa_1]^2 + 8\kappa_1\kappa_2}$

Fig. 12.1.: Relative effective thermal conductivities κ_e for the series, the parallel, the Maxwell-Eucken (ME), and the effective medium theory (EMT) model and the corresponding equations. In the ME1 model, 1 is the continuous phase, whereas, in the ME2 model, 1 is the dispersed phase. Adapted with permission from Carson *et al.*[13]

Our contribution addresses two important aspects of composite materials. First, choosing a colloidal approach, we can fabricate nanostructured colloidal materials with any mixing ratio, which otherwise often suffer from macroscopic demixing or clustering. This allows us to investigate the applicability of the existing mixing models. Second, we shed light on thermal transport percolation in an isotropic particle ensemble. We compare our experimental results with finite element method (FEM) simulation, which provides a deeper understanding of the percolation evolution. We base our study on two types of electrically insulating building blocks: polymer latex beads and hollow silica nanoparticles. These are self-assembled into randomly close-packed colloidal ensembles.

12.3 Experimental Section

12.3.1 Fabrication of binary colloidal glasses

Two types of nanoparticles were synthesized: hollow silica spheres and poly(methyl methacrylate-*co*-*n*-butyl acrylate) [P(MMA-*co*-nBA)] particles. For the synthesis of hollow silica nanoparticles, first, monodisperse polystyrene particles were prepared by emulsifier-free emulsion polymerization.[16] Subsequently, the particles were coated with a thin silica shell by a modified Stoeber condensation process.[17] Finally,

the polystyrene core was removed by calcination. Monodisperse P(MMA-*co*-nBA) particles were synthesized by emulsifier free emulsion polymerization. Colloidal assemblies were prepared by a vacuum filtration system. Detailed information on the synthesis of the particles and the assembly process can be found in the Appendix.

12.3.2 Characterization methods

Scanning electron microscopy (SEM) was used to image the colloidal assemblies. Differential scanning calorimetry measurements were performed to determine the specific heat capacity. Thermal diffusivity of the colloidal assemblies was measured using a xenon flash apparatus. These measurements were done in vacuum at room temperature and in helium (1000 mbar) in the temperature range from 25 °C to 150 °C. The densities of the hollow sphere assemblies and the unmolten colloidal glasses were determined using a 3D digital surface profiler. The densities of the molten colloidal assemblies were measured using Archimedes' principle (buoyancy method). Further details about the characterization methods are summarized in the Appendix.

12.3.3 Finite element modeling

An amorphous particle assembly was obtained from molecular dynamics simulation. This particle structure was used for the finite element method (FEM) simulations of the effective thermal transport properties using COMSOL MULTIPHYSICS. A random selection process assigned the two types of particles to the colloidal glass. An exemplary structure used for FEM is shown in Fig. 12.2(c). For the simulation, a temperature difference was applied to opposite faces of the cube. This defines the temperature gradient across the colloidal structure and serves at the same time as a heat source and a heat sink, respectively. Thermal insulation boundary conditions are imposed on all other faces of the cube. The computed heat flux normal to the isothermal faces was used to calculate the effective thermal conductivity using Fourier's law. More details on the finite element modeling can be found in the Appendix.

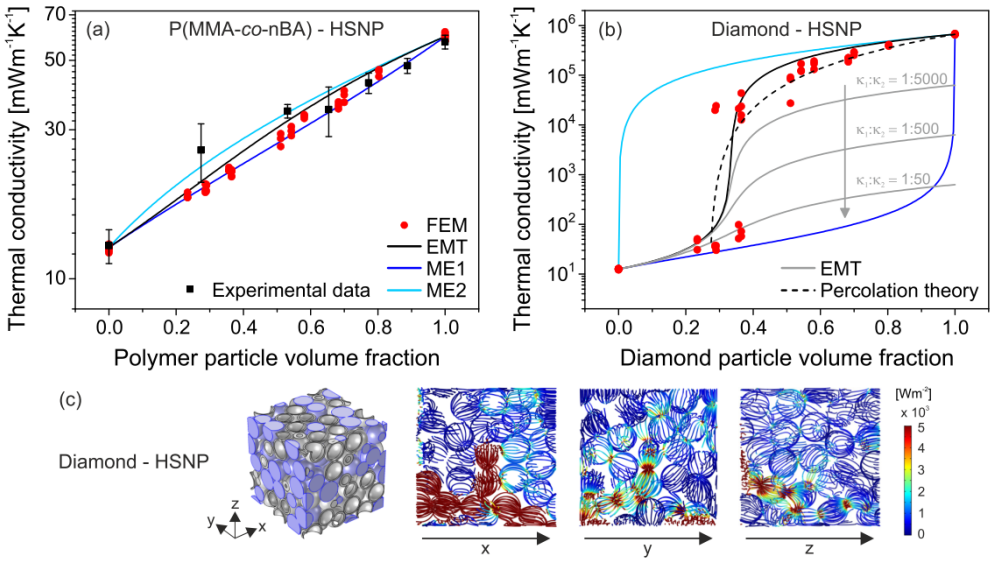


Fig. 12.2.: Finite element modeling (FEM) of binary colloidal assemblies with hollow silica spheres. (a) Thermal conductivity data of the binary particle mixtures, the EMT, ME1, and ME2 models in comparison to simulated data obtained from FEM. (b) Thermal conductivity data from FEM of binary colloidal glasses consisting of hollow spheres and diamond particles and the corresponding EMT (black line), ME1 (dark blue line), and ME2 (light blue line) models. The gray lines show the EMT model for binary particle mixtures with a decreasing thermal conductivity ratio (κ_1:κ_2) between the two types of particles. The black dashed line represents the fit by a percolation model [Eq.12.2]. (c) The geometry of the binary colloidal glass with a diamond particle volume fraction of 36.5% (blue particles) used for the FEM simulations and the corresponding heat flux streamlines for the different directions (x, y, and z).

12.4 Results and Discussion

12.4.1 Binary colloidal assemblies

The binary colloidal assemblies are built from two types of particles with comparable outer diameters and surface charges: copolymer particles P(MMA-*co*-nBA) with a diameter of 441 ± 9 nm (ζ potential of -43 mV) and hollow silica nanoparticles (HSNPs) with a diameter of 423 ± 8 nm, a shell thickness of 25.3 ± 1.2 nm (ζ potential of -27 mV). The electron microscopy images of these particles are shown in Fig. 12.3. Both images highlight the monodispersity of the particles. The polymeric particles feature a shrunk and rough surface due to the sensitivity of the PMMA to the electron beam.

We used vacuum filtration (see Fig. 12.4 and Appendix for more details) for the self-assembly process, since this method enables a fast and facile assembly of the particles into a colloidal glass with a random distribution of the two particle types.

Particularly, minor density differences do not influence the particle distribution in the colloidal glass using the filtration process. The number of polymer particles was varied from 100% to 0% particles (see Table 12.1). The comparable particle size and surface potential lead to a good particle distribution on the sub-µm scale in the composite. This is validated by side-view SEM images. The inset in Fig. 12.5 shows the 50/50 mixture with red-colored hollow spheres for easier identification. Additional SEM images of this 53P particle mixture demonstrate the homogeneous particle distribution throughout the entire colloidal glass (Fig. 12.6). Only on a local scale, small clusters comprising several particles of the same type can be seen. The other particle composites are shown in Fig. 12.7.

12.4.2 Thermal transport properties of binary colloidal glasses - Percolation effect

The thermal transport properties of the colloidal glasses were determined by xenon flash analysis. The method allows the direct determination of the thermal diffusivity α. The thermal conductivity κ can be calculated using the following equation:

$$\kappa = \alpha c_{\mathrm{p}} \rho. \tag{12.3}$$

The specific heat capacity c_{p} and density ρ data are plotted in Fig. 12.8.

Figure 12.5 shows the thermal conductivity data of the binary colloidal glasses for different polymer particle volume fractions. For a colloidal glass consisting of hollow silica spheres (0P), the thermal conductivity is extremely low with a value of only $13\,\mathrm{mWm^{-1}K^{-1}}$. The heat transport is limited in these colloidal glasses due to several structural properties on the nano- and mesoscale.[18],[19]

These are

(i) the hollow structure, which reduces the density to only $0.30\,\mathrm{gcm^{-3}}$,

(ii) the large amount of interfaces due to the particulate structure, which increases the thermal resistance by means of inter particle constriction, and

(iii) the low packing parameter of the particles to reduce the density and inter particle contact points even further.

Thus, silica hollow sphere colloidal glasses represent a highly insulating material class.

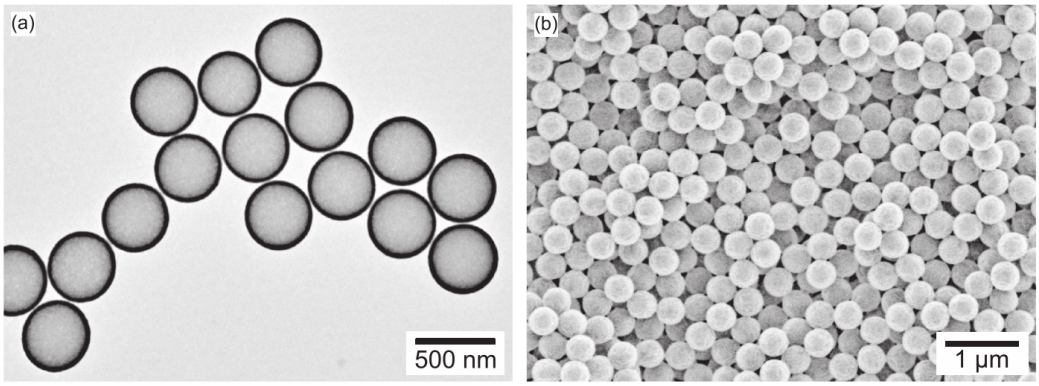


Fig. 12.3.: (a) Transmission electron microscopy images of the hollow silica nanoparticles. (b) Scanning electron microscopy images of the P(MMA-co-nBA) particles.

The thermal conductivity of the investigated P(MMA-*co*-nBA) particle colloidal glass is about $57\,\mathrm{mWm^{-1}K^{-1}}$ at a density of $0.77\,\mathrm{gcm^{-3}}$, which compares well to the value reported for 366 nm polystyrene colloidal crystals.[20] By mixing the two types of particles, the thermal conductivity can be tuned within these bounds. We find that the thermal conductivity increases monotonically with increasing polymer particle ratio within the colloidal assemblies. Thus, we do not find any sign of reduced thermal conductivity caused by mixing unlike particles, e.g., by additional interfacial resistances. We also cannot infer a distinct percolation threshold, which would be indicative for a preferred energy transport along a specific particle pathway.

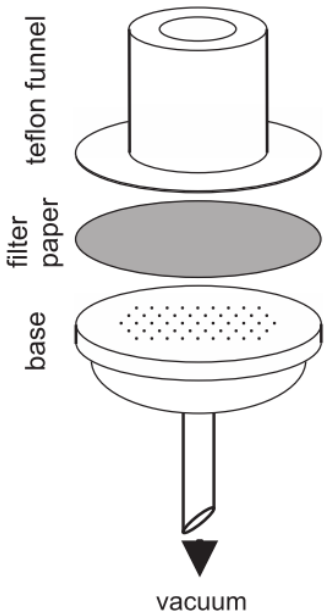


Fig. 12.4.: Schematic setup of the vacuum filtration system. The parts were held together by a spring clamp and put on a filter flask.

We compare now our binary colloidal alloys to the existing set of mixing models.[13] The EMT model describes our system across the whole volume fraction range reasonably well. At low and high volume fractions, the ME models are also in reasonable agreement. For these regions, the separated inclusion assumption of the ME model is valid since the minority particle phase is dispersed in the majority phase. Furthermore, the parallel and the series model are not adequate for our colloidal alloys. The standard deviation of the experimental data and the proximity of the highest and lowest thermal conductivity of the pure phase components does not allow for further discrimination between these models. We prefer the EMT model since it is conceptually applicable over the entire mixing range.

The finite element method is a useful tool to validate and understand the experimental data in greater detail. Figure 12.2(a) compares the experimental values (black squares) to the simulation results (red circles) based on a random close-packed particle structure with random binary particle selection. FEM allows the screening of various composition ratios easily. The resulting thermal conductivity data are plotted along with the EMT mixing model (black line), the ME1 (dark blue line), and the ME2 (light blue line) models. The data obtained from finite element modeling fits very well to the experimental data and are well described by the EMT and ME1 models. Thus, the FEM, EMT, and ME1 models are all suitable methods to predict the thermal conductivity of binary colloidal glasses (with comparable thermal conductivities of the two phases, which is discussed below). Furthermore, the assumptions implemented in the FEM simulation, such as diffusive thermal transport and no interfacial thermal contact resistance between adjacent spheres, describe the experimental situation adequately well.

Tab. 12.1.: Designation of the binary colloidal assemblies and the corresponding polymer particle volume fraction.

	100P	89P	77P	65P	53P	27P	0P
Number of polymer particles [%]	100	87.5	75	62.5	50	25	0
Polymer particle volume fraction	1	0.89	0.77	0.65	0.53	0.27	0

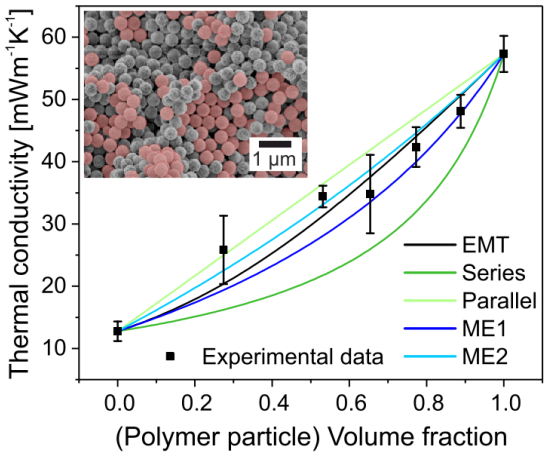


Fig. 12.5.: Thermal conductivity data of the binary colloidal glasses measured in vacuum at 25 °C (black squares) in comparison to different mixing models: effective medium theory (EMT), series model, parallel model, and Maxwell-Eucken (ME) model 1 and 2. The inset shows a side-view scanning electron microscopy (SEM) image of the colloidal assembly with a polymer particle volume fraction of 53% (53P). The hollow silica spheres are red-colored.

Based on this validation of the applicability of our FEM simulation, we can now use it to investigate the onset of percolation in binary colloidal crystals in greater detail. Therefore, we selected two types of dielectric particles which possess a large thermal conductivity contrast, which is experimentally hardly accessible: hollow silica nanospheres and diamond particles. The large thermal conductivity contrast should allow for a clear assessment of the percolation through the colloidal assembly. We want to emphasize that the structure of the colloidal glass remains unchanged since we only replace one type of particle species in the simulation volume. The FEM results in comparison to the EMT (black line), ME1 (dark blue line), and ME2 (light blue line) models are shown in Fig. 12.2(b). Additionally, we explore the effect of the thermal conductivity difference between the respective components and found that it should at least amount to κ_1:κ_2 = 1:500 in order to experimentally observe a step like increase in thermal conductivity [gray lines in Fig. 12.2(b)]. The formation of percolation pathways is found at a diamond particle volume fraction of about 30 %. In the range of this mixing ratio, a high degree of variability between the individual simulation points results. As outlined in the Appendix, each simulation was conducted along the x, y, and z axis of the simulation cube. In Fig. 12.2(c) heat flow streamlines are shown for the different orientations with a diamond particle volume fraction of 36.5%. While in y- and z direction the heat flow is relatively low (thermal conductivity of $\sim 60\,\mathrm{mWm^{-1}K^{-1}}$), the heat flow in the x direction shows a percolation pathway which increases the thermal conductivity to $\sim 24000\,\mathrm{mWm^{-1}K^{-1}}$). This large variability is indicative of a critical mixing ratio close to the percolation threshold. Owing to the hardware limitation of the size of our simulation cube, fully percolated pathways only span across distinct orientations

in this regime. Increasing the number of diamond particles ($> 50\%$) leads to a colloidal glass with robust percolating pathways in all directions. The effective medium theory describes this mechanism also quite well but overestimates the thermal conductivity at high diamond particle ratios. A major difference compared to the data shown in Fig. 12.2(a) (small differences in thermal conductivity between the two components) is obvious in Fig. 12.2(b). Neither of the ME models describes the FEM data adequately anymore. Both models strongly over- or underestimate the effective thermal conductivity, respectively. The explicit percolation model [Eq. 12.2], however, provides a complementary way to describe the simulated data. In contrast to the EMT model, the percolation theory depends on the underlying particle structure. In Eq. 12.2, κ_1 and κ_2 are the effective thermal conductivities of the pure hollow sphere or diamond particle colloidal assembly. Therefore, the maximum filling packing factor F equals 1. The percolation model is plotted in Fig. 12.2(b) as a black dashed line and shows a clear consistency with the simulated data above the percolation threshold at a mixing ratio of 27%. Owing to the random particle packing of our isotropic spheres, both models (EMT and percolation theory) predict a comparable percolation threshold. Furthermore, the percolation model does not overestimate the effective thermal conductivity of the particles mixture, as is the case for the EMT model.

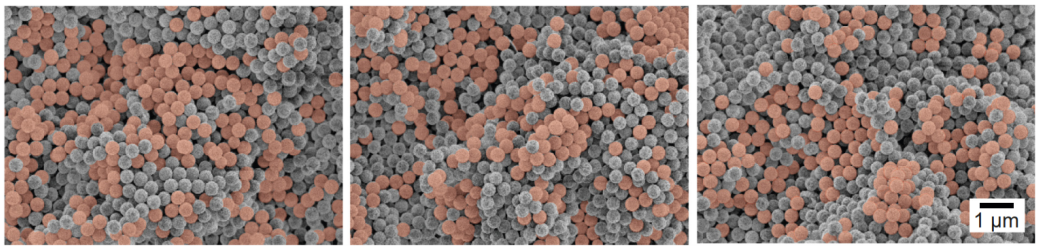


Fig. 12.6.: Scanning electron microscopy images of the colloidal assembly with a polymer particle volume fraction of 53% (53P) taken at different positions of the sample. The hollow silica spheres are red colored.

To sum up, the EMT model describes the effective thermal conductivity of binary particle mixtures across the entire mixing range reasonably well. The EMT model is also valid for particle mixtures which possess a significant difference between the individual thermal conductivities. However, the EMT model does not account for the explicit structure of the particle ensemble. The percolation theory, on the other hand, describes the onset of percolation equally well and is explicitly based on a randomly close-packed structure. The percolation theory provides a complementary possibility to describe the effective thermal conductivity beyond the percolation threshold. In contrast to the EMT model, a lower bound of the effective thermal conductivity is given by the percolation theory considering the right particle ordering. Our simulation results thereby validate the analytical solution.

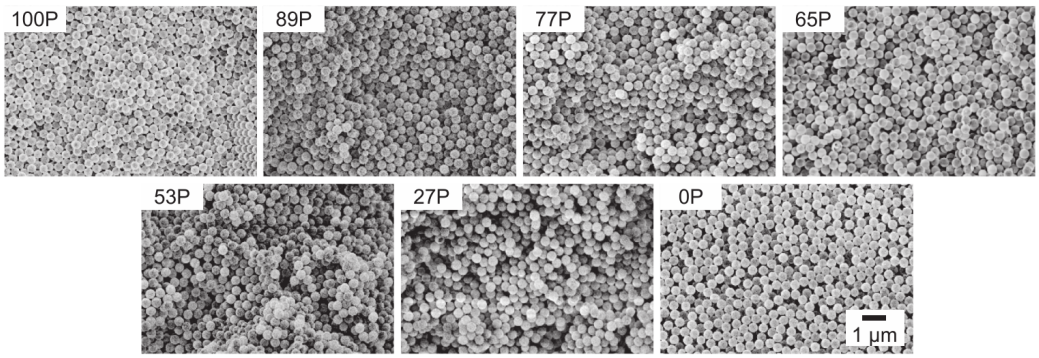


Fig. 12.7.: Scanning electron microscopy images of the binary colloidal glasses comprising polymer particle volume fractions between 100% and 0% before the heating process.

12.4.3 Thermal transport properties of binary colloidal glasses – Temperature dependency

We followed an additional route to investigate the heat transport through our binary colloidal mixture experimentally by exploiting the known temperature dependence of the thermal conductivity of polymer colloidal crystals.[20] Briefly, upon crossing the glass transition temperature (T_g) of the constituting polymer, the effective thermal conductivity of a pure polymer colloidal crystal increases by 200 %–300 %. We, therefore, measured the temperature-dependent thermal conductivity. To facilitate a homogeneous temperature distribution in the sample, measurements were performed in helium at 1000 mbar. The thermal conductivity of the different particle ratios is shown in Fig. 12.9(a). The thermal conductivity of the pure hollow silica nanoparticle colloidal glasses shows a monotonic growth driven by the specific heat capacity and no irreversible increase of the thermal conductivity. At the other extreme, the pure polymer particle glass exhibits the expected sharp step-up of the thermal conductivity after exceeding T_g due to the loss of the particulate structure [see SEM images in Fig. 12.9(d)].

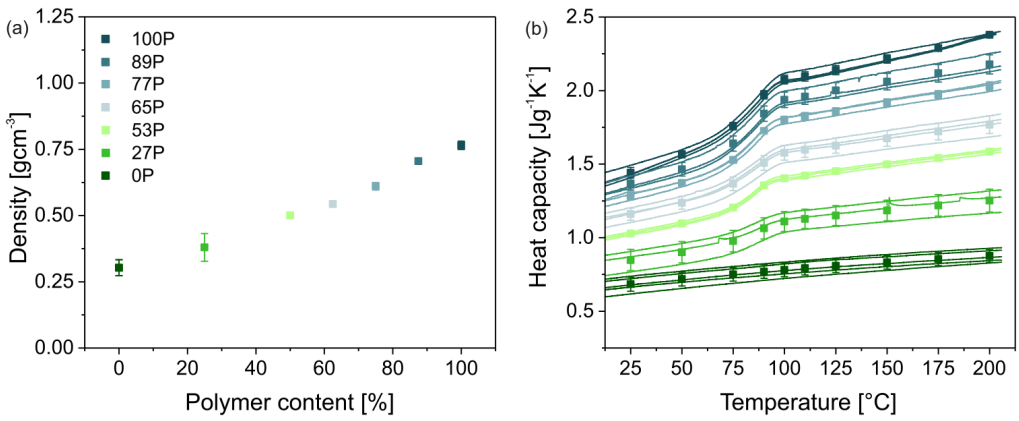


Fig. 12.8.: (a) The density of the different binary colloidal glasses. (b) Differential scanning calorimetry measurements of the particle mixtures.

The other particle mixtures feature a transient behavior between these two cases, however, with marked differences between the heating and cooling cycle. Figure 12.9(b) summarizes the differences before and after heating. In this case, any contribution from gaseous transport can be excluded since these data were measured in vacuum. While the thermal conductivity shows only minor increases of up to 50 % polymer content, the rise is distinct for higher ratios, where the polymer constitutes the majority phase. This can be attributed to the formation of a continuous polymer network due to the loss of interfaces. This is accompanied by an increase of the density shown in Fig. 12.9(c). Thus heat can be conducted faster through the molten polymer phase [compare SEM image in Fig. 12.9(d)]. It is important to differentiate this loss of structural integrity to the presence of percolating thermal transport pathways across a particular particle type, as discussed in the previous section. A percolating particle network should form at a polymer particle volume fraction of about 27 %; yet, the temperature induced film formation becomes significant only at polymer particle volume fractions $>$ 50 %. The structural integrity at elevated temperatures is compromised as soon as the polymer particles constitute the majority phase. The comparison of the molten film to the effective medium model [see Fig. 12.9(b)] reveals a second severe change to the homogeneity of the nanocomposite. The agreement between the EMT model and the measurement is much worse compared to the initial particle mixture. It now underestimates the thermal conductivity of high polymer mixing ratios and overestimates it at lower ratios. This hints towards a concomitant phase separation upon polymer annealing. The SEM images after the heating cycle qualitatively support this interpretation. Particularly for the case of low hollow sphere contents (89P and 77P), areas with large hollowsphere aggregates, even with a high degree of local order, can be inferred. These could originate from a relocation of the silica hollow spheres upon polymer film formation and confirm the challenge to fabricate homogeneous composite materials on the nanometer scale.

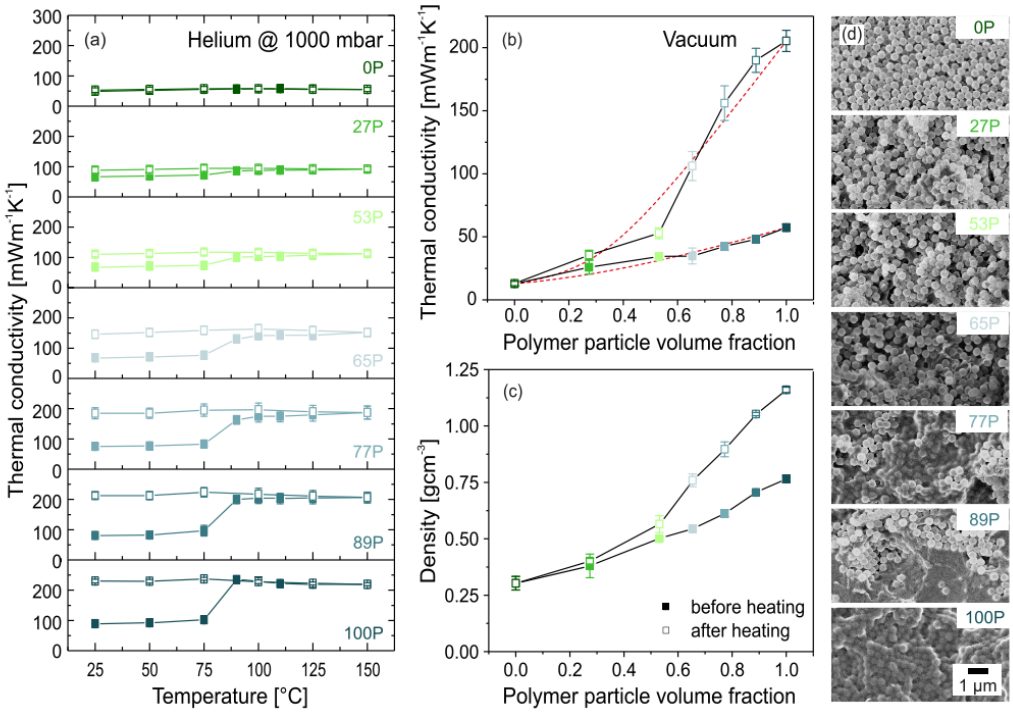


Fig. 12.9.: (a) Temperature-dependent thermal conductivity of colloidal glasses in helium at 1000 mbar. (b) Thermal conductivity of the particle assemblies in vacuum before and after heating to 150 °C. The red dashed line represents the EMT model.(c) Density of the colloidal assemblies before and after heating to 150 °C. (d) Scanning electron microscopy images of the colloidal assemblies after heating to 150 °C.

12.5 Conclusion

We investigated the thermal transport properties of binary colloidal glasses consisting of hollow silica nanoparticles and P(MMA-*co*-nBA) polymer particles. Mixing these two types of particles enables the fabrication of a nanocomposite material with a homogeneous distribution of the two components. The thermal conductivity increases from $13\,\mathrm{mWm^{-1}K^{-1}}$ to $57\,\mathrm{mWm^{-1}K^{-1}}$ in vacuum, depending on the particle mixture. Established mixing models are suitable to describe the effective thermal conductivity adequately. However, no percolation threshold can be observed for such similar particle mixtures (regarding their thermal conductivity). Using finite element modeling, we validated our experimental results and provided a better understanding of the percolation threshold. We find that the thermal conductivity ratio (κ_1:κ_2) between two types of particles must be at least 1:500 to observe a distinct step in the mixing ratio dependency. Furthermore, the Maxwell-Eucken mixing models fail to describe such nanocomposites with a large thermal conductivity contrast.

Heating the binary colloidal glasses above the glass transition temperature leads to a melting of the polymer and a loss of the interparticle interfaces. Thus, a continuous polymer network forms with solid particle inclusions for polymer contents $> 50\,\%$. This can be understood as a threshold of structural integrity. Concomitantly, the film formation induces a demixing of the hollow silica beads, resulting in a heterogeneous structure. This is less well described by the EMT mixing model. Our study contributes to a thorough understanding of percolation and effective thermal transport properties in particulate nanocomposite materials.

Acknowledgments

P.R. and B.A.F.K. acknowledge support from the Elite Network of Bavaria (ENB). Further support was provided by the SFB840 and the Bavarian Polymer Institute, BPI. This research was funded by the Volkswagen Foundation through a Lichtenberg Professorship.

P.R. and A.P. contributed equally to this work.

12.6 Appendix: Supporting Information

12.6.1 Materials

Styrene ($\geq$ 99%, Aldrich), 2,2'-azobis(2-methylpropionamidine) dihydrochloride (AIBA; 97%, Aldrich), [2-(methacryloxy)ethyl]trimethylammonium chloride (MTC; 70% solution in water, Polyscience), polyvinylpyrrolidone K30 (PVP, $M_\mathrm{w} \sim 55\,\mathrm{kgmol}^{-1}$, Aldrich), tetraethylorthosilicate (TEOS, $\geq$ 99%, Aldrich), ammonium hydroxide solution (30%–33% in water, Aldrich), and potassium peroxodisulfate (KPS; $\geq$ 99%, Aldrich) were used as received. Methyl methacrylate (MMA, 99%, Aldrich) and *n*-butyl acrylate (nBA, $\geq$ 99%, Aldrich) were purified by filtration over an alumina column (activated, basic, Brockmann I, Sigma-Aldrich). Ethanol was used in technical grade, and water was taken from a Millipore Direct Q3UV unit for all the synthesis and purification steps.

12.6.2 Synthesis of hollow silica nanoparticles

The synthesis of hollow silica spheres comprises three steps. (i) First, monodisperse polystyrene (PS) particles were prepared by emulsifier-free emulsion polymerization.[16] For the synthesis, 1.8 g PVP were dissolved in 10 mL water and added to a

500-mL three-necked flask equipped with a reflux condenser and a gas inlet. 225 mL water, 26 mL styrene, and 75 µL MTC were added. The emulsion was degassed and heated to the reaction temperature of 70 °C at a stirring speed of 850 rpm for 30 min. For the initiation of the polymerization, 0.6 g AIBA, dissolved in 5 mL water, were added. After the nucleation, the stirring speed was reduced to 450 rpm, and the reaction was carried out overnight under an argon atmosphere. (ii) In the second step, the polystyrene particles were coated with a thin silica shell by a modified Stoeber condensation process to obtain core-shell particles.[17] For the synthesis, 1.9 vol% TEOS were added to an 81.4 vol% ethanol, 10.7 vol% water, 6.0 vol% ammonium hydroxide solution containing 9.73 mg mL^{-1} PS latex particles. (iii) Subsequently, the polystyrene core was removed by calcining the particles at 500 °C for 12 h in air.

12.6.3 Synthesis of P(MMA-*co*-nBA) particles

The polymer particles were prepared by emulsifier-free emulsion polymerization. For the synthesis, 450 mL water and a mixture of 40 mL and 10 mL MMA and nBA were added to a 1000-mL three-necked flask, equipped with a reflux condenser and a gas inlet. The emulsion was degassed and heated to the reaction temperature of 75 °C at a stirring speed of 650 rpm. After 15 minutes, 2 mL of acrylic acid were added to the mixture followed by a further equilibration step for 5 min. The polymerization was initiated by a rapid injection of 150 mg KPS dissolved in 5 mL water. After the nucleation, the stirring speed was reduced to 400 rpm to avoid the formation of aggregates. The reaction was carried out overnight under an argon atmosphere. Residual reagents were removed by dialysis against water for five days, changing water twice a day.

12.6.4 Assembly into colloidal glasses

Colloidal assemblies were prepared by a vacuum filtration system from Merck Millipore (see Fig. 12.4). As a filter paper, an MF-Millipore membrane filter with a pore size of 0.2 µm was used. The sample size was reduced by a Teflon inset with a diameter of 2 cm. After the assembly, the samples were dried in vacuum overnight.

12.6.5 Characterization methods

SEM was performed using a Leo 1530 instrument (Zeiss) using acceleration voltages of 2 or 3 kV. The images were taken with an InLens and Everhart-Thornley secondary electron detector.

Differential scanning calorimetry (DSC) measurements were performed on a Discovery Q2500 DSC (TAInstruments). For the determination of the specific heat capacity the ASTM E1269 standard test method was used. Powders of the particles ($\sim$ 10 mg) were scanned in closed aluminum pans under a dry nitrogen flow of 50 ml min^{-1} over a temperature range from –40 °C to 210 °C with a heating rate of 20 °Cmin^{-1}. Two heating cycles were conducted, whereas only the second cycle was used for evaluation.

Thermal diffusivity measurements of the colloidal assemblies were performed on a xenon flash apparatus (Linseis, XFA500) equipped with an InSb infrared detector. Before the measurements, the samples were coated with a thin graphitic layer ($\sim$ 15 µm) on each side to ensure a good absorbance at the bottom and a high emissivity at the top side of the sample. The thickness of the graphite layer is negligible compared to the thickness of the colloidal glasses (700–1500 µm). The measurements were performed on at least three samples in vacuum ($\sim$ 0.05 mbar) at room temperature, and in helium (1000 mbar) in the temperature range from 25 °C to 150 °C. The received data were evaluated by the software APROSOFT LASER FLASH Evaluation v1.06 using the radiation fit model. The schematic setup and the corresponding measurement signal, fitted by the radiation model, are shown in Fig. 12.10. The thicknesses of the colloidal glasses were measured using a Litematic VL50 (Mitutoyo) measuring unit.

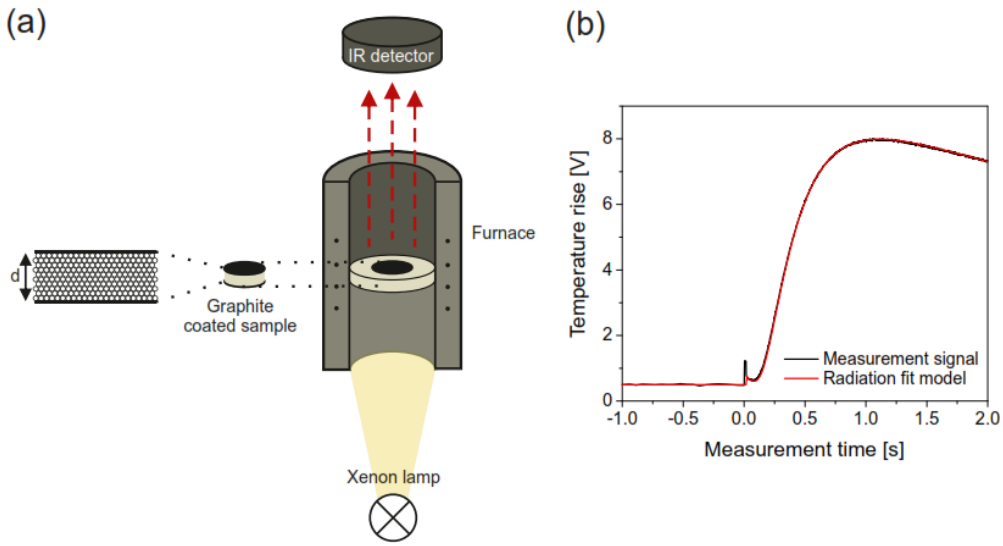


Fig. 12.10.: Schematic setup of the xenon flash analysis and the corresponding measurement signal with radiation fit. This fit model represents an extension to the finite-pulse and heat loss corrections given by the combined fit model from Dusza.[21] Moreover, it takes into account that part of the xenon flash is transmitted to the rear sample surface, leading to an instantaneous temperature jump analogous to Blumm *et al.*[22]

The densities of the hollow sphere assemblies and the unmolten colloidal glasses (0P–100P) were determined from the mass and volume of the monoliths. The volume was measured using a 3D digital surface profiler (Keyence V3100), and the mass was determined by weighing the colloidal glasses. The densities of the molten colloidal assemblies (27P–100P) were measured using Archimedes principle (buoyancy method). The colloidal glasses were hydrophobized by chemical vapor deposition using hexamethyldisilazane (HMDS) to avoid the adsorption of water by the hygroscopic silica network.

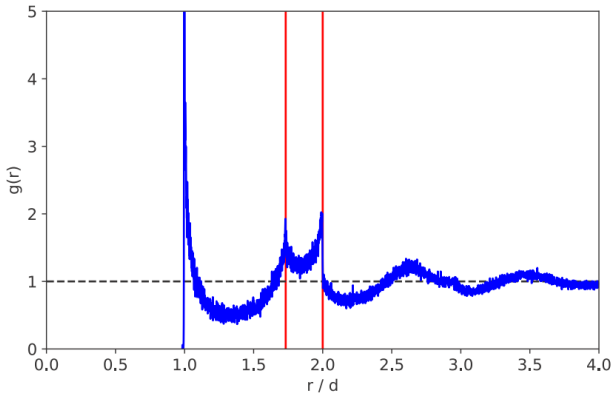


Fig. 12.11.: The radial distribution function of a random close-packed particle assembly from molecular dynamics simulation (program LAMPPS). The two red lines are at $\sqrt{3}$ and 2.

12.6.6 Finite element modeling

An amorphous structure of equally sized particles was compiled using a molecular dynamics simulation. Precisely, we used the gran-hooke pair potential[23]–[25] in LAMMPS on a set of 20,000 particles that were pulled by a gravitational force to the bottom of a simulation box with periodic boundaries in x and y direction. The $g(r)$ of the resulting structure (Fig. 12.11) shows the distinct features of a random sphere packing as described in the literature.[26]–[30]

This amorphous particle structure was used for the finite element method of the effective thermal transport properties using *COMSOL MULTIPHYSICS*. A random selection process assigned the two types of particles [hollow silica nanoparticles and P(MMA-*co*-nBA) particles] to the simulated colloidal glass, yielding a binary colloidal assembly with different polymer particle volume fractions. The particle size has been adjusted such that the mutual contact points between the spheres resemble the experiment. There, we find a relative contact area of about 0.6%–1% of the total particle surface area per contact point. The absolute value of this contact area is of minor relevance for the discussion, as it will only offset the overall thermal conductivity. For the hollow silica spheres, the experimental shell

thickness-to-radius ratio was used. Three cubes per mixture, each with around 100 particles, were cut out of the amorphous structure. The allowed number of particles in the cube is limited by the physical memory of the simulation computer (128 GB). An exemplary structure used for FEM is shown in Fig. 12.2(c). A temperature difference was applied to opposite faces of the cube, while all remaining surfaces were thermally insulating. Therefore, purely one dimensional thermal transport can occur. Furthermore, no thermal resistance between the particles was considered. The computed heat flux normal to the isothermal faces was used to calculate the effective thermal conductivity using Fourier's law [see Eq. 12.4],

$$\kappa_{\text{eff}} = \frac{Q/A}{\Delta T/h}, \tag{12.4}$$

where Q is the heat flow rate normal to the isothermal faces, A the cross-sectional area of the cube, ΔT the temperature difference ($T_{\text{hot}} - T_{\text{cold}}$), and h the cube edge length.

The simulation was conducted along all three directions (x, y, and z) of the cube. The same simulations were undertaken using diamond instead of P(MMA-*co*-nBA) to obtain data for a binary colloidal assembly having a higher thermal conductivity contrast. The specific heat capacity, density, and thermal conductivity of all materials used in FEM simulations are listed in Table 12.2.

Tab. 12.2.: Specific heat capacity, density, and thermal conductivity of silica, P(MMA-*co*-nBA), and diamond used in FEM simulations.

	Specific heat capacity [$\text{Jg}^{-1}\text{K}^{-1}$]	Density [g cm^{-3}]	Thermal conductivity [$\text{mW m}^{-1}\text{K}^{-1}$]
Silica	0.684	2.20	173[a]
P(MMA-*co*-nBA)	1.441	1.16	205
Diamond[b]	0.507	3.51	2,300,000

[a]The thermal conductivity of silica is chosen such that the same effective thermal conductivity is obtained as experimentally in vacuum. [b]The literature values were obtained from Salazar[31] and Slack.[32]

12.7 References

[1] F. Tan and C. Tso, „Cooling of mobile electronic devices using phase change materials," *Applied Thermal Engineering*, vol. 24, no. 2-3, pp. 159–169, 2004.

[2] A. L. Moore and L. Shi, „Emerging challenges and materials for thermal management of electronics," *Materials Today*, vol. 17, no. 4, pp. 163–174, 2014.

[3] Y. Xia, B. Gates, Y. Yin, and Y. Lu, „Monodispersed colloidal spheres: Old materials with new applications," *Advanced Materials*, vol. 12, no. 10, pp. 693–713, 2000.

[4] G.-W. Lee, M. Park, J. Kim, J. I. Lee, and H. G. Yoon, „Enhanced thermal conductivity of polymer composites filled with hybrid filler," *Composites Part A: Applied Science and Manufacturing*, vol. 37, no. 5, pp. 727–734, 2006.

[5] Z. Han and A. Fina, „Thermal conductivity of carbon nanotubes and their polymer nanocomposites: A review," *Progress in Polymer Science*, vol. 36, no. 7, pp. 914–944, 2011.

[6] D. Hasselman and L. F. Johnson, „Effective thermal conductivity of composites with interfacial thermal barrier resistance," *Journal of Composite Materials*, vol. 21, no. 6, pp. 508–515, 1987.

[7] D. Stauffer, *Introduction To Percolation Theory*. Taylor & Francis, 2014.

[8] W. Tian; and R. Yang, „Phonon transport and thermal conductivity percolation in random nanoparticle composites," *Computer Modeling in Engineering & Sciences*, vol. 24, no. 2, pp. 123–141, 2008.

[9] Q. Xue, „The influence of particle shape and size on electric conductivity of metal–polymer composites," *European Polymer Journal*, vol. 40, no. 2, pp. 323–327, 2004.

[10] S. Kumar, M. A. Alam, and J. Y. Murthy, „Effect of percolation on thermal transport in nanotube composites," *Applied Physics Letters*, vol. 90, no. 10, p. 104 105, 2007.

[11] Y. Mamunya, V. Davydenko, P. Pissis, and E. Lebedev, „Electrical and thermal conductivity of polymers filled with metal powders," *European Polymer Journal*, vol. 38, no. 9, pp. 1887–1897, 2002.

[12] F. Lux, „Models proposed to explain the electrical conductivity of mixtures made of conductive and insulating materials," *Journal of Materials Science*, vol. 28, no. 2, pp. 285–301, 1993.

[13] J. K. Carson, S. J. Lovatt, D. J. Tanner, and A. C. Cleland, „Thermal conductivity bounds for isotropic, porous materials," *International Journal of Heat and Mass Transfer*, vol. 48, no. 11, pp. 2150–2158, 2005.

[14] J. Wang, J. K. Carson, M. F. North, and D. J. Cleland, „A new approach to modelling the effective thermal conductivity of heterogeneous materials," *International Journal of Heat and Mass Transfer*, vol. 49, no. 17-18, pp. 3075–3083, 2006.

[15] Z. Hashin and S. Shtrikman, „A variational approach to the theory of the effective magnetic permeability of multiphase materials," *Journal of Applied Physics*, vol. 33, no. 10, pp. 3125–3131, 1962.

[16] J. W. Goodwin, J. Hearn, C. C. Ho, and R. H. Ottewill, „Studies on the preparation and characterisation of monodisperse polystyrene laticee,“ *Colloid and Polymer Science*, vol. 252, no. 6, pp. 464–471, 1974.

[17] C. Graf, D. L. J. Vossen, A. Imhof, and A. van Blaaderen, „A general method to coat colloidal particles with silica,“ *Langmuir*, vol. 19, no. 17, pp. 6693–6700, 2003.

[18] P. Ruckdeschel, T. W. Kemnitzer, F. A. Nutz, J. Senker, and M. Retsch, „Hollow silica sphere colloidal crystals: Insights into calcination dependent thermal transport,“ *Nanoscale*, vol. 7, no. 22, pp. 10 059–10 070, 2015.

[19] P. Ruckdeschel and M. Retsch, „Interface and morphology control of the thermal conductivity in core-shell particle colloidal crystals,“ *Advanced Materials Interfaces*, vol. 4, no. 24, p. 1 700 963, 2017.

[20] F. A. Nutz, P. Ruckdeschel, and M. Retsch, „Polystyrene colloidal crystals: Interface controlled thermal conductivity in an open-porous mesoparticle superstructure,“ *Journal of Colloid and Interface Science*, vol. 457, pp. 96–101, 2015.

[21] L. Dusza, „Combined solution of the simultaneous heat loss and finite pulse corrections with the laser flash method,“ *High Temperatures-High Pressures*, vol. 467, pp. 27–28, 1995.

[22] J. Blumm, J. Henderson, O. Nilsson, and J. Fricke, „Laser flash measurement of the phononic thermal diffusivity of glasses in the presence of ballistic radiative transfer,“ *High Temperatures-High Pressures*, vol. 29, no. 5, pp. 555–560, 1997.

[23] N. V. Brilliantov, F. Spahn, J.-M. Hertzsch, and T. Pöschel, „Model for collisions in granular gases,“ *Physical Review E*, vol. 53, no. 5, pp. 5382–5392, 1996.

[24] L. E. Silbert, D. Ertaş, G. S. Grest, *et al.*, „Granular flow down an inclined plane: Bagnold scaling and rheology,“ *Physical Review E*, vol. 64, no. 5, 2001.

[25] H. P. Zhang and H. A. Makse, „Jamming transition in emulsions and granular materials,“ *Physical Review E*, vol. 72, no. 1, 2005.

[26] G. Mason, „Radial distribution functions from small packings of spheres,“ *Nature*, vol. 217, no. 5130, pp. 733–735, 1968.

[27] G. T. Seidler, G. Martinez, L. H. Seeley, *et al.*, „Granule-by-granule reconstruction of a sandpile from x-ray microtomography data,“ *Physical Review E*, vol. 62, no. 6, pp. 8175–8181, 2000.

[28] A. S. Clarke and H. Jónsson, „Structural changes accompanying densification of random hard-sphere packings,“ *Physical Review E*, vol. 47, no. 6, pp. 3975–3984, 1993.

[29] R. Y. Yang, R. P. Zou, and A. B. Yu, „Computer simulation of the packing of fine particles,“ *Physical Review E*, vol. 62, no. 3, pp. 3900–3908, 2000.

[30] A. R. Kansal, S. Torquato, and F. H. Stillinger, „Diversity of order and densities in jammed hard-particle packings,“ *Physical Review E*, vol. 66, no. 4, 2002.

[31] A. Salazar, „On thermal diffusivity,“ *European Journal of Physics*, vol. 24, no. 4, pp. 351–358, 2003.

[32] G. A. Slack, „Thermal conductivity of pure and impure silicon, silicon carbide, and diamond,“ *Journal of Applied Physics*, vol. 35, no. 12, pp. 3460–3466, 1964.

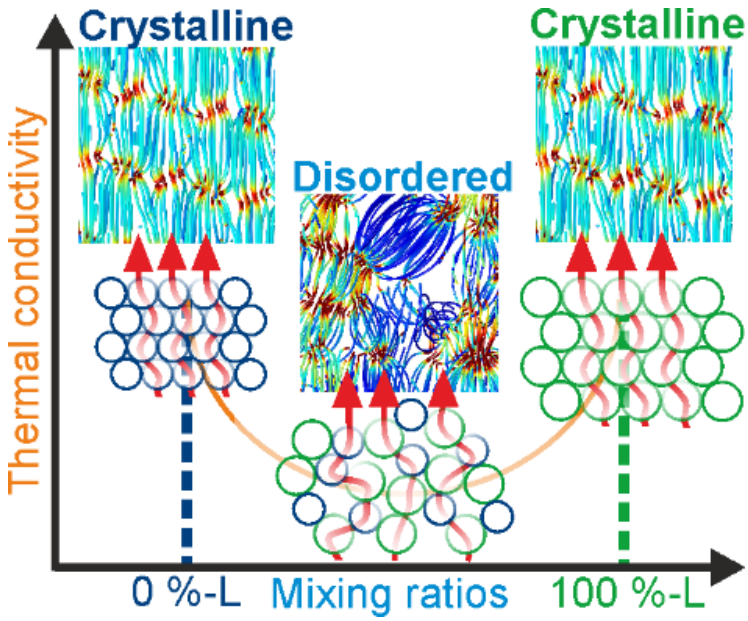
13

Low Thermal Conductivity through Dense Particle Packings with Optimum Disorder

Fabian A. Nutz,[1] Alexandra Philipp,[1] Bernd A. F. Kopera,[1] Martin Dulle[2] and Markus Retsch[1]

[1] Department of Chemistry, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

[2] JCNS-1/ICS-1, Forschungszentrum Jülich, 52425 Jülich, Germany



Published in *Advanced Materials*, **2018**, 30, 1704910
Reproduced with permission from John Wiley and Sons

13.1 Abstract

Heat transport plays a critical role in modern batteries, electrodes, and capacitors. This is caused by the ongoing miniaturization of such nano-technological devices, which increases the local power density and hence temperature. Even worse, the introduction of heterostructures and interfaces is often accompanied by a reduction in thermal conductivity, which can ultimately lead to the failure of the entire device. Surprisingly, a fundamental understanding of the governing heat transport processes even in simple systems, such as binary particle mixtures is still missing. This contribution closes this gap and elucidates how strongly the polydispersity of a model particulate system influences the effective thermal conductivity across such a heterogeneous system. In a combined experimental and modeling approach, well-defined mixtures of monodisperse particles with varying size ratios are investigated. The transition from order to disorder can reduce the effective thermal conductivity by as much as $\approx 50\%$. This is caused by an increase in the thermal transport path length and is governed by the number of interparticle contact points. These results are of general importance for many particulate and heterostructured materials and will help to conceive improved device layouts with more reliable heat dissipation or conservation properties in the future.

13.2 Introduction

Materials with low thermal conductivity are employed in many fields such as clothing, refrigeration, building insulation, and thermoelectrics.[1] Concepts to reduce the thermal conductivity of a given material either target the material composition or its nano-, meso-, or microstructure. Generally, for bulk materials, a high thermal conductivity is found in crystalline solids where phonons are able to transport thermal energy over several hundred nanometers based on a well-defined crystal lattice.[2]–[5] In contrary, low thermal conductivities are mostly found in disordered, amorphous materials. Based on an increased scattering at the disordered structure, the mean free path of phonons is strongly reduced, resulting in diffusive thermal transport.[6],[7] Allen *et al.*[8],[9] proposed three different vibrational modes, namely diffusons, propagons, and locons, to describe thermal transport in amorphous matter. Increasing phonon scattering in bulk materials, subsequently leads to a reduction in thermal conductivity. In dense materials this can be achieved by layered structures,[10],[11] doping,[12],[13] or the embedding of nanoparticles,[14],[15] where particularly crystalline solids are strongly affected. Zhang and Minnich investigated how nanoparticle clusters with a particular size distribution can lower the thermal conductivity in crystalline SiGe alloys even below its amorphous limit.[14]

Besides the chemical composition of a certain material, the influence of the overall structure is of high importance, too. The most obvious and well-known structural influence is given by the introduction of porosity. This severely decreases the density of a bulk material and consequently reduces the effective thermal conductivity. Depending on the cell size of the porous material, highly insulating properties are accessible.[16]–[19] Moreover, classical porous, granular matter becomes increasingly important for a range of applications, which involve heterostructures and interfaces. For instance, thermal management in high-density storage devices such as batteries or supercapacitors is a critical safety concern.[20]–[22]

Colloidal assembly structures have been proven to be particular suitable to investigate the influence of 3D nanostructuring on the effective thermal transport in granular matter.[23] The thermal conductivity of colloidal crystals is mainly governed by the geometrical constrictions at the interparticle contact points, and the material of the particles itself.[24]–[26] While most colloidal assembly structures targeted periodically ordered materials, the controlled introduction of disorder also showed to provide fundamental insights, for instance, into the physics of phononic bandgaps in the hypersonic regime.[27]

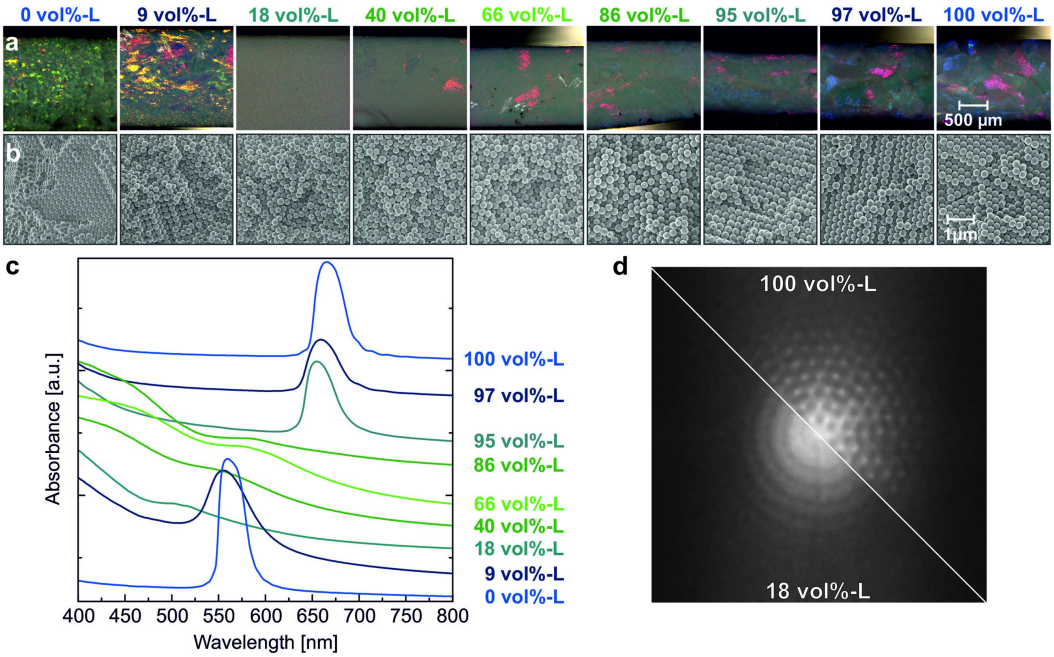


Fig. 13.1.: Structural characterization of binary colloidal assemblies. a) Optical and b) scanning electron microscopy side view images of the edges of split colloidal monoliths. c) UV–vis transmission spectra measured on dip-coated assemblies. d) Fast Fourier transformation (FFT) images from scanning electron microscopy images with a mixing ratio of 18 vol% L compared to a FFT image of 100 vol% L colloidal crystal.

In this work, we provide a conceptual understanding of the influence of structural order on the thermal transport properties in particulate mesostructures. Therefore, binary colloidal assemblies were fabricated from two monodisperse, differently sized polystyrene (PS) particles. These buildings blocks are easily accessible by common polymerization techniques[28],[29] and their self-assembly into superstructures is well-established.[30] By mixing two differently sized particles, it is possible to control the structural order of the resulting colloidal assembly. We clarify the underlying effect of the thermal conductivity reduction by finite element modeling combined with molecular dynamic simulations. We further demonstrate how to reduce heat transport in disordered binary assemblies in a rational way.

Binary colloidal assemblies of PS particles possessing a diameter of 243 nm (S) and 306 nm (L) were fabricated by evaporation-induced self-assembly (size ratio ≈ 0.8). For spectroscopic characterization, binary assemblies were additionally immobilized on glass slides by dip-coating of 3 wt% aqueous particle dispersions. The mixing ratio ranged from 0 to 100 vol% L. Optical microscopy and scanning electron images of the edges of split monoliths are shown in Figure 13.1a,b and provide a qualitative expression of how the mixing ratio translates into order and disorder.

From optical microscopy (Figure 13.1a), strong opalescent colors are visible for samples consisting of only one particle type (0 and 100 vol% L) as well as for binary assemblies only containing a small amount of differently sized specimen (9, 95, and 97 vol% L). This is due to Bragg reflection. The difference in color between 0 and 100 vol% L monoliths arises from the size dependence of the Bragg reflectivity. Different colors within one monolith can be attributed to the angle dependence of the Bragg law, based on different crystal planes exposed to the surface.

For other mixing ratios, only weak overall Bragg reflection (66 and 86 vol% L) or almost only diffuse scattering (18 and 40 vol% L) is visible. At these mixing ratios, the polydispersity of the binary particle dispersion prevents large area particle crystallization. Noteworthy, some embedded crystalline regions are observable for several intermediate mixing ratios (40, 66, and 86 vol% L). This can be explained by the strong tendency of demixing in binary colloidal dispersions.[31]–[33] The slow self-assembly process (several days) of the dispersion enables the formation of phase separated, multicrystalline, and amorphous areas. Scanning electron microscopy (SEM) images validate the optical impression (Figure 13.1b). The pure colloidal crystals feature a high degree of crystalline particle ordering. In contrast, for intermediate mixing ratios, only a random packing of the particles is observed. This is additionally highlighted by fast Fourier transformation (FFT) images of the SEM images illustrated in Figure 13.1d. Whereas distinct reflexes are visible for 100 vol% L due to the hexagonal symmetry of the particle lattice, only smeared Debye–Scherrer rings are obtained for 18 vol%-L monoliths. The smearing is based

on the disordered mesostructure, whereas the observed Debye–Scherrer rings arise from the different particle diameters present within the monoliths.

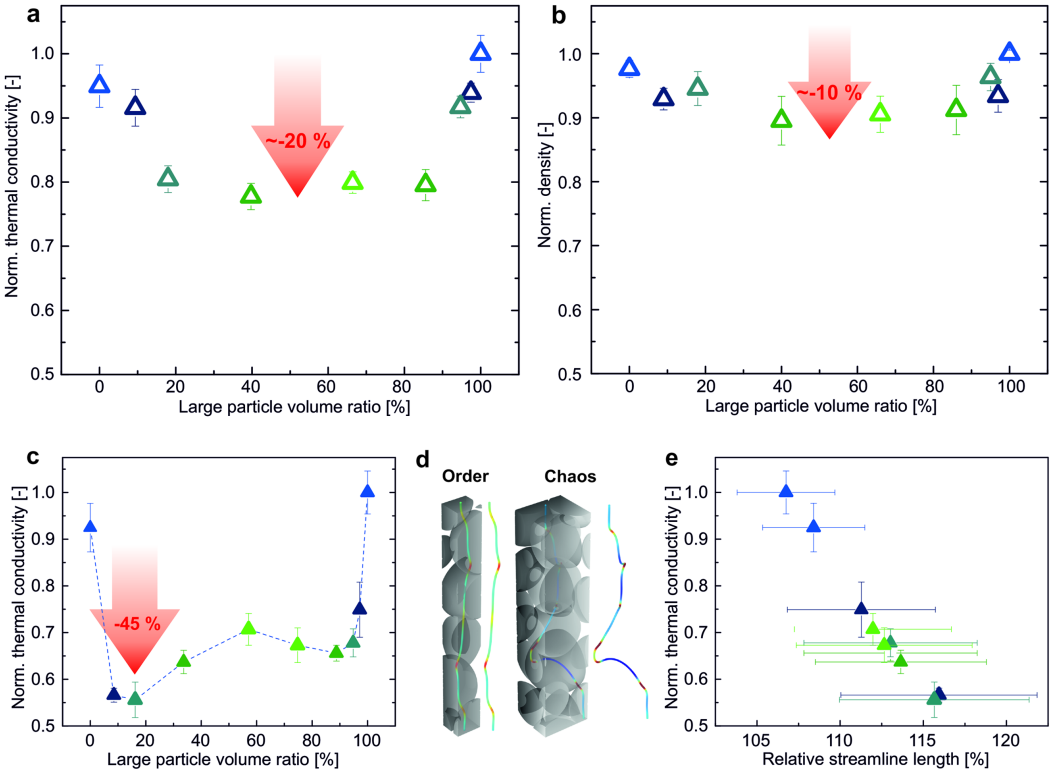


Fig. 13.2.: Thermal conductivity of the investigated assemblies with a size ratio of 0.8. Color code indicates ordered (bluish) and disordered (greenish) assemblies. a) Normalized thermal conductivity of binary PS colloidal assemblies possessing different mixing ratios. The red arrow indicates the reduction of the thermal conductivity due to the introduced disorder. b) Normalized density of the colloidal assemblies. c) Thermal conductivity versus volume fraction of large particles obtained by FEM. d) Exemplary streamlines present within a crystalline and a disordered assembly with a size ratio of 0.8 and 16 vol% L. e) Thermal conductivity versus relative streamline length of various size ratios. Error bars in (c) and (e) arise from three individual simulations boxes at three different directions in space.

We further quantify the intrinsic order of the binary colloidal films by UV-vis transmission spectra of dip-coated samples,displayed in Figure 13.1c. A strong Bragg peak is visible for 0 vol% L colloidal crystals at a wavelength of 559 nm. At small mixing ratios of 9 vol% L, the Bragg peak broadens and the maxima is less allocable due to the disturbance of the crystal lattice. A further increase of the large particle volume fraction to 18 vol% L leads to a vanishing of the Bragg peak. No distinct Bragg peaks are observable for intermediate mixing ratios from 44 to 86 vol% L, indicating a randomly packed structure. The Bragg peak of the larger particle species reoccurs at 95 vol% L and reaches a maximum intensity for 100 vol% L at a wavelength 667 nm.

Optical microscopy, SEM, and UV-vis characterization prove the highly crystalline nature of the homo-particle ensembles, which define the edges of the mixture phase diagram. The long-range order is quickly lost, when mixing two particles with a size ratio of 0.8. Depending on the assembly process a complete suppression of crystallinity can be found (dip-coating) or small, multicrystalline regions caused by demixing prevail (dispersion drying).

Laser flash analysis (LFA) was performed on colloidal monoliths obtained from evaporation-induced self-assembly. By determining the sample density and specific heat capacity, it is possible to calculate the specimen's effective thermal conductivity (see the Supporting Information for details). The thermal conductivity, normalized to the maximum initial value of the pure colloidal assemblies, as well as the normalized thermal conductivity of intermediate mixing ratios are show in Figure 13.2a. The color code helps to separate ordered samples (bluish) from randomly packed assemblies (greenish) and correspond to the mixing ratios given in Figure 13.1.

Homo-particle colloidal assemblies (mixing ratio 0 and 100 vol% L) show a thermal conductivity of 73 and 77 $\mathrm{mWm^{-1}K^{-1}}$in absolute numbers, respectively. At small mixing ratios of differently sized particles, the thermal conductivity slightly drops to $\approx 93\%$ of the initial value. For intermediate mixing ratios, the thermal conductivity drops significantly to $\approx 80\%$. This reduction can only partially be explained by the reduced density of the intermediate assemblies as displayed in Figure 13.2b.

The reduction of the density for intermediate mixing ratios originates from the prevented close packing of the particles during assembly. This leads to a space filling which deviates from the theoretical maximum of 74%. Still, the density only decreases to $\approx 90\%$ of its initial value. Therefore, an additional effect is present, which is responsible for the overall reduction of the thermal conductivity to 80%.

To elucidate this additional contribution, we performed finite element method (FEM) modeling on binary particle mixtures possessing a size ratio of 0.8. The required virtual assemblies were obtained by molecular dynamic (MD) simulations of the assembly process. Exemplary particle assemblies obtained from MD simulations are shown in Figure 13.7a (Supporting Information). We found highly crystalline particle assemblies at the edges of the mixing diagram. In contrast to our experimental findings, the crystallization process is already fully prevented at very small mixing ratios (Figure 13.7a, left, Supporting Information). This is caused by the lack of demixing or phase separation of the particles due to the short assembly timescale used for the MD simulation. FFT images of 0 and 9 vol% L assemblies from MD simulation are compared to the experiment in Figure 13.7a (Supporting Information), right. Thus, the fully amorphous structure obtained from the MD simulation can be regarded as an ideally disordered case. Experimentally, some

degree of crystallization will be difficult to exclude owing to the long timescale allowed for the self-assembly process. FEM has been conducted using the software COMSOL Multiphysics. Details on the modeling are described in the Supporting Information. The calculated thermal conductivity against the volume fraction of large particles is displayed in Figure 13.2c. Analogous to Figure 13.2a, the color code indicates ordered (bluish) and disordered assemblies (green). The mixing diagram shows a similar trend for the thermal conductivity as observed for the experiment, with high thermal conductivities for ordered, and a reduced thermal conductivity for disordered assemblies. A minimum thermal conductivity of only 55% of the initial value is observed at a volume fraction of ≈ 16 vol% L. This is an almost twofold enhancement of the insulation capability caused by the disordered structure, and it is significantly lower compared to the experimental results. We ascribe this deviation to the structural difference between the MD simulation and the experiment. As outlined before, by MD simulation one can observe a more disordered distribution of both particle types within the monolith, leading to a stronger reduction of the thermal conductivity.

To explain the origin of the thermal conductivity reduction beyond the decrease in density, we utilized FEM to analyze the length of the thermal transport pathway through such a particulate assembly structure. This can be done by evaluating the length of heat flux streamlines, which reach from the hot to the cold side of the simulation box. Figure 13.2d shows two individual cases through an ordered (100 vol% L) and disordered (≈ 16 vol% L) particle ensemble. Whereas in the ordered case, a rather straight and unperturbed streamline is obtained, the introduction of disorder strongly bents and perturbs the thermal transport path. The statistic evaluation over the entire simulation box is given in Figure 13.2e, where we normalized the streamline length on the length of the simulation box. One finds a direct correlation between the streamline length and the thermal conductivity reduction. This finding can be intuitively explained by a sort of effective thermal length, which differs from the plain geometric size of the box. One may draw an analogy to the concept of the diffusive optical path in a particulate powder, which is governed by the scattering cross section of the constituting sphere.[34]

Most interestingly, the FEM simulation finds a strongly asymmetric shape of the thermal conductivity reduction with two minima at moderately high and low mixing ratios, respectively. These are not found in the experiment, presumably due to the presence of mesocrystalline domains, embedded in the amorphous structure. We now want to understand the origin of these minima and how the thermal conductivity could be reduced even further. This question could be rephrased to how the thermal streamline length can be further increased in such colloidal superstructures. We, therefore, return to MD simulation and evaluate the local structure around one particle more explicitly. We did this in a twofold way: (1) Using the radial

distribution function we evaluate the number of particles surrounding a center particle, which are within the local space of about +10% of the respective particle diameter. (2) Using the individual coordinates of each particle pair, we extracted the number of touching spheres around a central particle. We term the first one next neighbors, and the second one contact neighbors. Quite expectedly, when introducing disorder the average number of next and contact neighbors is reduced. For crystalline assemblies, one can find an average number of next neighbors per particles of ≈ 12, based on the face-centered cubic symmetry. For all disordered intermediate mixing ratios, the number of next neighbors is reduced to ≈ 10 (Figure 13.7b, Supporting Information). The next contact neighbor analysis yields ≈ 8 owing to the more rigorous exclusion criterion. One can imagine that this reduction leads to less pathways for heat to travel through the particle structures and therefore reduces the thermal conductivity of the assemblies. However, we find that the mean number of next neighbor particles merely reaches a broad plateau, and does not explain the asymmetric shape shown in Figure 13.2c. We, therefore, evaluated the next neighbor particle histograms explicitly. This is shown in Figure 13.3, where we present the next neighbor histograms at selected mixing ratios (bottom row). We chose a special illustration, to highlight the structural heterogeneity, when heat travels through such a system. We, therefore, present a projection of all particles with the same number of next neighbors that can be found within the simulated box. The color code indicates particles possessing the same number of next neighbors from high (blue) to low (red). It therefore provides a direct impression of how many particles are clustered around one central sphere.

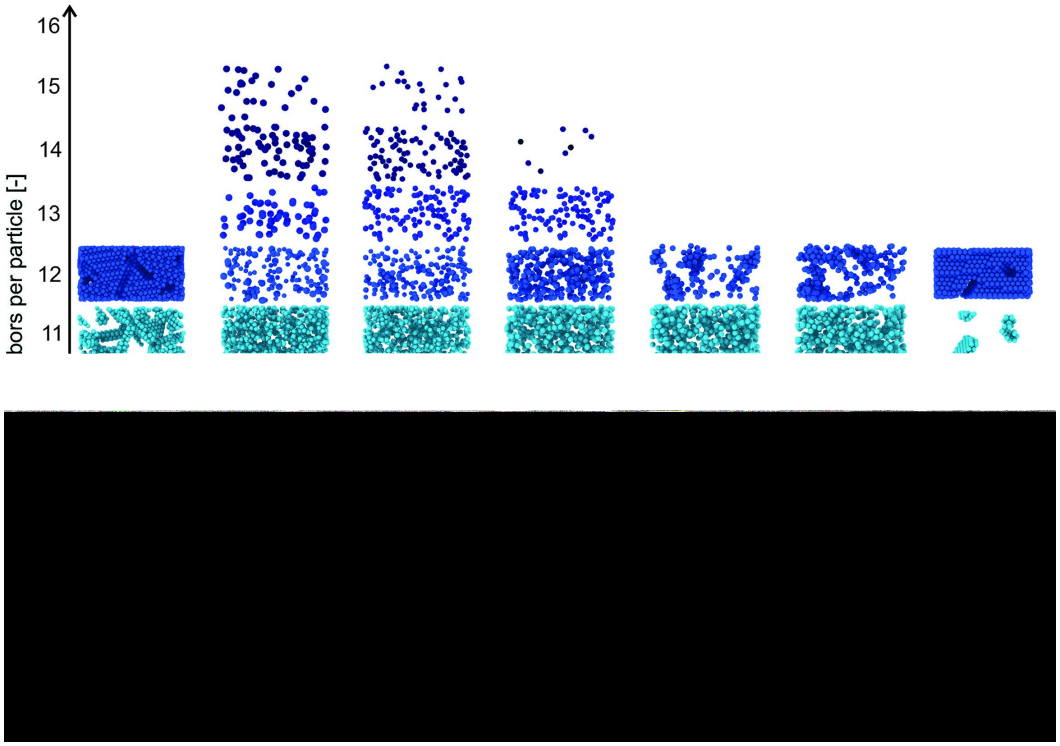


Fig. 13.3.: Molecular dynamics simulations of the assembly process of a binary particle mixture. Number of next neighbors per particle against volume fraction of large particles. Size ratio is 0.8. Projection of the assemblies visualizes the particles possessing the indicated number of neighbors. Histograms show the relative frequency of particle neighbors per particle present within the different assemblies.

For mixing ratios of 0 and 100 vol% L almost every particle possesses 12 next neighbors. This indicates a nearly fully crystalline face-centered cubic assembly, with few line and point defects being visible. The next neighbor distribution function quickly broadens at intermediate mixing ratios. Most importantly, this distribution function shows a strongly asymmetric behavior. For mixing ratios of 9–66 vol% L, even more than 12 next neighbors per particles are present within the assembly, whereas for high mixing ratios, the number of next neighbors is limited to 12. For a low number of large particles (small mixing ratios), it is possible that one large particle is decorated by a high number of smaller particles. In contrary, for high mixing ratios of large particles, it is geometrically not possible to decorate a small particle with a high number of large particles, since there is simply not sufficient space. This asymmetry is less pronounced when considering the contact neighbors (Figure 13.10, Supporting Information).The average contact number stays constant with a tendency toward fewer contact points at the minimum volume fraction ≈ 20 vol% L. This discrepancy between a constant or reduced number of contact neighbor particles compared to the evolution of many next neighbor particles hints toward the importance of a jammed particle corona in close vicinity to the central bead.

We also conducted a thorough analysis of the interparticle constriction effect in the Supporting Information (Figures 13.11–13.13, Supporting Information). We, therefore, analyzed the average interparticle contact area and relate this to the contact neighbor distribution. Using an ideal face-centered cubic reference system we show that a reduction of the interparticle contact area leads to an increase in thermal path length caused by a stronger bending of the heat flux stream lines (Figure 13.11b, Supporting Information). However, the changes of the interparticle contact area play a secondary role compared to the introduction of disorder and consequently to the thermal path length.

The consequence of the asymmetric next neighbor distribution for the effective thermal length is rather surprising. A reduction of the number of next neighbor particles indeed leads to a reduction in thermal conductivity. However, even more efficient is the introduction of a certain amount of many jammed and not necessarily directly contacting particles, since this leads to an even stronger disturbance of the streamline length due to a geometric extension of the predetermined thermal transport path. This is accompanied by a minimum of the total number of interparticle contact points within the simulation volume (Figure 13.10, Supporting Information).

To clarify how this transport pathway is governed by the size mismatch of the two particles, we varied the size ratio of the particles between 0.9 and 0.54 at the mixing ratio around 20 vol% L, which showed the lowest thermal conductivity. Noteworthy for this data set, the experimental samples were prepared by filtration to speed up the assembly time and therefore obtain particle structures, which are more comparable to the MD assemblies. Both, experiment and FEM simulation, show a systematic decrease of the thermal conductivity with a decreasing size ratio from 0.9 to 0.54 (Figure 13.4d). The increase of the thermal conductivity for the smallest size ratio (0.54) for the experiment can be explained by an increasing tendency to demix as outlined above. One also needs to take into account that fully crystalline binary assemblies can be obtained for size ratios of < 0.41 owing to the octahedral voids in face-centered cubic structures, which we approach with this small size ratio.[35]

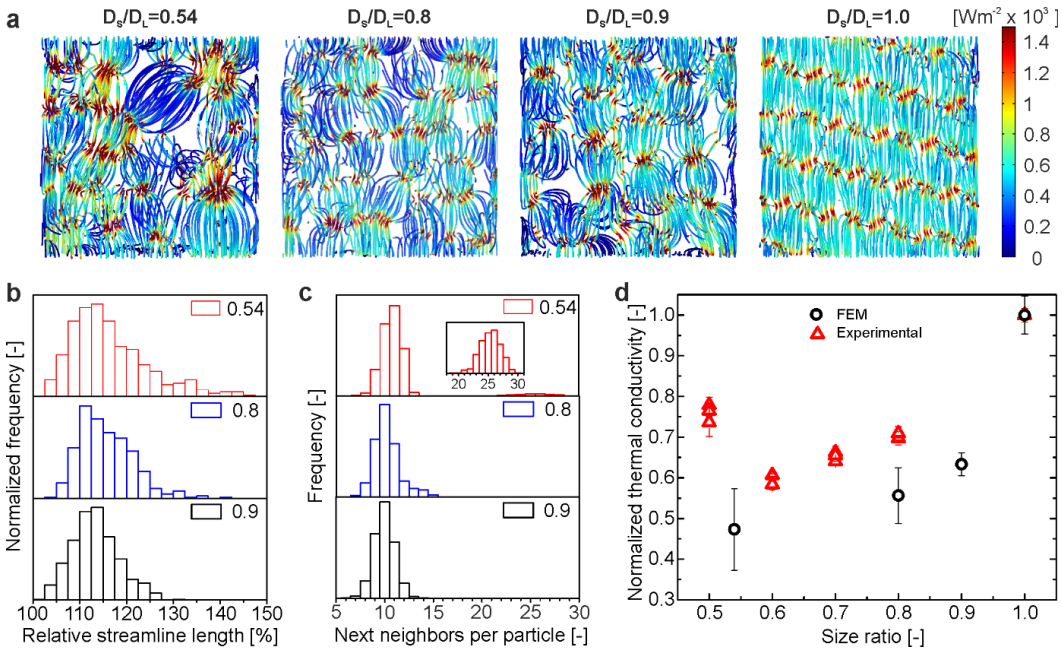


Fig. 13.4.: Influence of the size ratio (D_S/D_L) on the thermal conductivity at ≈ 20 vol% L. a) Comparison of the heat flux densities of particle assemblies possessing different size ratios. b) Histograms of the streamline length and c) next neighbors per particle of the assemblies. d) Resulting thermal conductivity from FEM compared to experimental data.

Figure 13.4a compares the heat flux density of the particle assemblies from size ratios of 0.54, 0.8, and 0.9 to the heat flux density of a fully crystalline assembly (size ratio: 1.0). Crystalline assemblies feature uniform and straight streamlines (Figure 13.4a, 1.0), whereas a decreasing size ratio evokes a high degree of distortion. This is especially obvious for the heat flux density of the size ratio 0.54, where one large particle heavily impedes the heat flux through the entire particle box. The qualitative impression from the heat flux density images is verified by comparing the relative streamline lengths of the different size ratios (Figure 13.4b). The smaller the size ratio, the higher the number of long streamlines, which can reach up to 140% relative to the length of the simulation box. This indicates that the less the particle sizes match, the stronger the heat flux is bent along the particle network, leading to an elongation of the thermal transport path.

The higher fraction of long streamlines coincides with a broadening of the next neighbor distribution function with decreasing size ratio (Figure 13.4c). The broadening increases the number of > 10 next neighbors, indicating an increasing decoration of a large particle with small particles. However, due to the small number of large particles at this volume fraction and size ratio combination (about 4% number ratio of large particles) this effect is not visible in the average number of neighbors for the whole assembly. Only when looking at the detailed next neighbor histograms the results from the heat flux simulations can be understood (Figure 13.4c). For a size

ratio of 0.54 a second fraction with even $25 - 30$ next neighbors is found. Looking at the contact neighbor distribution for this sample, the number of touching spheres remains rather constant at about 8 (Figure 13.10, Supporting Information). This highlights the evolution of a densely jammed corona, where a small amount of large particle clusters are decorated with smaller spheres. This constitutes an efficient way to increase the thermal path length. Overall, the larger mismatch between the two particle sizes led to a thermal conductivity reduction by 40% (size ratio 0.6) in the experiment and 53% (size ratio 0.54) in the FEM simulation. We finally also checked, whether 19 vol% L represents the absolute minimum with respect to the mixing ratio. We, indeed, find the highest fraction of > 25 next neighbor particles for this mixing ratio (Figure 13.9, Supporting Information). Higher or lower mixing ratios (± 5 vol% L) lead rapidly to a reduced number of next neighbors. Therefore, a mixing ratio of 19 vol% L of large particles is sufficient to strongly reduce the effective thermal transport across a binary particle mixture.

We finally compare this disorder derived thermal insulation effect to the thermal conductivity of known polymer materials. The unique properties of colloidal crystals and especially colloidal glasses can be inferred from the Ashby plot in Figure 13.5.[36] It shows a direct relation between the thermal conductivity and the polymer density. Using polymer foams, highly insulating materials are accessible, however, at the expense of the materials density. Transitioning from colloidal crystals to colloidal glasses, retains the materials density to a large extent, while the thermal conductivity decreases. This opens the avenue to design highly insulating granular materials at a retained high density.

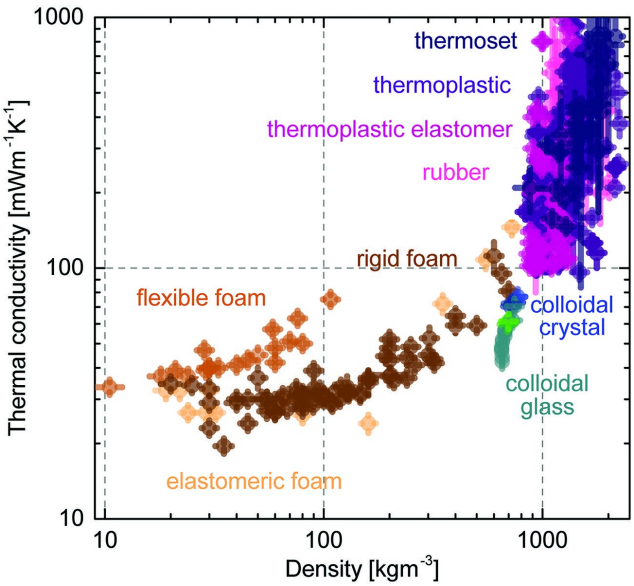


Fig. 13.5.: Ashby plot of density versus thermal conductivity of bulk and foam polymer materials. The introduction of disorder to a colloidal ensemble reduces the thermal conductivity (measured in vacuum) while retaining a comparatively high density.

In summary, we investigated the influence of order and disorder in particle mixtures on their thermal transport properties. We therefore choose a controlled mixture of binary latex particles as a model system. In a combined approach, which comprised experimental characterization, MD simulation as well as FEM, we were able to relate the observed reduction in thermal conductivity to the local environment of the colloidal particles. Quite surprisingly, the increase of next neighbor contact points at low mixing ratios and at a high size mismatch between the binary spheres, represents the most efficient way to attain a minimum thermal conductivity. This is achieved at comparatively high density. The decisive concept is the increase in the effective thermal length, across which the thermal energy needs to be transported. Overall, disorder in a binary colloidal glass can reduce the thermal conductivity by as much as $\approx 50\%$ compared to its single phase, crystalline counterpart. Our findings are of high importance for heterostructured and particulate materials, for applications were a high insulation, but also a good heat dissipation are paramount. Owing to the diffusive thermal transport in our system, our findings are of conceptual nature and applicable to many other nano-, micro-, and macrostructured materials, beyond simple latex beads. Most importantly, we want to stress the relevance of the concept of an effective thermal length, which cannot only be increased by constriction but may be even more strongly affected by additional pathways and detours.

13.3 Experimental Section

Details on the particle synthesis, binary colloidal self-assembly methods, and characterization methods, as well as a detailed description of the thermal transport characterization by laser flash analysis, molecular dynamics simulations, finite element modeling, complementary contact particle analysis, and a discussion on the influence of the particle contact area can be found in the Supporting Information.

Acknowledgments

F.A.N. and A.P. contributed equally to this work. The authors thank Ute Kuhn for help with DSC measurements. Tobias Lauster, Julian Sindram, and Dominik Skrybeck are acknowledged for performing firsts UV–vis and LFA experiments. B.A.F.K. acknowledges support from the Elite Network of Bavaria (ENB). Additional support was provided by the SFB840 and the Bavarian Polymer Institute, BPI. This work was funded by the Volkswagen foundation through a Lichtenberg professorship.

Conflict of Interest

The authors declare no conflict of interest.

Keywords

colloidal glass, next neighbor correlation, photonic crystal, powder sintering, thermal transport

13.4 References

[1] B. Poudel, Q. Hao, Y. Ma, *et al.*, „High-thermoelectric performance of nanostructured bismuth antimony telluride bulk alloys,“ *Science*, vol. 320, no. 5876, pp. 634–638, 2008.

[2] D. G. Cahill, P. V. Braun, G. Chen, *et al.*, „Nanoscale thermal transport. II. 2003–2012,“ *Applied Physics Reviews*, vol. 1, no. 1, p. 011 305, 2014.

[3] D. G. Cahill, W. K. Ford, K. E. Goodson, *et al.*, „Nanoscale thermal transport,“ *Journal of Applied Physics*, vol. 93, no. 2, pp. 793–818, 2003.

[4] D. G. Cahill and R. O. Pohl, „Lattice vibrations and heat transport in crystals and glasses,“ *Annual Review of Physical Chemistry*, vol. 39, no. 1, pp. 93–121, 1988.

[5] M. Maldovan, „Sound and heat revolutions in phononics,“ *Nature*, vol. 503, no. 7475, pp. 209–217, 2013.

[6] A. Einstein, „Elementare betrachtungen über die thermische molekularbewegung in festen körpern,“ *Annalen der Physik*, vol. 340, no. 9, pp. 679–694, 1911.

[7] D. G. Cahill, S. K. Watson, and R. O. Pohl, „Lower limit to the thermal conductivity of disordered crystals,“ *Physical Review B*, vol. 46, no. 10, pp. 6131–6140, 1992.

[8] P. B. Allen and J. L. Feldman, „Thermal conductivity of disordered harmonic solids,“ *Physical Review B*, vol. 48, no. 17, pp. 12 581–12 588, 1993.

[9] P. B. Allen, J. L. Feldman, J. Fabian, and F. Wooten, „Diffusons, locons and propagons: Character of atomie yibrations in amorphous si,“ *Philosophical Magazine B*, vol. 79, no. 11-12, pp. 1715–1731, 1999.

[10] C. Chiritescu, D. G. Cahill, N. Nguyen, *et al.*, „Ultralow thermal conductivity in disordered, layered WSe_2 crystals,“ *Science*, vol. 315, no. 5810, pp. 351–353, 2007.

[11] M. D. Losego, I. P. Blitz, R. A. Vaia, D. G. Cahill, and P. V. Braun, „Ultralow thermal conductivity in organoclay nanolaminates synthesized via simple self-assembly,“ *Nano Letters*, vol. 13, no. 5, pp. 2215–2219, 2013.

[12] J. Moon and A. J. Minnich, „Sub-amorphous thermal conductivity in amorphous heterogeneous nanocomposites,“ *RSC Advances*, vol. 6, no. 107, pp. 105 154–105 160, 2016.

[13] H. Mizuno, S. Mossa, and J.-L. Barrat, „Beating the amorphous limit in thermal conductivity by superlattices design,“ *Scientific Reports*, vol. 5, no. 1, 2015.

[14] H. Zhang and A. J. Minnich, „The best nanoparticle size distribution for minimum thermal conductivity,“ *Scientific Reports*, vol. 5, no. 1, 2015.

[15] W. Kim, J. Zide, A. Gossard, *et al.*, „Thermal conductivity reduction and thermoelectric figure of merit increase by embedding nanoparticles in crystalline semiconductors,“ *Physical Review Letters*, vol. 96, no. 4, 2006.

[16] B. Notario, J. Pinto, E. Solorzano, *et al.*, „Experimental validation of the knudsen effect in nanocellular polymeric foams,“ *Polymer*, vol. 56, pp. 57–67, 2015.

[17] M. Koebel, A. Rigacci, and P. Achard, „Aerogel-based thermal superinsulation: An overview,“ *Journal of Sol-Gel Science and Technology*, vol. 63, no. 3, pp. 315–339, 2012.

[18] X. Lu, M. C. Arduini-Schuster, J. Kuhn, *et al.*, „Thermal conductivity of monolithic organic aerogels,“ *Science*, vol. 255, no. 5047, pp. 971–972, 1992.

[19] S. N. Schiffres, K. H. Kim, L. Hu, *et al.*, „Gas diffusion, energy transport, and thermal accommodation in single-walled carbon nanotube aerogels,“ *Advanced Functional Materials*, vol. 22, no. 24, pp. 5251–5258, 2012.

[20] Z. Chen, P.-C. Hsu, J. Lopez, *et al.*, „Fast and reversible thermoresponsive polymer switching materials for safer batteries,“ *Nature Energy*, vol. 1, no. 1, 2016.

[21] C. Sangros, C. Schilde, and A. Kwade, „Effect of microstructure on thermal conduction within lithium-ion battery electrodes using discrete element method simulations,“ *Energy Technology*, vol. 4, no. 12, pp. 1611–1619, 2016.

[22] C. Masarapu, H. F. Zeng, K. H. Hung, and B. Wei, „Effect of temperature on the capacitance of carbon nanotube supercapacitors,“ *ACS Nano*, vol. 3, no. 8, pp. 2199–2206, 2009.

[23] P. Ruckdeschel and M. Retsch, „Interface and morphology control of the thermal conductivity in core-shell particle colloidal crystals,“ *Advanced Materials Interfaces*, vol. 4, no. 24, p. 1 700 963, 2017.

[24] F. A. Nutz and M. Retsch, „Interfacial and volumetric sensitivity of the dry sintering process of polymer colloidal crystals: A thermal transport and photonic bandgap study,“ *Physical Chemistry Chemical Physics*, vol. 19, no. 24, pp. 16 124–16 130, 2017.

[25] F. A. Nutz, P. Ruckdeschel, and M. Retsch, „Polystyrene colloidal crystals: Interface controlled thermal conductivity in an open-porous mesoparticle superstructure,“ *Journal of Colloid and Interface Science*, vol. 457, pp. 96–101, 2015.

[26] P. Ruckdeschel, T. W. Kemnitzer, F. A. Nutz, J. Senker, and M. Retsch, „Hollow silica sphere colloidal crystals: Insights into calcination dependent thermal transport,“ *Nanoscale*, vol. 7, no. 22, pp. 10 059–10 070, 2015.

[27] T. Still, W. Cheng, M. Retsch, *et al.*, „Simultaneous occurrence of structure-directed and particle-resonance-induced phononic gaps in colloidal films,“ *Physical Review Letters*, vol. 100, no. 19, 2008.

[28] J. W. Goodwin, J. Hearn, C. C. Ho, and R. H. Ottewill, „Studies on the preparation and characterisation of monodisperse polystyrene laticee,“ *Colloid and Polymer Science*, vol. 252, no. 6, pp. 464–471, 1974.

[29] R. H. Ottewill and J. N. Shaw, „Studies on the preparation and characterisation of monodisperse polystyrene latices,“ *Kolloid-Zeitschrift und Zeitschrift für Polymere*, vol. 218, no. 1, pp. 34–40, 1967.

[30] N. Vogel, S. Utech, G. T. England, *et al.*, „Color from hierarchy: Diverse optical properties of micron-sized spherical colloidal assemblies,“ *Proceedings of the National Academy of Sciences*, vol. 112, no. 35, pp. 10 845–10 850, 2015.

[31] P. N. Pusey and W. van Megen, „Phase behaviour of concentrated suspensions of nearly hard colloidal spheres,“ *Nature*, vol. 320, no. 6060, pp. 340–342, 1986.

[32] M. Dijkstra, R. van Roij, and R. Evans, „Phase behavior and structure of binary hard-sphere mixtures,“ *Physical Review Letters*, vol. 81, no. 11, pp. 2268–2271, 1998.

[33] A. V. A. Kumar, „Binary colloidal mixtures in a potential barrier: Demixing due to depletion,“ *The Journal of Chemical Physics*, vol. 138, no. 15, p. 154 903, 2013.

[34] M. Retsch, M. Schmelzeisen, H.-J. Butt, and E. L. Thomas, „Visible mie scattering in nonabsorbing hollow sphere powders,“ *Nano Letters*, vol. 11, no. 3, pp. 1389–1394, 2011.

[35] J. Wang, S. Ahl, Q. Li, *et al.*, „Structural and optical characterization of 3d binary colloidal crystal and inverse opal films prepared by direct co-deposition,“ *Journal of Materials Chemistry*, vol. 18, no. 9, p. 981, 2008.

[36] Granta Design, *Ces edupack*, 2018.

13.5 Supporting Information

Experimental Section

Materials: Potassium persulfate (KPS, $\geq 99\,\%$, Aldrich), styrene ($\geq 99\,\%$, Aldrich) and acrylic acid (99 %, AA, Aldrich) were used as received. Water was taken from a Millipore Direct Q3UV unit for the entire synthesis and purification steps.

Particle Synthesis: In a typical synthesis, a 500 ml three-neck flask was charged with 245 ml milliQ water and 30 ml of styrene. The mixture was heat to 75 °C at a stirring speed of 650 rpm and allowed to equilibrate under a slight argon flow for $\sim$ 5 min. Subsequently, 3 ml of acrylic acid were added, and the mixture was allowed to further equilibrate for $\sim$ 10 min. To initiate the polymerization, 100 mg KPS, dissolved in 5 ml milliQ water were rapidly added to the mixture. The polymerization was carried out overnight under a slight argon flow. To adjust the size of the particles the amount of acrylic acid has been varied.[1],[2] The reaction was stop by exposing the mixture to ambient atmosphere. The particles were purified by dialysis against milliQ water for five days, changing water twice a day.

Binary colloidal self-assembly: Binary colloidal assemblies were fabricated by evaporation-induced self-assembly of a given amount of prior mixed binary particle dispersions in a PTFE beaker. The obtained monoliths generally possess a diameter of $\sim$ 20 mm and a thickness of several hundred micrometers.

Dip-coating: Samples for UV-vis measurements were prepared on glass slides by dip-coating 3 wt% of binary particle solutions. The glass slides were purified with Hellmanex solution an ethanol prior to use. Furthermore, the dip-coating process the glass slides were treated with oxygen plasma to ensure a constant wetting of the particle solutions.

Methods:

Dynamic light scattering: Dynamic light scattering measurements were performed on aqueous particle dispersion at a Malvern Zetasizer with 175° backscattering geometry to determine the hydrodynamic diameter of the particles and their size distribution
Light microscopy: Bright field light microscopy side view images of the edges of split of the colloidal assemblies were recorded on a Carl Zeiss Axio Imager.A2m light microscope equipped with an AxioCam Icc 1 camera to get a qualitative impression about the long range ordering of the colloidal particles.

Scanning electron microscopy: To determine the hard sphere diameter of the particles, scanning electron microscopy (SEM) was performed on a Zeiss Leo 1530 electron microscope. Therefore, dilute particle dispersions were drop-casted on a silicon wafer and sputtered with 1.3 nm platinum. Furthermore, SEM side-view images of the edges of split of the colloidal monoliths were obtained to gain an impression of the order within the interior of the monoliths. Fast Fourier transformation was applied on the recorded images using the FFT function, implemented in the software ImageJ 146.r.

Differential scanning calorimetry: Heat capacity determination was carried out according to ASTM E1269 on a TA Instruments Q1000 differential scanning calorimeter under a nitrogen flow of 50 ml min^{-1} at a heating rate of 20 K min^{-1}. Two heating cycles were conducted between -40 °C and 200 °C. The specific heat capacity was evaluated from the second heating cycle. The mean value from all polystyrene particles under investigation has been used for evaluation (Figure 13.6).

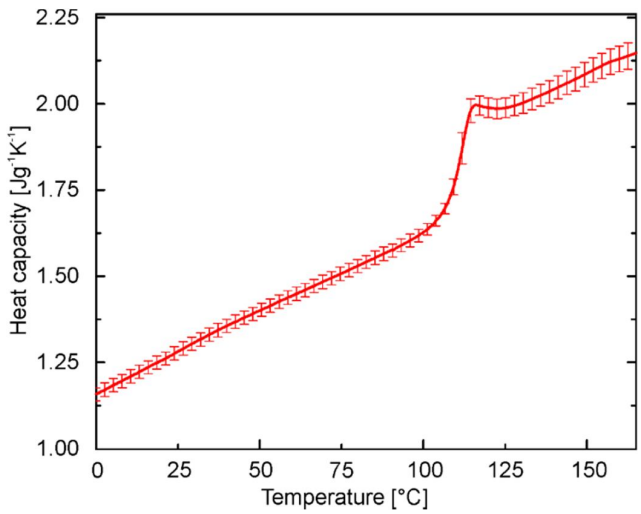


Fig. 13.6.: Mean specific heat capacity of the various polystyrene binary assemblies. Error bars represent the standard deviation of the mean value of the specific heat of every mixing ratio.

Density determination: The density of the colloidal assemblies was determined by the volume and mass of the samples. The volume was obtained from a Keyence V3100 3D digital macroscope.

Laser flash analysis: Laser flash analysis was performed on a Linseis XFA 500 XenonFlash apparatus equipped with an InSb infrared detector to obtain the thermal diffusivity of the colloidal assemblies. The thickness of the samples were determined on a Mitotoyo Litematic VL-50. Prior to the measurement, the bottom and top side of the samples were coated with a thin layer of black air-brush color serving as a blocking layer. Furthermore, the samples were coated with a thin graphite

layer ($< 15\,\mu m$). This enables a good absorbance at the bottom, and high emissivity at the top side of the monoliths. Measurements were performed on at least three individual samples. The standard deviation derives from the average of these individual measurements. Experiments were conducted in vacuum atmosphere at a pressure of $\sim 5 \cdot 10^{-2}$ mbar to exclude any influence of the ambient atmosphere. The raw data was fitted with the radiation fit model provided by the software Aprosoft Laser Flash Evaluation v.1.06. Measurements were performed on at least three samples. A xenon lamp emits a light pulse onto the sample. The thermal energy of the light pulse is absorbed at the bottom graphite layer and travels through the specimen. The thermal energy is then emitted from the top side. This temperature rise is recorded in dependence of the elapsed time since the light pulse by a fast mid-infrared detector. A numerical fitting procedure provided by the commercial software package Aprosoft Laser Flash Evaluation v1.06 based on the one-dimensional temperature diffusion equation was used to determine the thermal diffusivity α.[3] For this, accurate knowledge of the sample thickness is a prerequisite, as the thickness strongly influences half-rise time as given by:

$$\alpha(T) = \frac{1.38 \cdot d^2}{\pi^2 t_{1/2}} \tag{13.1}$$

$t_{1/2}$ represents the time needed for the half maximum temperature rise at the top surface.[4] The thermal conductivity κ of the sample is calculated by

$$\kappa(T) = \alpha(T) \cdot c_{\mathrm{p}}(T) \cdot \rho(T) \tag{13.2}$$

with the specific heat capacity c_{p} and the density ρ.

Molecular dynamics (MD) simulations: All dense colloidal assemblies used in the finite element modeling were obtained using LAMMPS.[5] Visualization was done with Ovito.[6] Every simulation consisted of 10000 arbitrarily placed spheres of the desired size and number ratio. The simulation box had periodic boundaries and was elongated in the z-direction at the start. This was done to mimic the evaporation of solvent for the real-world system. Using the NPT barostat we compressed the box first along the z-axis up to a pressure where all axes had a comparable length. After that the box was compressed isometrically to a pressure at which no more compression without overlap of the spheres was possible. We used the colloidal Yukawa potential included in the LAMMPS code with a short range repulsive part. Each simulation was run for 60 million steps with a stepsize 0.0001. We chose the colloidal Yukawa potential in LAMMPS[7] because it treats the spheres as impenetrable bodies with a fixed radius which prevents unwanted overlap even at very high pressures. The Yukawa potential was necessary in order to facilitate the formation of the fcc phases at the edges of the phase diagram and the real particles

Tab. 13.1.: MD simulation interaction parameters for the size ratio 0.54.

Pair style	Screening length	Interaction cutoff (r_c)	A for 1:1	A for 1:2	A for 2:2
Yukawa/colloid	50	1.5	10000	7500	5000

also carry a negative net charge. The pairwise interaction is calculated using the formula:

$$E = \frac{A}{\kappa} \exp\left(-\kappa\left(r - (r_i + r_j)\right)\right); \; r < r_c \tag{13.3}$$

With κ being the screening length (inverse distance units), r_i and r_j are the radii of the two interacting spheres. The prefactor A has to be specified for each pair type. It is determined from the relationship of the surface charge to surface potential in the presence of an electrolyte. In our case this prefactor is proportional only to the radius of each particle. All other parameters are the same.

We chose a large screening length of 50 which corresponds to 1/50 of the diameter of the large particles in order to hinder crystallization at higher interparticle distances and ensure good mixing of the differently sized particles. As the particles in the experiments are a few hundred nanometers in size this large screening length also approximates the charge interaction distance relative to the particle size. An example of the interaction parameters is given in Table 13.1. The analysis of the resulting dense colloidal assemblies was done with a self-written software using C++. The software calculates the radial distribution function (G(r)) for the given set of coordinates and uses the first three peaks to determine the three possible next neighbor distances as well as the next neighbor distance cut off. This cut-off was set to 10% of the particle diameter. In the MD-simulation the type of particle (small or big) is set and the software choses according to this type the appropriate condition to test whether a particle constitutes a next neighbor or not. This enabled us to obtain very detailed information on the local structure around each particle. This procedure also counts particles as next neighbors that are not touching, which is why introduced the additional term contact neighbors to distinguish these two sets of neighbors. It also calculates the density as well as the average number of neighbors.

We used a self-written Python script to evaluate the contact area between particles. The script evaluates the number of contacting particles based on the coordinates of the individual spheres. We defined the contact area as the circle created by the intersection of two spherical particles. The area was normalized to the number averaged particle surface of all particles. We increased the size of the original spheres from the MD simulation by a factor of 1.04 in order to make the spheres touch (interpenetrating spheres cannot be simulated by MD). The same scaling factor was used for finite element modeling, and was selected such to give comparable

interparticle contact areas to our experimental findings. From this we deduced the contact neighbor histogram. The individual contact areas were further correlated to number of contact neighbors.

Finite element modeling: We used the finite element method (FEM) software COMSOL Multiphysics to simulate the heat transport through colloidal assemblies. For this, the crystalline (0 and 100% large particles) and amorphous particle structures (size ratios: 0.54, 0.8, and 0.9) obtained from the MD simulations were used. The particle sizes were adjusted such that the contact areas between adjacent spheres are comparable to the experimental values. All particles were fused into one object and, thus no thermal resistance between the particles was considered. Due to the limited physical memory (128 GB), the system size has been reduced to a feasible size. Therefore, three cubes with around 100 particles were cut out of each invested colloidal assemblies. The mesh size was as chosen to ensure mesh size independent results (see Figure 13.14c). A temperature difference was imposed to two opposite surfaces of the cubes. Since all remaining surfaces were kept thermally insulating, purely one-dimensional thermal transport is obtained. The simulation was conducted along all three directions of the cube (x, y, and z). Using Fourier's law (see Equation 13.4), the effective thermal conductivity of the assembly is calculated from the computed heat flux, normal to the isothermal faces. Figure 13.15 demonstrates that three cubes are sufficient to calculate a meaningful average value. Furthermore, heat flux streamlines have been plotted to visualize the heat path length through the different colloidal assemblies. The length of the streamlines (L_{SL}) has been normalized by the total length of the cube (L_C) to calculate the relative streamline length (L_{SL}/L_Cin %). Figure 13.15 demonstrates that three cubes are sufficient to calculate a meaningful average value. Furthermore, heat flux streamlines have been plotted to visualize the heat path length through the different colloidal assemblies. The length of the streamlines (L_{SL}) has been normalized by the total length of the cube (L_C) to calculate the relative streamline length (L_{SL}/L_C in %).

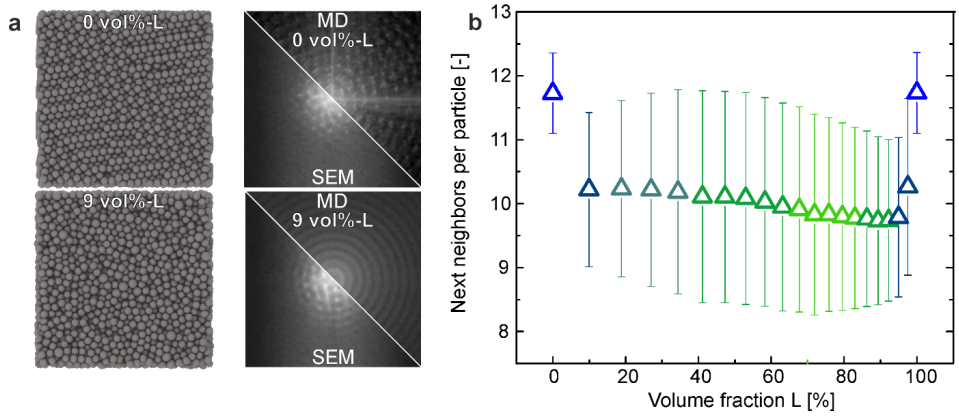


Fig. 13.7.: Side views received from MD simulations and number of next neighbors per particles. (a, left) typically obtained particle assemblies from MD studies; (a, right) FFT comparison between experiment and simulation. (b) Average next neighbors per particle vs. volume fraction of large particles. Color code indicates ordered (blueish) and disordered (greenish) assemblies.

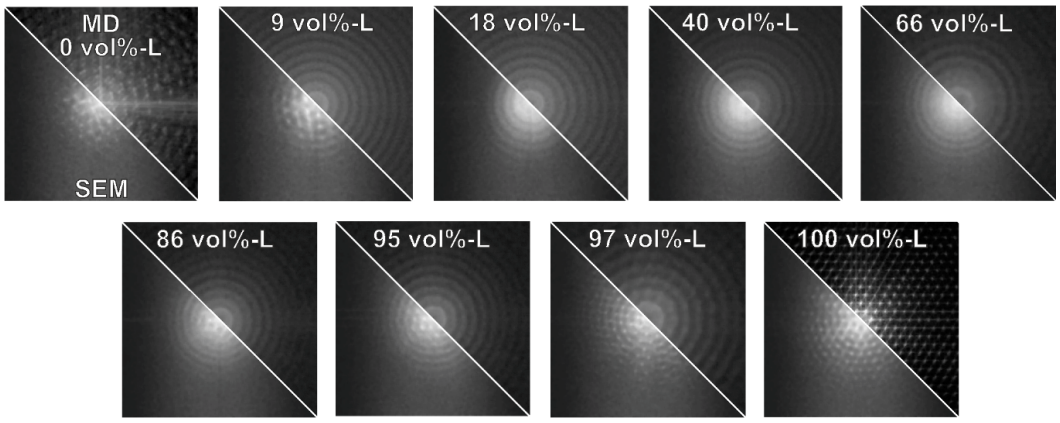


Fig. 13.8.: FFT comparison between the particle structures received from MD simulations and SEM side view images. Particularly at low and high mixing ratios a partial crystallinity can be inferred from the experimental SEM images. It indicates an easier colloidal glass formation in the MD simulation compared to the experimental conditions.

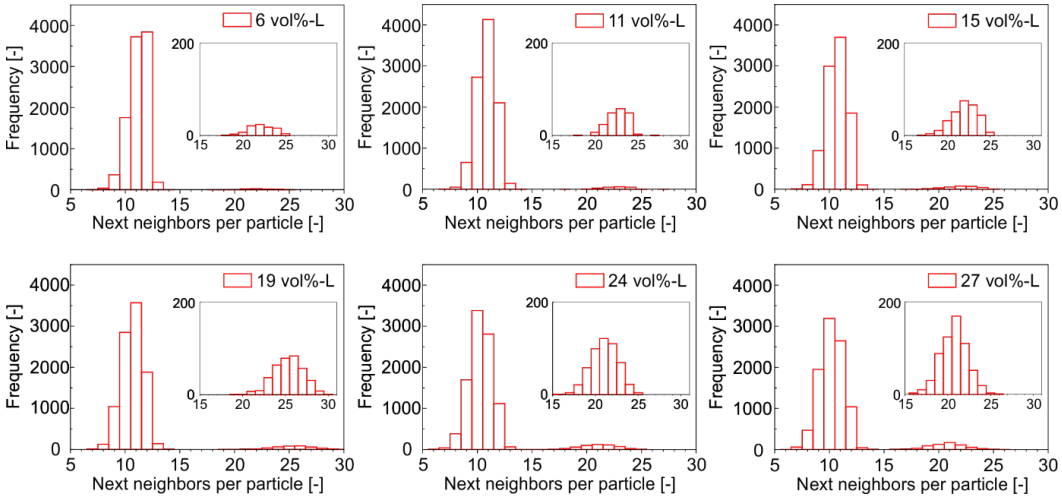


Fig. 13.9.: Histograms of the next neighbors per particle for a size ratio of 0.54. Various particle volume ratios close to the minimum thermal conductivity (19 vol%-L) are compared. All assemblies have the same average number of next neighbors ($\sim 10-11$), but only the assembly with 19 vol%-L shows a second fraction with the highest number of next neighbors ($20-30$). Thus, this assembly is expected to have the lowest thermal conductivity.

Contact neighbor analysis

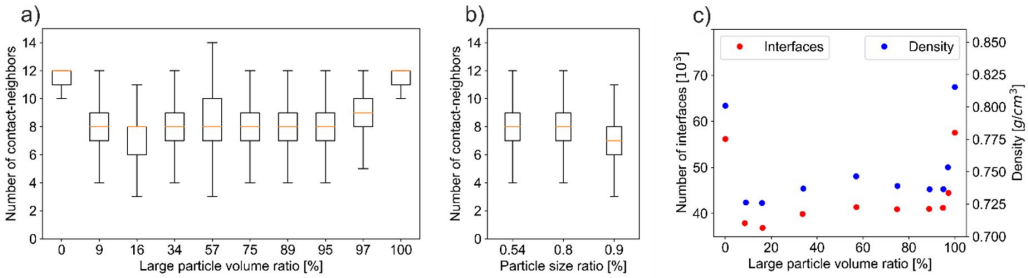


Fig. 13.10.: a) Evaluation of the number of contacting next neighbors. An almost constant average number at ~ 8 is found with a slight tendency towards fewer contact neighbors for 16 vol%-L. b) Contact neighbor analysis for particle mixture of ~ 20 vol%-L with different size ratios. c) Total number of interparticle contact areas and effective density in the entire simulation volume.

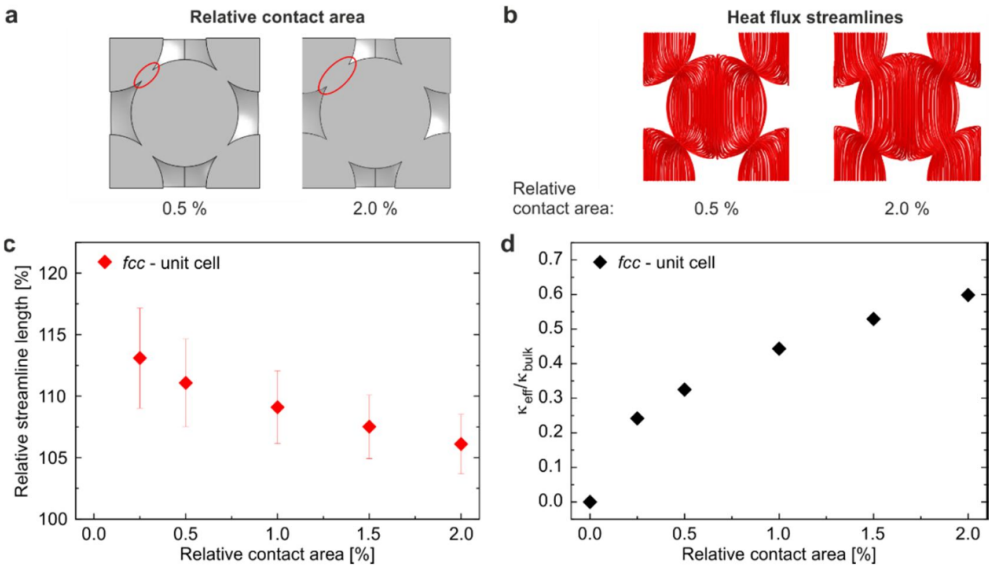


Fig. 13.11.: a) Ideal face-centered cubic colloidal crystal with increasing interparticle contact area. b) Heat flux streamlines corresponding to a small (0.5 %) and large (2.0 %) contact area. The streamlines bend more strongly in case of 0.5 % contact area. c) Correlation between the interparticle contact area and the streamline length. The relative contact area is the ratio between the circular connection between adjacent particles and the surface area of the complete particle. The total contact area takes the number of next neighbors into account (here 12 for the fcc structure). d) Correlation between the reduced thermal conductivity (relative to the corresponding bulk material) and the relative contact area.

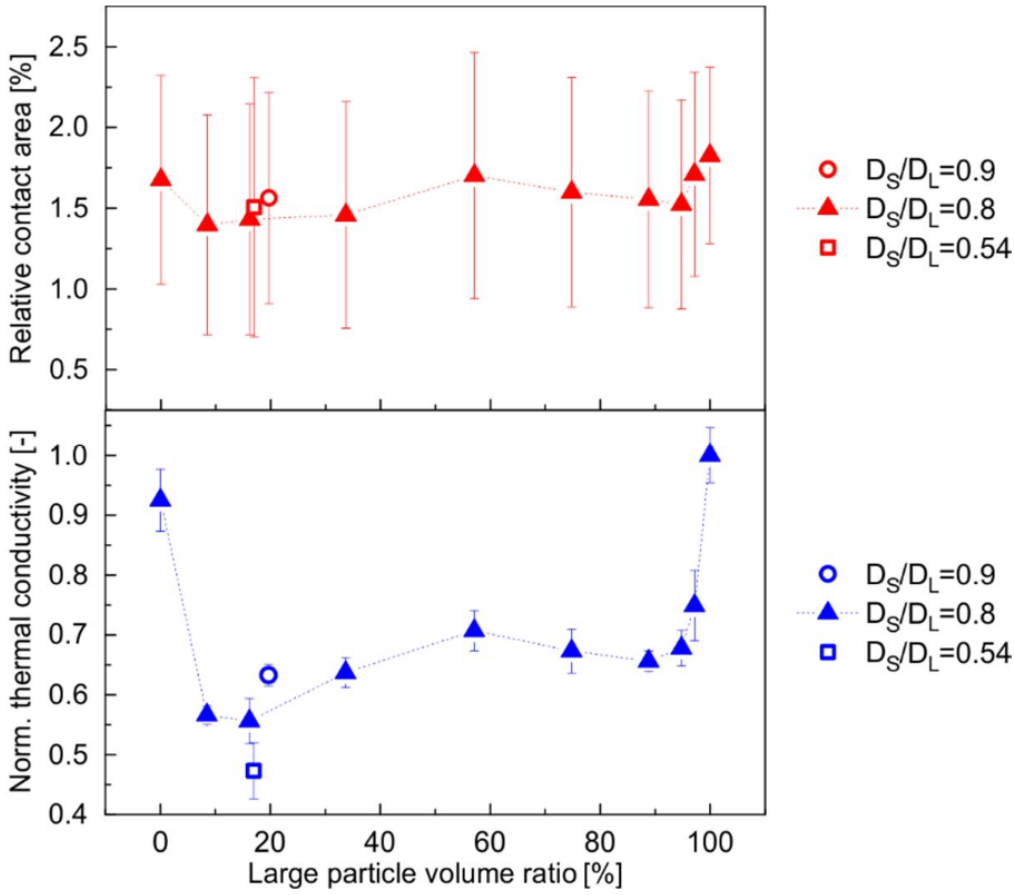


Fig. 13.12.: Dependency of normalized thermal conductivity (bottom) and relative contact area (top) on the particle mixing ratio.

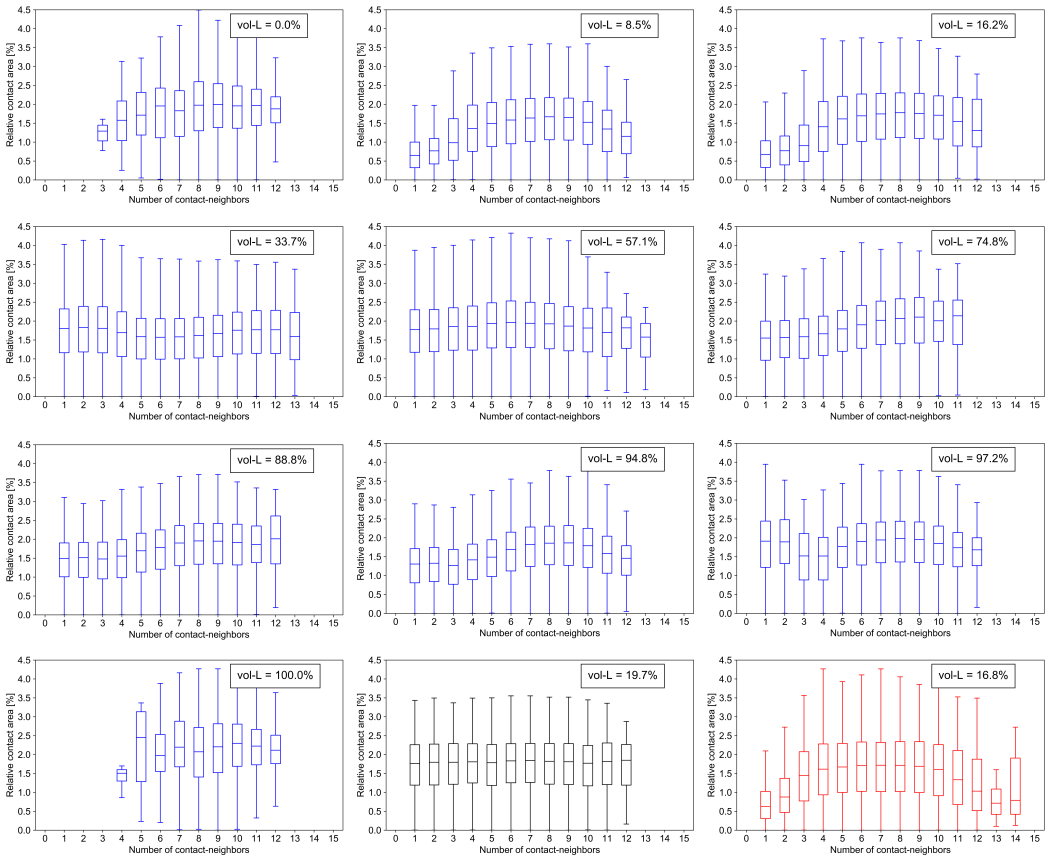


Fig. 13.13.: Statistic evaluation of the influence of contacting neighbors on the relative contact area. The box plots give an impression on the average contact are and the width and symmetry of relative contact area distributions. The individual bars at each number of contact neighbors does not include any information on the number of occurrences. The data of size ratio 0.8, 0.9, and 0.54 are colored in blue, black, and red, respectively.

Figure 13.11 demonstrates the influence of the particle contact area on the streamline length and the effective thermal conductivity for an ideal fcc crystal. Figure 13.12 shows the average contact area in dependence of the mixing ratio and relates it to our experimental findings. Figure 13.11 shows that contact area and streamline length indeed depend on one another. A strong reduction, however, is only found for relative small contact areas $< 0.5\,\%$, which are rarely found in our experiment and simulation. The interparticle constriction adds an additional possibility to increase the thermal path length through the particulate structure. Yet, based on the shown results, we conclude that the changes of the particle contact area play a secondary role, compared to the increase in thermal path length caused by the disordered structure: a) The relative contact area in our FEM simulation boxes changes in a range from $1.44\,\%$ to $1.83\,\%$, which can account for a maximum of about $10\,\%$ thermal conductivity reduction, when comparing to an fcc reference system. This is much less compared to our findings, when a disordered structure evolves. b) When looking at the two shallow minima at about 16 vol%-L and 89 vol%-L mixing ratio,

these turn out to possess rather comparable average contact areas of about 1.5 %. The analysis of the contact area distribution (box plots in Figure 13.13) shows a bell-shaped distribution for the 16 vol%-L case and a decrease towards fewer contact particles for the 89 vol%-L case.

The majority of the other mixing ratios exhibit a flat distribution of relative contact areas. (One has to bear in mind that this representation does not give any information on how many particles are present at the respective number of contact neighbors). We assign the more pronounced drop in relative contact area of the 16 vol%-L sample to the presence of large particles jammed in a sphere of smaller particles. Hence, the extension of the thermal path length is not only caused by the disordered structure, but additionally aided by an increased constriction around these defect sites. c) The effect of small particle jamming around larger central spheres becomes even more obvious, when looking at the influence of the size ratio at comparable volume ratios (16-20 vol%-L). The bell-shape of the histogram becomes more pronounced for a decreasing size ratio and shows particularly small relative contact areas for very small and high numbers of contacting neighbors. More similar particle sizes recover a very flat histogram distribution. This is associated with the lack and presence of a high amount of next neighbor spheres, respectively. Consequently, the extension of the thermal path length caused by the introduction of additional next neighbor particles is supported by a stronger constriction around these structures.

Finite element modeling Fourier's law is used to calculate the effective thermal conductivity κ_{eff}:

$$\kappa_{\text{eff}} = \frac{Q/A}{\Delta T/h} \tag{13.4}$$

Q is the heat flow rate normal to the cross-sectional area A of the cube, ΔT the temperature difference, and h the cube length.

Tab. 13.2.: Specific heat capacity, density, and thermal conductivity of polystyrene used in the FEM simulations

Specific heat capacity $\left[\text{Jg}^{-1}\text{K}^{-1}\right]$	Density $[\text{g cm}^{-3}]$	Thermal conductivity $\left[\text{mWm}^{-1}\text{K}^{-1}\right]$
1.25	1.05	150

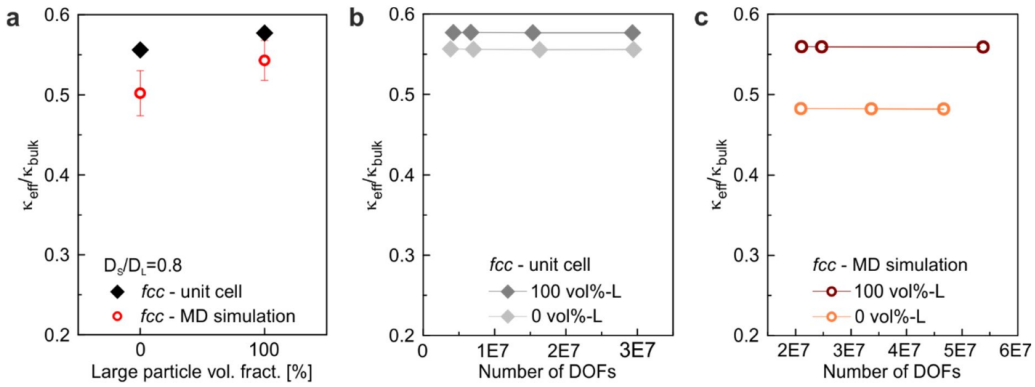


Fig. 13.14.: (a) Comparison of the effective thermal conductivity divided by the bulk thermal conductivity of polystyrene obtained for an ideal fcc unit cell and the fcc structure from MD simulation. The average particle contact area of the fcc structure from MD simulation was used to create the unit cell. The slight reduction of the MD simulation compared to the ideal unit cell originates from line and point defects, which are not present in the ideal case. (b, c) The effective thermal conductivity is independent of the number of DOFs and, thus of the mesh size. (c) The data are exemplarily plotted for one position in the fcc structure from MD simulation, respectively.

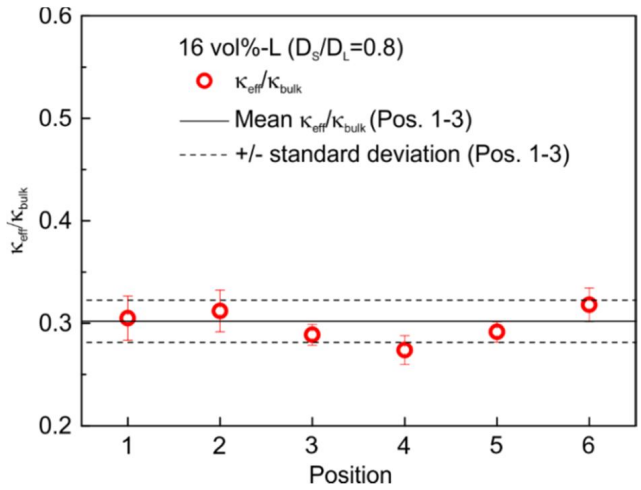


Fig. 13.15.: Effective thermal conductivity of the particle assembly with 16 vol% large particles (size ratio 0.8) obtained for six cubes cut out of different positions of the amorphous particle structure (red circles). The error bars arise from the three different simulation directions per cube (x, y, and z). The black solid line represents the mean effective thermal conductivity calculated from the first three values, the dashed line is the corresponding standard deviation. This plot shows that three positions are sufficient to calculate a meaningful average value.

References

[1] J. W. Goodwin, J. Hearn, C. C. Ho, and R. H. Ottewill, „Studies on the preparation and characterisation of monodisperse polystyrene laticee,“ *Colloid and Polymer Science*, vol. 252, no. 6, pp. 464–471, 1974.

[2] R. H. Ottewill and J. N. Shaw, „Studies on the preparation and characterisation of monodisperse polystyrene latices,“ *Kolloid-Zeitschrift und Zeitschrift für Polymere*, vol. 218, no. 1, pp. 34–40, 1967.

[3] L. Dusza, „Combined solution of the simultaneous heat loss and finite pulse corrections with the laser flash method,“ *High Temperatures-High Pressures*, vol. 467, pp. 27–28, 1995.

[4] W. J. Parker, R. J. Jenkins, C. P. Butler, and G. L. Abbott, „Flash method of determining thermal diffusivity, heat capacity, and thermal conductivity,“ *Journal of Applied Physics*, vol. 32, no. 9, pp. 1679–1684, 1961.

[5] S. Plimpton, „Fast parallel algorithms for short-range molecular dynamics,“ *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, 1995.

[6] A. Stukowski, „Visualization and analysis of atomistic simulation data with OVITO–the open visualization tool,“ *Modelling and Simulation in Materials Science and Engineering*, vol. 18, no. 1, p. 015 012, 2009.

[7] S. Safran, *Statistical Thermodynamics Of Surfaces, Interfaces, And Membranes*, Reprint. New York: Avalon Publishing, 2003.

14

Tunable thermoelastic anisotropy in hybrid Bragg stacks with extreme polymer confinement

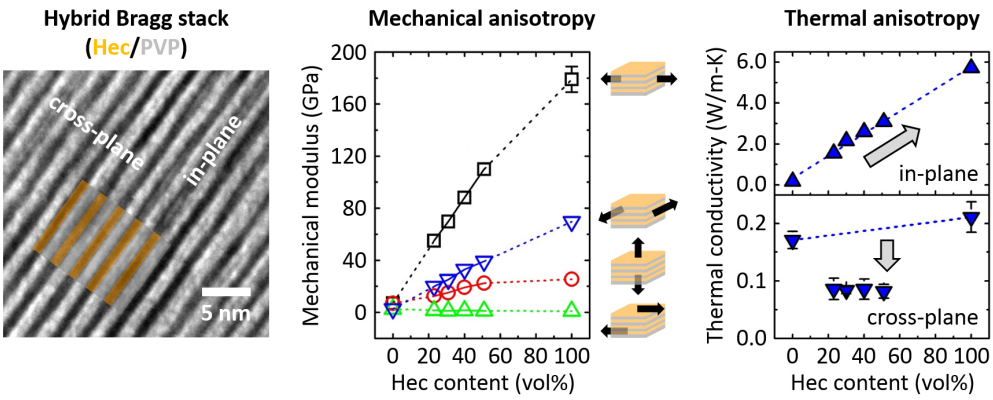
Zuyuan Wang,[1] Konrad Rolle,[1] Theresa Schilling,[2,3] Patrick Hummel,[2,3] Alexandra Philipp,[2,3] Bernd A. F. Kopera,[2,3] Anna M. Lechner,[2,3] Markus Retsch,[2,3] Josef Breu,[2,3] and George Fytas[1,4]

[1] Max Planck Institute for Polymer Research, Ackermannweg 10, 55128 Mainz, Germany

[2] Bavarian Polymer Institute, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

[3] Department of Chemistry, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

[4] Institute of Electronic Structure and Laser, F.O.R.T.H., 70013 Heraklion, Greece



Published in *Angewandte Chemie Int. Ed.*, **2019**, DOI: 10.1002/anie.201911546
Reproduced with permission from John Wiley and Sons

14.1 Abstract

Controlling thermomechanical anisotropy is important for emerging heat management applications such as thermal interface and electronic packaging materials. Whereas many studies report on thermal transport in anisotropic nanocomposite materials, a fundamental understanding of the interplay between mechanical and thermal properties is missing, due to the lack of measurements of direction-dependent mechanical properties. In this work, exceptionally coherent and transparent hybrid Bragg stacks made of strictly alternating mica-type nanosheets (synthetic hectorite) and polymer layers (polyvinylpyrrolidone) were fabricated at large scale. Distinct from ordinary nanocomposites, these stacks display long-range periodicity, which is tunable down to angstrom precision. A large thermal transport anisotropy (up to 38) is consequently observed, with the high in-plane thermal conductivity (up to 5.7 $^{\mathrm{W}}/_{\mathrm{mK}}$) exhibiting an effective medium behavior. The unique hybrid material combined with advanced characterization techniques allows correlating the full elastic tensors to the direction-dependent thermal conductivities. We, therefore, provide a first analysis on how the direction-dependent Young's and shear moduli influence the flow of heat.

14.2 Introduction

Heat management is crucial in many applications important for fueling the growth of our technology-driven society. It needs to address not only very small length scales to dissipate the heat produced, for example, by electronic circuits, but also very large length scales to realize air conditioning, for instance, for commercial buildings. The ubiquity of heat makes it obvious that heat management is a key-technology to realize international long-term goals regarding global warming. Controlling the elusive flow of heat is a complex challenge across multiple materials, length scales, and ultimately devices. This results in stringent requirements for directional control over the heat flux based on advanced material design. Whereas heat transport represents an effective, far-field phenomenon, it is decisively governed by the material structure[1]–[3] and chemistry[4],[5] on the microscale. Extreme phenomena of both heat dissipation and thermal insulation have been demonstrated in nanostructured and hybrid materials. For heat dissipation, surprisingly high thermal conductivities have been reported for one-dimensional (1D) fibers comprising synthetic[6] and natural polymers.[7],[8] For thermal insulation, unusually low thermal conductivities have been shown for (disordered) stacks of two-dimensional (2D) materials.[9],[10] Extremely efficient anisotropic thermal insulation materials have been demonstrated with various mixtures of polymers and nanoparticles or 2D materials.[11]–[13]

The combination of inherently different materials, such as soft and hard matter, is attractive, as new properties, deviating from those based on the simple linear interpolation, could emerge. This is often accompanied by improved processability, which is aided by the complementary properties of the constituent components. For instance, the soft component can serve as a binder to enable fabrication of large-area, thin films of an otherwise brittle, hard component. On the contrary, the expected effective material properties, such as mechanical reinforcement, optical transparency, and electrical or thermal conductivity, have been often found inferior to the high expectations. The reason for such shortcomings is that the nanocomposite structure, particularly the soft-hard interface, is poorly controlled. Furthermore, although many characterization techniques, such as tensile testing, indentation, and abrasion tests, are capable of assessing engineering properties, they are unsuitable for directly identifying and quantifying anisotropies or microscopic contributions to the effective properties.

Nevertheless, hybrid systems have been reported to drastically alter the materials' thermal transport properties,[6]–[13] depending on the geometry, dimensionality, crystallographic symmetry, and confinement. Interestingly, layered structures inherently exhibit structural anisotropy, a feature that can be detrimental or desirable depending on the application.[14] In particular, when polymer films are filled with nanosheets of huge aspect ratio, the resulting nanocomposite properties ought to be exceedingly anisotropic. Yet, only effective material properties such as electrical or thermal conductivity have been reported in a direction-dependent manner. Direction-dependent mechanical properties, which fundamentally translate into thermal transport properties are still missing. Strong anisotropies in hybrid materials are preferentially achieved at small stacking periodicities[10] or by combining components with a large property contrast.[15]

For a thorough characterization of such nanosheet/polymer stacks (also known as "nacre-mimics"[16]–[19]), macroscopically oriented and homogeneous systems are paramount. Such ideal model system should also exhibit translational crystallographic symmetry, tunability, and strong anisotropy. Direction-dependent studies benefit significantly from the availability of various light scattering methods, rendering a transparent filler such as the synthetic clay hectorite with a mica-type structure desirable. For fundamental investigations of elastic properties, Brillouin light spectroscopy (BLS) has established itself as a technique of choice, as it allows for microscopic observations of high frequency (GHz) dynamics, at which viscoelasticity effects are usually negligible.[20],[21] On the other hand, lock-in thermography and photoacoustic techniques have been proven reliable in accessing the in-plane and cross-plane thermal conductivities of thin films.[22],[23] Here, we show for the first time the complete mechanical properties of clay/polymer Bragg stacks that are fabricated using a uniquely defined, scalable spray-coating process meeting all aforementioned

specifications of a suitable model system. We, therefore, introduce 1D hybrid Bragg stacks based on nacremimetic clay/polymer with small stacking periods and large property contrast. These Bragg stacks are scalable in both lateral extension and thickness, and they are macroscopically oriented. The fully controlled microstructure allows a detailed orientation dependent characterization of the thermal and mechanical properties. We couple the thermal and mechanical analyses to achieve an in-depth understanding of the interplay between the thermal conductivities and mechanical moduli in a directiondependent manner. The extreme confinement of polymer between the clay sheets further prompts a question regarding the validity of continuum mechanics that we also address. The combination of unique hybrid materials and advanced characterization techniques provides an unprecedented insight into the physics of direction-dependent nanomechanical and thermal transport properties in strongly anisotropic materials with polymer confinement.

14.3 Results and Discussion

14.3.1 Hybrid Bragg stacks with extreme polymer confinement

The Bragg stacks comprise synthetic clay sodium fluorohectorite (Hec, $[Na_{0.5}]^{inter}[Mg_{2.5}Li_{0.5}]^{oct}[Si_4]^{tet}O_{10}F_2$) and polyvinylpyrrolidone (PVP, $M_w = 40000\,g/mol$). Like layered titanates[24] and antimony phosphates,[25] Hec belongs to a handful of compounds showing a rare phenomenon of osmotic swelling.[26] In contrast to mechanical exfoliation by e.g. sonication in the liquid phase,[27] osmotic swelling is a thermodynamically favored, repulsive process,[28] allowing for complete and gentle delamination that preserves the diameter of the parent crystals. In general, exfoliation describes the process of slicing tactoids into thinner stacks, whereas by delamination, the layered material is exfoliated to the level of individual single nanosheets.[29] For Hec, nanosheets with a thickness of 10 Å and a median diameter of 20 µm (Figure 14.5) are obtained by simply immersing the material into deionized water.[30] Phase purity and a homogeneous charge density guaranteeing a uniform intracrystalline reactivity are prerequisite for such a well-controlled delamination. For Hec this is achieved by long-term annealing, while less uniform natural or other synthetic clays commonly applied for nacre-mimics comprise mixtures of auxiliary minerals, mono-, few- and multilayer stacks.[30] Because of the large aspect (diameter to thickness) ratio, polar rotation of the nanosheets in suspension is hindered, leading to parallel nanosheets after osmotic swelling. Even dilute (< 1 vol%) suspensions of Hec represent nematic phases.[31] The parallel pre-orientation of adjacent nanosheets in the highly swollen dispersion is indispensable for the fabrication of homogenous and periodic Bragg stacks via spray coating. Similar to titanate nanosheets,[32] Hec

nanosheets adopt this cofacial arrangement due to strong electrostatic repulsion with inter-nanosheet distances exceeding 50 nm. Polymers can easily diffuse into these spacious galleries. By mixing Hec suspensions with varying aliquots of an aqueous PVP solution, we obtained perfectly homogeneous, nematic dispersions, as evidenced by small-angle X-ray scattering (SAXS) measurements (Figure 14.6).

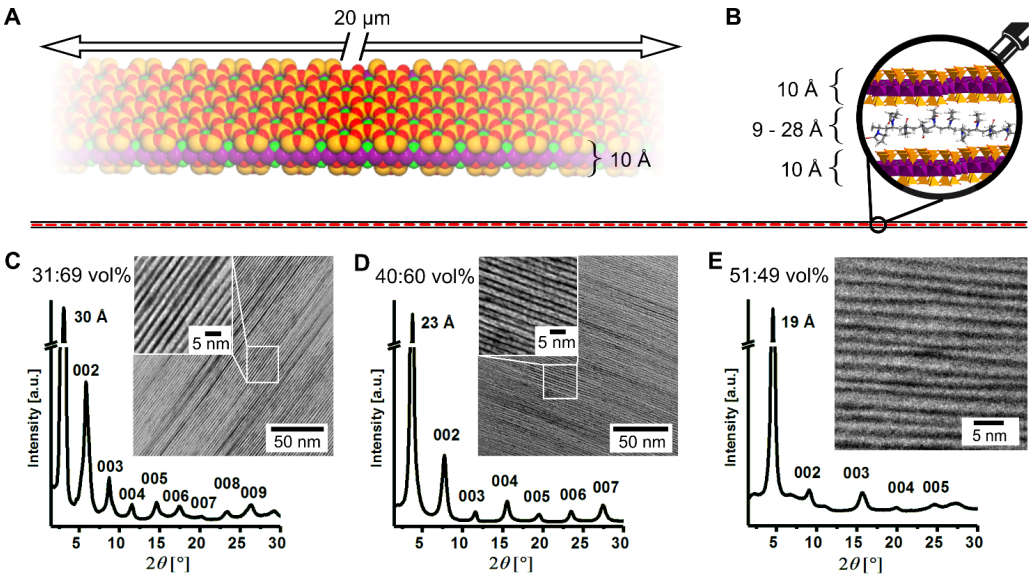


Fig. 14.1.: Schematic and microscopic images of ultra-anisotropic and extremely confined Hec/PVP Bragg stacks. (A) Space-filling model of one single Hec nanosheet emphasizing the anisotropy of the nanosheet and the corrugation of the clay nanosheet allowing for interdigitation with PVP. (B) True to scale schematic of the pronounced structural anisotropy. The ultra-high-aspect-ratio nanosheets stretch from left to right and have lateral dimensions much larger than the length of the PVP polymer chains. The gallery height is on the order of magnitude of the molecular dimensions. (C-D) XRD patterns of Hec31/PVP69 and Hec40/PVP60 (defect-free materials) showing intense 001-reflections and a rational series of basal reflections up to the ninth order. The cross-sectional TEM images show exceptionally periodic homogeneity of these hybrid films over large length scales. (E) XRD patterns and cross-sectional TEM image of Hec51/PVP49 displaying a random stacking of two gallery heights.

Through spray coating of dilute nematic mixtures of high-aspectratio Hec nanosheets with PVP (1-2 wt% total solid content, see Section 14.7.1) highly coherent Bragg stack films with tunable gallery spacings are fabricated.[33] The transverse flexibility of clay monolayers[34] and their large aspect ratio are essentials assuring the high degree of precision obtained in the self-assembly.[35] Both, all nanosheets and the macroscopic film are aligned parallel to a polyethylene terephthalate substrate. The microscopic orientation of the Hec nanosheets prescribes the macroscopic film orientation, which is prerequisite for the direction-dependent measurements. The macroscopic film orientation is, consequently, equivalent to the microscopic polymer/clay direction and allows using far-field and integrating characterization techniques to reveal direction-dependent properties. After drying, self supporting

hybrid films with lateral extensions of several square centimeters are peeled off the substrate and used in the BLS and thermal conductivity measurements. Only by generating a nematic phase consisting of a homogeneous mixture of large aspect ratio and flexible nanosheets allows for fabrication of large area, self-standing 1D single crystals referred to in literature as Bragg-stacks or smectic films.[35] Furthermore, appropriate processing like spray coating fostering the thermodynamic equilibration of the hybrid structure during drying has to be employed.

In total, we prepared six samples: pure polymer, pure Hec, and four hybrid Bragg stacks, which are denoted as Hec0/PVP100, Hec100/PVP0, Hec23/PVP77, Hec31/PVP69, Hec40/PVP60, and Hec51/PVP49, respectively. Here, the numbers indicate the volume fractions (vol%) of Hec and PVP, as confirmed by thermogravimetric analysis (Table 14.1 and Figure 14.7).

The Hec surface is corrugated (Figure 14.1A) allowing for interdigitation and anchoring of PVP chains. Such interdigitation has been documented for intercalated molecular moieties, where structures based on single crystal data refinement are available.[36],[37] An in-scale impression of the ultra-high aspect ratio provided by the Hec nanosheet gallery is shown in Figure 14.1B, where the length of the line corresponds to the typical lateral size of a clay nanosheet, and the thickness of the line to the height of a Hec/PVP/Hec layer. The magnifying lens highlights the extreme polymer confinement in the cross-plane direction. The perfect homogeneous arrangement of Hec nanosheets and PVP is demonstrated by TEM and SEM images over different dimensions (Figure 14.1C-E, Figure 14.11). Note that the lateral dimensions of the Hec nanosheets are much larger than the typical persistence and even contour lengths of the PVP chains. While for the polymer chains the Hec nanosheet confinement appears infinite, at the length scale of the Bragg-stack films extending over tens of centimeters, they are of course finite. At the magnification where single 1 nm thick nanosheets are observable (Figure 14.1C-E, Figure 14.9), the occurrence of nanosheet edges is very rare (fewer than one per 2500 nm^2. Careful inspection, however, reveals few (Figure 14.10) of these nanosheet edges. The clay nanosheets show in-plane crystalline order resembling the structure of mica. While mica possesses 3D crystalline order, our nanocomposite films belong to the transversely isotropic symmetry class, because the adjacent Hec nanosheets are positioned randomly in the lateral direction. However, all hybrid films show translational crystallographic symmetry along 001 (the cross-plane direction), as indicated by several orders of Bragg reflections (Figure 14.1C-E and Figure 14.9). By varying the PVP content, the basal spacing was tuned in the range from 19 to 38 Å, leading to PVP layer thicknesses ranging from 9 to 28 Å. For all samples, the gallery height is, therefore, significantly smaller than the PVP chains' radius of gyration (R_g, PVP $\approx$ 15 nm[38]), implying strong polymer confinement.

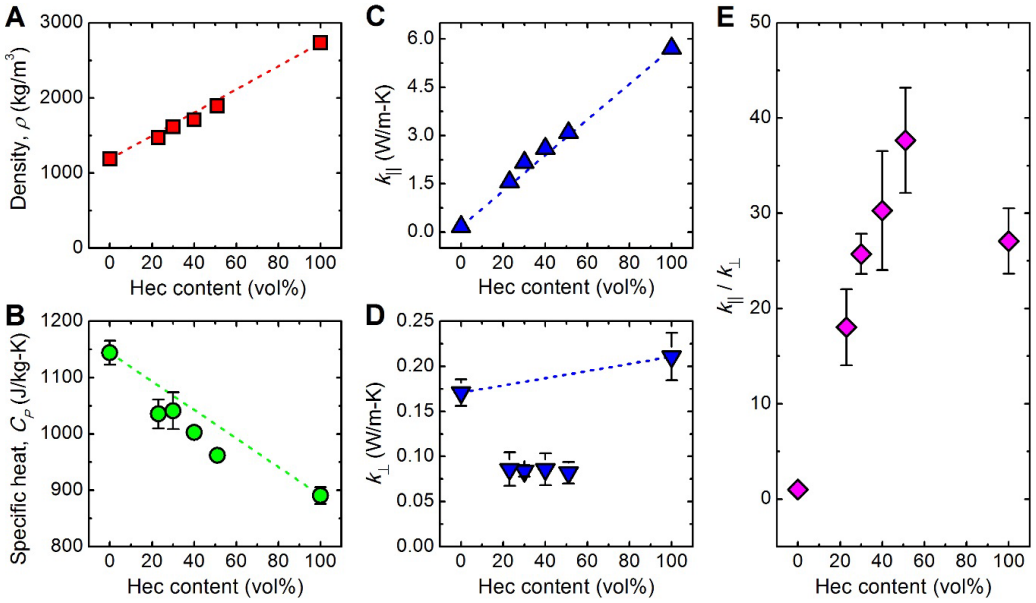


Fig. 14.2.: Direction-dependent thermal conductivities of Hec/PVP hybrid Bragg stacks. (A) Film density, (B) specific heat, and effective (C) in-plane and (D) cross plane thermal conductivities, as a function of the Hec volume fraction. The red, green, and blue dashed lines in (A)-(D) show linear trends based on a simple mixing model, $A(x) = (1-x)A(0\%) + xA(100\%)$, where A represents ρ, C_{p}, $k_{\|}$, or $k_{\perp}$, and x denotes the Hec volume fraction. (E) Ratio of the in-plane to cross-plane thermal conductivities vs. the Hec volume fraction. For clarity, error bars smaller than the symbol size are not shown.

In contrast to known nanocomposite films,[17],[39] Hec and PVP are thermodynamically miscible over a wide range of compositions. This miscibility is a prerequisite for tuning the basal spacing over a wide range with angstrom precision, which typically is only observed when vapour-phase deposition techniques are applied.[9] The miscibility is also reflected in an agreement of the basal spacing observed by X-ray diffraction (d_{XRD}) with the nominal values calculated based on the Hec and PVP volume fractions (d_{nominal}, Table14.1). To the best of our knowledge, our Hec/PVP Bragg stack films are the first of its kind showing such an agreement. Because the polymer confinement, however, is getting to the point where the gallery height is on the order of the size of an individual polymer chain, it is not possible to vary the gallery height continuously but only in incremental steps that relate to the diameter of the polymer chain. Consequently, only discrete polymer volume fractions lead to essentially defect-free Bragg stacks, as seen in Hec40/PVP60 and Hec31/PVP69. XRD patterns reflect this with a rational 001-series, where the average basal spacing (d_{XRD}) calculated from individual reflections shows a low coefficient of variation with the reflection peaks being sharp and intense (Table 14.1). The two gallery heights of the two defect free hybrid materials (1.3 nm for Hec40/PVP60 and 2.0 nm for Hec31/PVP69) might be attributed to the elliptical nature of PVP chains with principle axes of 1.0 nm and 1.3 nm (Figure 14.8A). The observed gallery heights

correspond to a monolayer with the longer principal axis (Figure 14.8B) oriented perpendicular to the Hec nanosheets and a bilayer with the longer principle axis (Figure 14.8C) lying in the plan of the Hec nanosheets, respectively.

In the two cases where the volume ratios do not happen to match (Hec23/PVP77 and Hec51/PVP49), the miscibility is nevertheless assured at small length scale by random interstratification of two gallery heights (Figure 14.1E; transmission electron microscopy (TEM) close-up), and the coefficient of variation of the 001-series increases[40] (Table 14.1) with the reflection peaks being less intense (Figure 14.1E and Figure 14.9).

14.3.2 In-plane and cross-plane thermal conductivities

The in-plane and cross-plane thermal conductivities of the Hec/PVP hybrid Bragg stacks were characterized by lock-in thermography and photoacoustic measurements,[22],[23],[41]respectively. Since the density, ρ, and specific heat, C_p, are prerequisites for the thermal conductivity analysis, they were also determined experimentally by using helium pycnometry and differential scanning calorimetry (DSC) (Section 14.7.4), respectively. As the Hec volume fraction increases from 0 to 100%, the density increases from 1190 to 2730 $\mathrm{kg/m^3}$ (Figure 14.2A). This is well captured by a volume fraction-based mixing model (dashed line in Figure 14.2A). Correspondingly, the specific heat decreases from 1140 to 890 $\mathrm{J/kgK}$ (Figure 14.2B), which also follows the prediction by an effective medium model (Figure 14.12B). Both analyses indicate that despite the extreme polymer confinement the properties of the hybrid stacks could be described by linearly interpolating the properties of the two bulk constituents. The polymer confinement, however, leads to a significant increase in the glass transition temperature (T_g) of PVP, with no discernable T_g below 250 °C even at the lowest Hec composition (Figure 14.12A). Expectedly, the thermal conductivity of the Bragg stacks strongly depends on the direction. The in-plane thermal conductivity achieves its maximum, $k_{\parallel,\mathrm{max}} = 5.71\,\mathrm{W/mK}$, in Hec100/PVP0, which is even higher than typical in-plane thermal conductivities of natural micas (Figure 14.4A).[42] The lower end is given by the isotropic thermal conductivity of Hec0/PVP100, i.e., $k_{\parallel,\mathrm{min}} = 0.17\,\mathrm{W/mK}$ (only determined by photoacoustic characterization). The four hybrid Bragg stacks have in-plane thermal conductivities between these limiting values following a parallel mixing model (Figure 14.2 C).[43] This is again surprising, as it implies that the confinement of PVP has no effect on the in-plane thermal conductivity of the Hec/PVP hybrid stacks compared to bulk PVP. The cross-plane thermal conductivity exhibits a broad minimum at $k_\perp \approx 0.09\,\mathrm{W/mK}$, which is comparable to previously reported data for organoclay laminates.[10] The deviation of the cross-plane thermal conductivities from an effective medium behavior (Figure 14.2D, dashed line) could be attributed to the Hec/PVP interfaces,

which are the dominating contributors to the cross-plane thermal resistance, as discussed below. The thermal conductivity anisotropy, $^{k_{\parallel}}/k_{\perp}$, depends strongly on the hybrid composition, attaining a maximum of 38 in Hec51/PVP49 (Figure 14.2E). We note that this anisotropy is exceptionally high for electrically insulating hybrid materials[44] and outperforms natural nacre by a factor of ≈ 20.[45] All in all, the structural perfection of the pure components and hybrid Bragg stacks translate into a record-high in-plane thermal conductivity and thermal transport anisotropy.

14.3.3 Anisotropic mechanical properties

The unique macroscopic orientation in the hybrid Bragg stacks allows us to track down the origin of their high thermal conductivity anisotropy by measuring their full mechanical tensors. The measurements were conducted by using BLS, which probes the phonon wave vector, q, dependent sound velocity, v, through inelastic light scattering by thermally excited, high frequency (GHz) phonons.[20],[21] Since the hybrid Bragg stacks are transversely isotropic, only q vectors in a single plane containing the symmetry axis have to be considered. For such a q vector, the direction can be denoted by α, the angle between q and the normal to the sample film, and because of symmetry α can be restricted in the range from 0° to 90°. The measurements corresponding to $\alpha = 0°$, $0° < \alpha < 90°$ and $\alpha = 90°$ were conducted in the reflection, backscattering, and transmission scattering geometries, respectively, while the polarization of the phonon mode was selected using different incident and scattered light polarization configurations (e.g., VV for quasi-longitudinal (Q-L) and quasi-transverse (Q-T) modes, and VH for a pure-transverse (P-T) mode).[46] This flexibility of accessible parameters makes BLS particularly suitable for characterizing anisotropic or crystalline structures, as demonstrated in previous experiments on mica crystals.[47] Since this work is the first one to apply the BLS technique to a hybrid Bragg stack material, we briefly outline the BLS measurement and data analysis.

Consider Hec31/PVP69 as an example. A typical BLS spectrum from the reflection geometry (inset to Figure 14.18A) displays a cross-plane longitudinal ($L_{\perp}$) mode in the VV polarization configuration (Figure 14.18B). A typical BLS spectrum from the backscattering geometry (Figure 14.3A, top-right inset) depicts a Q-L and a Q-T mode in the VV polarization configuration (Figure 14.2A) and a weak P-T mode in the VH polarization configuration (Figure 14.3A, top-left inset). Comparatively richer information exists in a typical BLS spectrum from the transmission geometry (Figure 14.3B, top-right inset). In the VV polarization configuration, the BLS spectrum features an in-plane longitudinal ($L_{\parallel}$) and a Q-L mode at a small laser incident angle, β, and an additional Q-T mode at a large β (Figure 14.3B). In the VH polarization configuration, a weak P-T mode at all β (Figure 14.3B, top-left

inset) is clearly resolved. In the transmission BLS spectra, the intensity ratio of the Q-L and Q-T peaks yields additional information (Figure 14.19B) and the Q-T mode intensity increases noticeably at higher Hec contents. By comparing the backscattering and transmission spectra, it becomes clear that the appearance of the Q-L and Q-T peaks in the latter (Figure 14.3B) results from the scattering of the laser beam internally reflected on the sample's backside.[48]

Based on the frequency shift, f, from the BLS spectrum and the phonon wave vector, q, from the momentum conservation analysis, we calculated the sound velocity along a certain q as $\nu = {}^{2\pi f}/_{|q|}$. Whereas the reflection measurements give the ν_{*+}, at $\alpha = 0°$ and the transmission measurements result in the ν_{*+}, and ν_{-+}. at $\alpha = 90°$, the backscattering measurements provide sound velocities for all the Q-L, Q-T, and P-T modes at intermediate α angles, as limited by the sample's refractive index. These direction-dependent sound velocities are reported in Figure 14.3C for Hec31/PVP69; additional data for the other samples are shown in Figure 14.20A-D. Since sound velocities are intimately related to the elastic stiffness tensor in the framework of the Christoffel equation,[49],[50] the availability of the former together with the measured sample densities (Figure 14.2A and Table 14.4) enables unique determination of the latter. For a transversely isotropic material, the elastic stiffness tensor contains five independent elastic constants (e.g., C_{11}, C_{12}, C_{13}, C_{33}, and C_{44}).[51],[52] Through χ^2 fitting,[53] we obtained the elastic stiffness constants (Figure 14.20E and Table 14.4), which allow theoretical representation of the direction-dependent sound velocities (solid lines in Figure 14.3C and Figure 14.20A-D) as well as determination of the engineering mechanical properties (Figure 14.3D and Table 14.5). In addition, we analyzed the error bars (standard deviations) of the quantities according to principles of uncertainty propagation (Section 14.7.11).

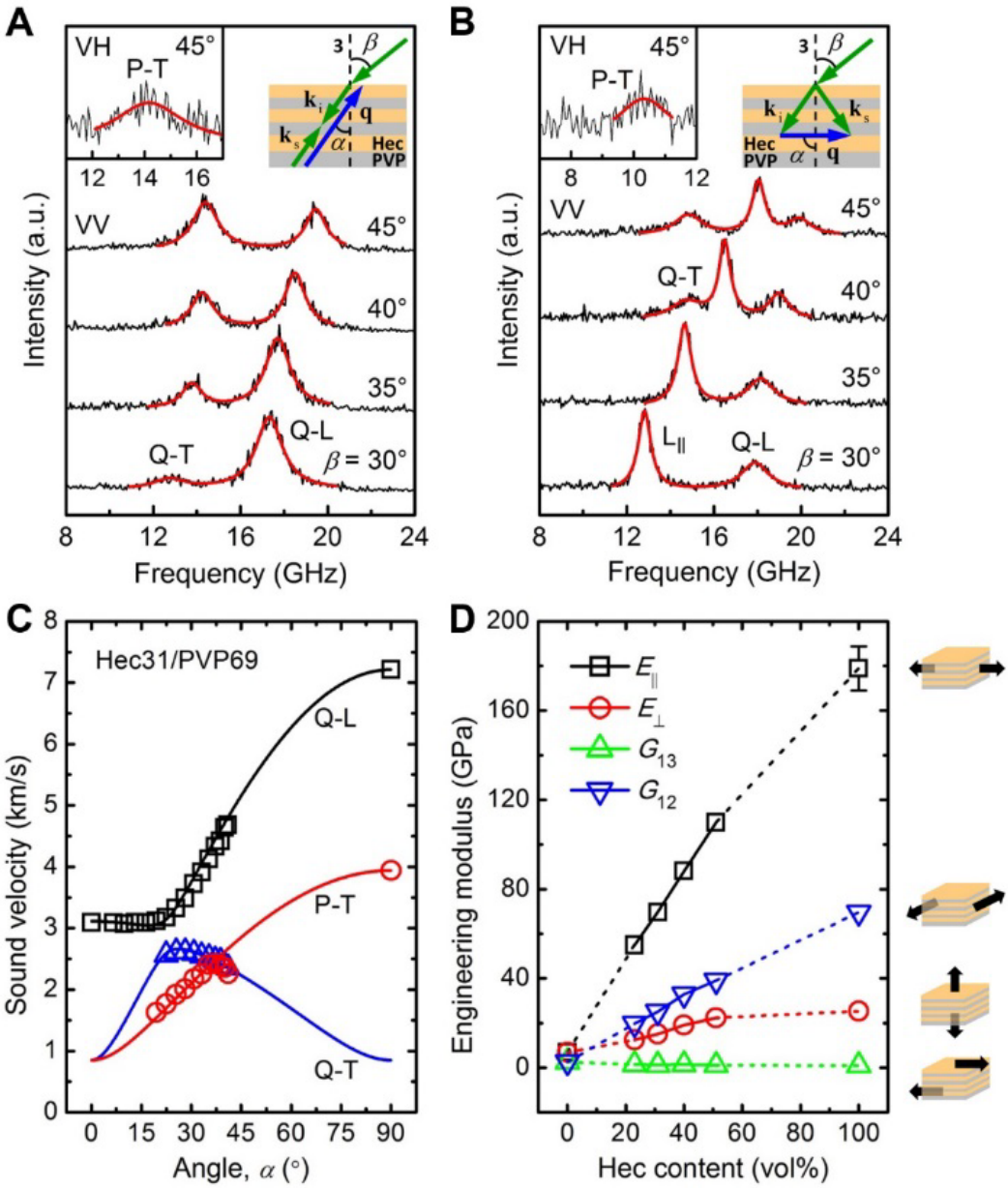


Fig. 14.3.: BLS measurements and strong mechanical anisotropy of Hec/PVP hybrid Bragg stacks. (A-B) Polarized BLS spectra (anti-Stokes side) of the Hec31/PVP69 hybrid stack film recorded in (A) the backscattering geometry with q forming a variable angle, α, with the normal to the sample film (top-right inset to (A)) and (B) the transmission geometry with the phonon wave vector, q, directed in-plane ($\alpha = 90°$; top-right inset to (B)). k_i and k_s are the wave vectors of the incident and scattered light beams, respectively. β is the incident angle of the laser beam. The quasi-longitudinal (Q-L), quasi-transverse (Q-T), and in-plane longitudinal ($L_{||}$) phonon modes are indicated in (A) and (B). The much weaker depolarized VH spectra of the pure transverse modes are shown in the top-left insets. Notice the correspondence between the Q-L and Q-T modes in (A) and those in (B). (C) Direction-dependent sound velocities of the observed acoustic phonons in the BLS spectra of Hec31/PVP69. The three solid lines indicate theoretical representations (Equations (14.6)-(14.8)) of the experimental sound velocities of the three modes. (D) Composition dependence of four engineering moduli. The moduli of the anisotropic hybrid films are extrapolated to those of the pure PVP and pure Hec films, as shown by the dashed lines. The four schematics beside (D) visualize the physical meanings of the corresponding moduli.

This analysis provides the first direction-dependent insights into the mechanical properties of hybrid Bragg stacks in general and of clay/polymer nanocomposites in particular. The Young's moduli, $E_{\parallel}$ and $E_{\perp}$, and torsional shear modulus, G_{12}, all increase with increasing Hec volume fraction. The sliding shear modulus, G_{13}, however, decreases from 2.6 GPa in Hec0/PVP100 to 1.0 GPa in Hec100/PVP0. A reduction in polymer chain entanglement upon confinement could be the cause of the decrease in G_{13}.[54] Since the elastic moduli of polymer nanocomposites depend on the specific filler-polymer and polymer-polymer interactions, a rationalization of the increase ($E_{\parallel}$, $E_{\perp}$, G_{12}) or decrease (for G_{13}) with Hec content would require computer simulations.[54] All the mechanical moduli of the Bragg stacks display an effective medium behavior, assuming values between those of the two bulk components. We point out that even though the PVP chains are strongly confined between the adjacent Hec nanosheets (note $R_{\mathrm{g,\,PVP}} > 5(d_{\mathrm{XRD}} - d_{\mathrm{Hec}})$), bulk properties (e.g., ρ_{PVP}, ρ_{Hec}) are sufficient to fully capture the BLS measurements. As expected from the structural anisotropy, the Young's moduli exhibit large differences between the in-plane and cross-plane directions (Figure 14.3D). As the Hec volume fraction increases from 0 to 100 %, the mechanical anisotropy ratio, $E_{\parallel}/E_{\perp}$, increases from 1 to 7. Concomitantly, the two characteristic Poisson's ratios, ν_{23} and ν_{24}, vary in ranges of 0.02-0.05 (nearly zero or cork-like values) and 0.34-0.41 (typical polymer values), respectively (Table 14.5). The reasonable values of the mechanical properties corroborate the validity of continuum mechanics at length scales of a few nanometers and in the presence of extreme polymer confinement.

In the last section, we summarize the new insights onto the anisotropic thermoelasticity that can be gained from this wholistic analysis. We firstly exploit the directly measured direction-dependent sound velocities, and secondly correlate the derived mechanical moduli to the direction-dependent thermal conductivities. We first apply a kinetic theory model, $k = C_{\mathrm{v}}\bar{\nu}_{\mathrm{g}}\bar{\Lambda}/3$ to estimate the average phonon mean free path $\bar{\Lambda}$; along different directions in the Bragg stacks.[55] We use $C_{\mathrm{v}} = C_{\mathrm{p}}$ and $\bar{\nu}_{\mathrm{g}} = \bar{\nu}_{\mathrm{s},\parallel} = \left(\nu_{\text{Q-L},\parallel} + \nu_{\text{Q-T},\parallel} + \nu_{\text{P-T},\parallel}\right)/3$; a similar analysis is done for $k_{\perp}$. The in-plane $\bar{\Lambda}$; strongly depends on the hybrid composition, ranging from 14 Å for Hec100/PVP0 to 2 Å for Hec0/PVP100 (Figure 14.4A). We note that these $\bar{\Lambda}$ values significantly underestimate the presence of longer ranged phonons, which are typically better described by a phonon mean free path accumulation function.[55],[56] It is well known that thermal transport involves phonons over a wide range of frequencies which have different specific heat capacities, group velocities, and mean free paths. The underestimated $\bar{\Lambda}$; in our analysis could be attributed to the overestimated $\bar{\nu}_{\mathrm{g}}$ from the BLS measurements, which mainly characterizes the propagation speed of a small fraction of the low frequency (long wavelength) phonons. These low frequency phonons carry only a negligible fraction of the overall heat. For the in-plane direction, the lateral size of the Hec nanosheets by far exceeds the average phonon mean free path. Hence, the high in-plane thermal conductivities are governed by the intrinsic

material properties, not by the presence of grain boundaries between the aligned Hec nanosheets. The complementary analysis for $k_\perp$ demonstrates a strong reduction of $\bar{\Lambda}$; down to less than 1 Å (Figure14.4B) with no discernible composition dependence along the cross-plane direction. Interfacial effects apparently dominate the thermal transport in this direction, which is better analyzed using a series resistance model (SRM, Figure 14.4C and F) as outlined by Losego *et al.*[10] The fitted value of the interfacial conductance, $G_{\mathrm{Hec/PVP}} = 89 \pm 8\,\mathrm{MW/m^2K}$ (see Section 14.7.10), falls well into the range of reported values for other inorganic/organic interfaces.[4],[57] The intercalation of PVP between the clay sheets leads to a strong reduction of the interfacial conductance, which is $G_{\mathrm{Hec/Hec}} = 291 \pm 28\,\mathrm{MW/m^2K}$ for the pure hectorite.

We next address the correlation between the anisotropic mechanical moduli and thermal conductivities. Two distinct conclusions can be drawn.

(i) In the direction parallel to the Hec nanosheets, a correlation between the thermal conductivity and all mechanical moduli is found. Along this direction the phonon mean free path is considerably shorter than the typical lateral size of a Hec nanosheet, rendering grain boundary effects insignificant. The influence of $E_\parallel$, $E_\perp$, and G_{12} on the thermal transport dominates over G_{13} since the former moduli show a direct relation to the in-plane thermal conductivity. $E_\parallel$ and G_{12} show a power scaling law close to one (0.93) between in-plane thermal conductivity and modulus ($E_\perp$ scales with 0.38). Whereas we find a clear correlation between the moduli and the thermalconductivity, we cannot deduce which change in mechanical modulus causes which effect to the thermal transport. The applicability of a simple mixing model along the parallel direction as outlined in Figure 14.2 A-C is certainly surprising in view of the strong polymer confinement effect on the glass transition (Figure 14.12).

(ii) In the direction perpendicular to the Hec nanosheets, the phonon mean free path is comparable to the periodicity of the Bragg stacks. Here, the composition dependence of the mechanical properties does not influence the reduction of the cross-plane thermal conductivity (vertical spread of the data points in Figure 14.4E). Thus, changes to the gallery height are insignificant, which indicates the overwhelming contribution of the interfacial conductance. Considering changes to the pure components we find that reducing the sliding shear modulus G_{13} decreases the cross-plane thermal transport properties of the polymer component. The reduction in $k_\perp$ of the hybrid stacks relative to pure hectorite correlates to losses in $E_\parallel$, $E_\perp$, and G_{12} that apparently counteract the increase in G_{13}.

Overall, the mechanical and thermal properties are uncorrelated along the perpendicular direction, and the thermal transport is governed by the Hec/PVP interfaces.

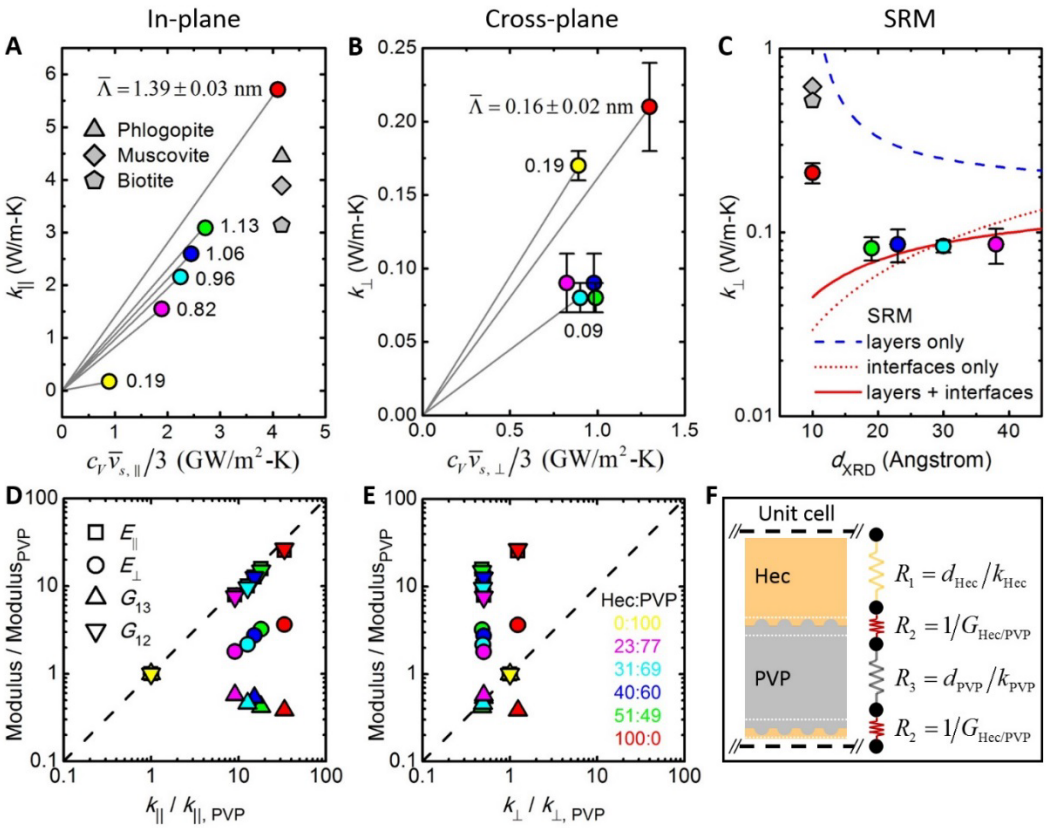


Fig. 14.4.: Analysis of anisotropic thermomechanical coupling in Hec/PVP hybrid Bragg stacks. (A) Effective in-plane thermal conductivity, $k_{\parallel}$, vs. $C_v \bar{\nu}_{s,\parallel}/3$. The yellow, purple, cyan, blue, green, and red colored symbols represent the Hec0/PVP100, Hec23/PVP77, Hec31/PVP69, Hec40/PVP60, Hec51/PVP49, and Hec100/PVP0 samples, respectively. The phlogopite data point is from our additional measurements; muscovite and biotite data points are from reference.[42] (B) Effective cross-plane thermal conductivity, $k_{\perp}$, vs. $C_v \bar{\nu}_{s,\parallel}/3$. In (A) and (B), the numbers beside the data points indicate the average phonon mean free paths, $\bar{\Lambda}$; (i.e., the slope of the gray lines). (C) Effective cross-plane thermal conductivity, $k_{\perp}$, vs. the basal spacing of the Bragg stacks. The red solid line is a fit to the experimental data based on the series resistance model (SRM) shown in (F). As a comparison, the blue dashed line considers only the thermal resistances of the Hec and PVP layers, and the red dotted line considers only the thermal resistances of the Hec/PVP interfaces. (D) Normalized mechanical moduli vs. normalized effective inplane thermal conductivity, $k_{\parallel}$. The dashed line shows a direct correlation between the two axes with a power of one. (E) Normalized mechanical moduli vs.normalized effective cross-plane thermal conductivity, $k_{\perp}$. In (D) and (E), the following values of the pure PVP film, $E_{\parallel} = E_{\perp} = 7.0\,\mathrm{GPa}$, $G_{13} = G_{12} = 2.6\,\mathrm{GPa}$, and $k_{\parallel} = k_{\perp} = 0.17\,\mathrm{W/mK}$, are used as references in the normalization. (F) A schematic of the SRM used to analyze the Hec/PVP interfacial thermal conductance.

14.4 Conclusion

In conclusion, fully delaminated hectorite platelets can be processed into hybrid Bragg stacks with unique properties, with the polymer polyvinylpyrrolidone being the intercalated second component. Such long-range 1D ordered materials become accessible by simply spray coating the desired nematic dispersions of adjusted volume fractions, which at the same time controls the periodicity of the hybrid stacks down to the angstrom level. The macroscopic lattice alignment enables the determination of direction-dependent thermoelastic properties, which we assessed by thermal transport characterization techniques and Brillouin light spectroscopy. We found a record high anisotropy between the in-plane and cross-plane thermal conductivities in clay/polymer hybrid materials. This is corroborated by the first report of direction-dependent Young's and shear moduli that are also strongly anisotropic. The effective gallery spacing, density, specific heat, and in-plane thermal conductivity were found to conform to composition-dependent simple mixing models. Despite the nanometer-level lattice periodicity and angstrom-level polymer confinement, the Christoffel-equation-based model, derived in the framework of continuum mechanics, remains applicable for determining the anisotropic elasticity. Of general relevance is the direction dependency of the way that the mechanical moduli and thermal conductivities correlate. In the in-plane direction, where grain boundaries are negligible relative to the phonon mean free path, $E_{\parallel}$, $E_{\perp}$, and G_{12} directly correlate with the in-plane thermal conductivity. In the cross-plane direction, where the phonon mean free path is comparable to the lattice periodicity, the thermal transport is governed by the clay/polymer interfaces. We are convinced that a wholistic understanding of direction-dependent thermoelastic properties will have a broad impact on important applications such as electronic packaging and thermoelectrics. This contribution is only a first step towards this goal. More work needs to be done for the deterministic - maybe even independent - design of mechanical and thermal properties. Future studies should also address the role of enthalpic interaction at the clay/polymer interface, interdigitation of the confined polymer, size effects of the platelets, and other nanosheet materials.

14.5 Experimental Section

14.5.1 Sample Preparation

The synthetic clay sodium fluorohectorite (Hec, $[Na_{0.5}]^{inter}[Mg_{2.5}Li_{0.5}]^{oct}[Si_4]^{tet}O_{10}F_2$) was delaminated by immersing it into Millipore water (0.5 wt%). The aqueous PVP solution (1 wt%) was added in the desired weight ratio. The suspension was mixed for at least one day in an overhead shaker. The homogeneity of the suspension was crosschecked by SAXS measurements. Self-supporting films were prepared using a fully automatic spray coating system. Every spraying cycle is followed by a drying cycle of 90 s at a temperature of 55 °C. We prepared pure PVP, pure Hec, and four hybrid Hec/PVP films. The self supporting films were characterized by thermogravimetric analysis, XRD, and TEM. Additional information about the sample preparation can be found in Section 14.7.1.

14.5.2 In-plane thermal conductivity measurements

Lock-in thermography measures the temperature spreading across the free-standing samples upon thermal excitation by a focused laser beam with a modulated intensity. To prevent convective heat losses, the experiments are conducted in a vacuum chamber. The amplitude and phase data are extracted from the radial temperature distribution. The thermal diffusivity is then fitted by the slope method for thermally thin films. With the density, determined by helium pycnometry, and the specific heat, determined by differential scanning calorimetry (DSC), the thermal conductivity can be calculated. More details are provided in Section 14.7.4.

14.5.3 Cross-plane thermal conductivity measurements

The photoacoustic method uses a modulated laser beam to periodically heat a transducer layer in intimate contact with the sample. The heat is converted into an acoustic wave propagating into a gas tight cell above the sample, which is filled with helium at 20 psi. A sensitive microphone detects the phase shift between the acoustic signal and the modulated laser by a lock-in amplifier. The frequency-dependent phase shift is then compared to a multilayer model, assuming one-dimensional heat transfer. Therefrom, the total layer resistance is obtained. With the film thickness determined by AFM, the effective thermal conductivity is calculated. More details are provided in Section 14.7.4.

14.5.4 Brillouin light spectroscopy (BLS)

BLS measures the phonon dispersion, $\omega(q)$, by detecting the Doppler frequency shift, ω, of the inelastically scattered light by sound waves ("phonons") with a wave vector, q. We recorded BLS spectra utilizing three scattering geometries (transmission, reflection, and backscattering) and two polarization configurations of the incident ($\lambda = 532\,\text{nm}$) and scattered light (polarized VV, depolarized VH), which allowed us to establish the nature of the observed phonons. We varied the incident angle to obtain the direction dependent sound velocities necessary for the determination of the anisotropic elasticity. The elastic stiffness tensors were obtained from the representation of the direction-dependent sound velocities by the Christoffel equation assuming transverse isotropy. The characteristic Young's moduli, shear moduli, and Poisson's ratios of the Bragg stacks were subsequently calculated. More details can be found in Section 14.7.9.

Acknowledgements

We thank Sabine Rosenfeldt for conducting SAXS measurements on the gel samples. We also thank Marco Schwarzmann for preparing TEM samples and taking TEM micrographs. The Volkswagen Foundation funded this project through a Lichtenberg professorship. DFG RE 3550/2-1 provided additional support. Z. W., K. R., and G. F. acknowledge the financial support by ERC AdG SmartPhon (Grant No. 694977).

14.6 References

[1] B. Graczykowski, A. E. Sachat, J. S. Reparaz, *et al.*, „Thermal conductivity and air-mediated losses in periodic porous silicon membranes at high temperatures," *Nature Communications*, vol. 8, no. 1, 2017.

[2] M. E. Siemens, Q. Li, R. Yang, *et al.*, „Quasi-ballistic thermal transport from nanoscale interfaces observed using ultrafast coherent soft x-ray beams," *Nature Materials*, vol. 9, no. 1, pp. 26–30, 2009.

[3] J. A. Tomko, A. Pena-Francesch, H. Jung, *et al.*, „Tunable thermal transport and reversible thermal conductivity switching in topologically networked bio-inspired materials," *Nature Nanotechnology*, vol. 13, no. 10, pp. 959–964, 2018.

[4] M. D. Losego, M. E. Grady, N. R. Sottos, D. G. Cahill, and P. V. Braun, „Effects of chemical bonding on heat transport across interfaces,“ *Nature Materials*, vol. 11, no. 6, pp. 502–506, 2012.

[5] G.-H. Kim, D. Lee, A. Shanker, *et al.*, „High thermal conductivity in amorphous polymer blends by engineered interchain interactions,“ *Nature Materials*, vol. 14, no. 3, pp. 295–300, 2014.

[6] S. Shen, A. Henry, J. Tong, R. Zheng, and G. Chen, „Polyethylene nanofibres with very high thermal conductivities,“ *Nature Nanotechnology*, vol. 5, no. 4, pp. 251–255, 2010.

[7] X. Huang, G. Liu, and X. Wang, „New secrets of spider silk: Exceptionally high thermal conductivity and its abnormal change under stretching,“ *Advanced Materials*, vol. 24, no. 11, pp. 1482–1486, 2012.

[8] R. Fuente, A. Mendioroz, and A. Salazar, „Revising the exceptionally high thermal diffusivity of spider silk,“ *Materials Letters*, vol. 114, pp. 1–3, 2014.

[9] C. Chiritescu, D. G. Cahill, N. Nguyen, *et al.*, „Ultralow thermal conductivity in disordered, layered WSe_2 crystals,“ *Science*, vol. 315, no. 5810, pp. 351–353, 2007.

[10] M. D. Losego, I. P. Blitz, R. A. Vaia, D. G. Cahill, and P. V. Braun, „Ultralow thermal conductivity in organoclay nanolaminates synthesized via simple self-assembly,“ *Nano Letters*, vol. 13, no. 5, pp. 2215–2219, 2013.

[11] B. Wicklein, A. Kocjan, G. Salazar-Alvarez, *et al.*, „Thermally insulating and fire-retardant lightweight anisotropic foams based on nanocellulose and graphene oxide,“ *Nature Nanotechnology*, vol. 10, no. 3, pp. 277–283, 2014.

[12] T. Li, J. Song, X. Zhao, *et al.*, „Anisotropic, lightweight, strong, and super thermally insulating nanowood with naturally aligned nanocellulose,“ *Science Advances*, vol. 4, no. 3, eaar3724, 2018.

[13] M. Chau, B. A. F. Kopera, V. R. Machado, *et al.*, „Reversible transition between isotropic and anisotropic thermal transport in elastic polyurethane foams,“ *Materials Horizons*, vol. 4, no. 2, pp. 236–241, 2017.

[14] J. Xiao, Q. Kuang, S. Yang, *et al.*, „Surface structure dependent electrocatalytic activity of co_3o_4 anchored on graphene sheets toward oxygen reduction reaction,“ *Scientific Reports*, vol. 3, no. 1, 2013.

[15] N. Song, D. Jiao, S. Cui, *et al.*, „Highly anisotropic thermal conductivity of layer-by-layer assembled nanofibrillated cellulose/graphene nanosheets hybrid films for thermal management,“ *ACS Applied Materials & Interfaces*, vol. 9, no. 3, pp. 2924–2932, 2017.

[16] A. Eckert, T. Rudolph, J. Guo, T. Mang, and A. Walther, „Exceptionally ductile and tough biomimetic artificial nacre with gas barrier function,“ *Advanced Materials*, vol. 30, no. 32, p. 1 802 477, 2018.

[17] T. Verho, M. Karesoja, P. Das, *et al.*, „Hydration and dynamic state of nanoconfined polymer layers govern toughness in nacre-mimetic nanocomposites,“ *Advanced Materials*, vol. 25, no. 36, pp. 5055–5059, 2013.

[18] T. Szabó, M. Szekeres, I. Dékány, *et al.*, „Layer-by-layer construction of ultrathin hybrid films with proteins and clay minerals,“ *The Journal of Physical Chemistry C*, vol. 111, no. 34, pp. 12 730–12 740, 2007.

[19] Z. Tang, N. A. Kotov, S. Magonov, and B. Ozturk, „Nanostructured artificial nacre,“ *Nature Materials*, vol. 2, no. 6, pp. 413–418, 2003.

[20] T. Still, *High Frequency Acoustics in Colloid-Based Meso- and Nanostructures by Spontaneous Brillouin Light Scattering*. Springer Berlin Heidelberg, 2010.

[21] E. Alonso-Redondo, M. Schmitt, Z. Urbach, *et al.*, „A new class of tunable hypersonic phononic crystals based on polymer-tethered colloids,“ *Nature Communications*, vol. 6, no. 1, 2015.

[22] V. Singh, T. L. Bougher, A. Weathers, *et al.*, „High thermal conductivity of chain-oriented amorphous polythiophene,“ *Nature Nanotechnology*, vol. 9, no. 5, pp. 384–390, 2014.

[23] A. Mendioroz, R. Fuente-Dacal, E. Apiñaniz, and A. Salazar, „Thermal diffusivity measurements of thin plates and filaments using lock-in thermography,“ *Review of Scientific Instruments*, vol. 80, no. 7, p. 074 904, 2009.

[24] L. Wang and T. Sasaki, „Titanium oxide nanosheets: Graphene analogues with versatile functionalities,“ *Chemical Reviews*, vol. 114, no. 19, pp. 9455–9486, 2014.

[25] P. Davidson, C. Penisson, D. Constantin, and J.-C. P. Gabriel, „Isotropic, nematic, and lamellar phases in colloidal suspensions of nanosheets,“ *Proceedings of the National Academy of Sciences*, vol. 115, no. 26, pp. 6662–6667, 2018.

[26] A. Lerf, „Storylines in intercalation chemistry,“ *Dalton Trans.*, vol. 43, no. 27, pp. 10 276–10 291, 2014.

[27] V. Nicolosi, M. Chhowalla, M. G. Kanatzidis, M. S. Strano, and J. N. Coleman, „Liquid exfoliation of layered materials,“ *Science*, vol. 340, no. 6139, pp. 1 226 419–1 226 419, 2013.

[28] M. Daab, N. J. Eichstaedt, A. Edenharter, S. Rosenfeldt, and J. Breu, „Layer charge robust delamination of organo-clays,“ *RSC Advances*, vol. 8, no. 50, pp. 28 797–28 803, 2018.

[29] J. E. F. C. Gardolinski and G. Lagaly, „Grafted organic derivatives of kaolinite: II. intercalation of primary n-alkylamines and delamination,“ *Clay Minerals*, vol. 40, no. 4, pp. 547–556, 2005.

[30] M. Stöter, D. A. Kunz, M. Schmidt, *et al.*, „Nanoplatelets of sodium hectorite showing aspect ratios of 20000 and superior purity,“ *Langmuir*, vol. 29, no. 4, pp. 1280–1285, 2013.

[31] S. Rosenfeldt, M. Stöter, M. Schlenk, *et al.*, „In-depth insights into the key steps of delamination of charged 2d nanomaterials,“ *Langmuir*, vol. 32, no. 41, pp. 10 582–10 588, 2016.

[32] K. Sano, Y. S. Kim, Y. Ishida, *et al.*, „Photonic water dynamically responsive to external stimuli,“ *Nature Communications*, vol. 7, no. 1, 2016.

[33] H. Kalo, W. Milius, and J. Breu, „Single crystal structure refinement of one- and two-layer hydrates of sodium fluorohectorite,“ *RSC Advances*, vol. 2, no. 22, p. 8452, 2012.

[34] D. A. Kunz, J. Erath, D. Kluge, *et al.*, „In-plane modulus of singular 2:1 clay lamellae applying a simple wrinkling technique,“ *ACS Applied Materials & Interfaces*, vol. 5, no. 12, pp. 5851–5855, 2013.

[35] M. Wong, R. Ishige, K. L. White, *et al.*, „Large-scale self-assembled zirconium phosphate smectic layers via a simple spray-coating process,“ *Nature Communications*, vol. 5, no. 1, 2014.

[36] W. Seidl and J. Breu, „Single crystal structure refinement of tetramethyl ammonium-hectorite,“ *Zeitschrift für Kristallographie - Crystalline Materials*, vol. 220, no. 2/3, 2005.

[37] A. Baumgartner, K. Sattler, J. Thun, and J. Breu, „A route to microporous materials through oxidative pillaring of micas,“ *Angewandte Chemie International Edition*, vol. 47, no. 9, pp. 1640–1644, 2008.

[38] N. L. McFarlane, N. J. Wagner, E. W. Kaler, and M. L. Lynch, „Poly(ethylene oxide) (PEO) and poly(vinyl pyrolidone) (PVP) induce different changes in the colloid stability of nanoparticles,“ *Langmuir*, vol. 26, no. 17, pp. 13 823–13 830, 2010.

[39] E. S. Tsurko, P. Feicht, F. Nehm, *et al.*, „Large scale self-assembly of smectic nanocomposite films by doctor blading versus spray coating: Impact of crystal quality on barrier properties,“ *Macromolecules*, vol. 50, no. 11, pp. 4344–4350, 2017.

[40] D. M. Moore and R. C. Reynolds, *X-ray Diffraction and the Identification and Analysis of Clay Minerals*, Second. New York: Oxford University Press, 1997.

[41] A. Philipp, N. W. Pech-May, B. A. F. Kopera, *et al.*, „Direct measurement of the in-plane thermal diffusivity of semitransparent thin films by lock-in thermography: An extension of the slopes method,“ *Analytical Chemistry*, vol. 91, no. 13, pp. 8476–8483, 2019.

[42] C. Clauser and E. Huenges, *Thermal Conductivity of Rocks and Minerals*. American Geophysical Union, 2013, pp. 105–126.

[43] J. K. Carson, S. J. Lovatt, D. J. Tanner, and A. C. Cleland, „Thermal conductivity bounds for isotropic, porous materials,“ *International Journal of Heat and Mass Transfer*, vol. 48, no. 11, pp. 2150–2158, 2005.

[44] Y.-F. Huang, Z.-G. Wang, H.-M. Yin, *et al.*, „Highly anisotropic, thermally conductive, and mechanically strong polymer composites with nacre-like structure for thermal management applications,“ *ACS Applied Nano Materials*, vol. 1, no. 7, pp. 3312–3320, 2018.

[45] L. P. Tremblay, M. B. Johnson, U. Werner-Zwanziger, and M. A. White, „Relationship between thermal conductivity and structure of nacre from haliotis fulgens,“ *Journal of Materials Research*, vol. 26, no. 10, pp. 1216–1224, 2011.

[46] R. W. Gammon, „Brillouin scattering experiments in the ferroelectric crystal triglycine sulfate,“ Ph.D. dissertation, John Hopkins University, 1967.

[47] L. E. McNeil and M. Grimsditch, „Elastic moduli of muscovite mica,“ *Journal of Physics: Condensed Matter*, vol. 5, no. 11, pp. 1681–1690, 1993.

[48] J. Krüger, L. Peetz, and M. Pietralla, „Brillouin scattering of semicrystalline poly(4-methyl-1-pentene): Study of surface effects of bulk and film material,“ *Polymer*, vol. 19, no. 12, pp. 1397–1404, 1978.

[49] S. P. Cheadle, R. J. Brown, and D. C. Lawton, „Orthorhombic anisotropy: A physical seismic modeling study,“ *Geophysics*, vol. 56, no. 10, pp. 1603–1613, 1991.

[50] M. Mah and D. R. Schmitt, „Experimental determination of the elastic coefficients of an orthorhombic material,“ *Geophysics*, vol. 66, no. 4, pp. 1217–1225, 2001.

[51] S. Cusack and A. Miller, „Determination of the elastic constants of collagen by brillouin light scattering,“ *Journal of Molecular Biology*, vol. 135, no. 1, pp. 39–51, 1979.

[52] Y. C. Chu and S. I. Rokhlin, „Comparative analysis of through-transmission ultrasonic bulk wave methods for phase velocity measurements in anisotropic materials,“ *The Journal of the Acoustical Society of America*, vol. 95, no. 6, pp. 3204–3212, 1994.

[53] M. Zgonik, P. Bernasconi, M. Duelli, *et al.*, „Dielectric, elastic, piezoelectric, electro-optic, and elasto-optic tensors of batio crystals,“ *Physical Review B*, vol. 50, no. 9, p. 5941, 1994.

[54] J. Midya, Y. Cang, S. A. Egorov, *et al.*, „Disentangling the role of chain conformation on the mechanics of polymer tethered particle materials,“ *Nano Letters*, vol. 19, no. 4, pp. 2715–2722, 2019.

[55] G. A. Elbaz, W.-L. Ong, E. A. Doud, *et al.*, „Phonon speed, not scattering, differentiates thermal transport in lead halide perovskites,“ *Nano Letters*, vol. 17, no. 9, pp. 5734–5739, 2017.

[56] A. S. Henry and G. Chen, „Spectral phonon transport properties of silicon based on molecular dynamics simulations and lattice dynamics,“ *Journal of Computational and Theoretical Nanoscience*, vol. 5, no. 2, pp. 141–152, 2008.

[57] W.-L. Ong, S. M. Rupich, D. V. Talapin, A. J. H. McGaughey, and J. A. Malen, „Surface chemistry mediates thermal transport in three-dimensional nanocrystal arrays,“ *Nature Materials*, vol. 12, no. 5, pp. 410–415, 2013.

14.7 Supporting Information

14.7.1 Sample Preparation

14.7.2 Materials

The synthetic clay sodium fluorohectorite (Hec, $[Na_{0.5}]^{inter}[Mg_{2.5}Li_{0.5}]^{oct}[Si_4]^{tet}O_{10}F_2$) was synthesized via melt synthesis followed by long-term annealing, according to an established procedure. The material featured a cation exchange capacity of 1.27 mmol/g (References[1],[2]). Polyvinylpyrrolidone (PVP; $M_w = 40.000$ g/mol) was provided by Sigma Aldrich.

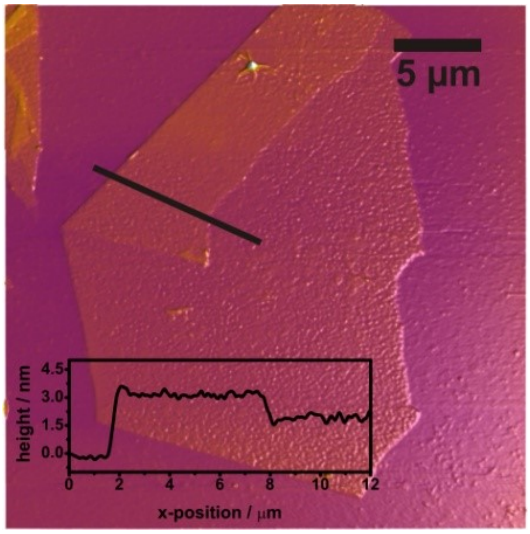


Fig. 14.5.: Atomic force microscopy image of a single delaminated Hec nanosheet. Reprinted with permission from Langmuir.[1],[2] Copyright 2019 American Chemical Society.

14.7.3 Film preparation

For the delamination, the synthetic Hec was immersed into Millipore water (0.5 wt%). The complete delamination was studied by small angle X-ray scattering (SAXS). The aqueous PVP solution (1 wt%) was added in the desired weight ratio. The suspension was mixed for at least 1 day in the overhead shaker. Homogeneity of the suspensions was cross-checked by SAXS measurements. All SAXS data were measured using the small-angle X-ray system "Double Ganesha AIR" (SAXSLAB, Denmark). The X-ray source of this laboratory-based system is a rotating anode (copper, MicroMax 007HF, Rigaku Corporation, Japan) providing a micro-focused beam. The data were recorded by a position sensitive detector (PILATUS 300 K, Dectris). To cover the range of scattering vectors between $0.004 - 1.0\,Å^{-1}$, different

detector positions were used. The measurements of the suspensions were done in 1 mm glass capillaries (Hilgenberg, code 4007610, Germany) at room temperature. To improve the detection limit of the in-house machine, the suspensions were first concentrated to $\approx$ 10 wt% by centrifugation at 10000 rpm for 1 hour. A rational basal series was found for all suspensions, indicating that all clay nanosheets were separated to exactly the same distance by water and PVP (Figure 14.6). Within experimental errors reaggregation to stacks of Hec can be excluded by the absence of a reflection typical for crystalline hydrated Hec phases at $q = 0.65 - 0.41\,\text{Å}^{-1}$.

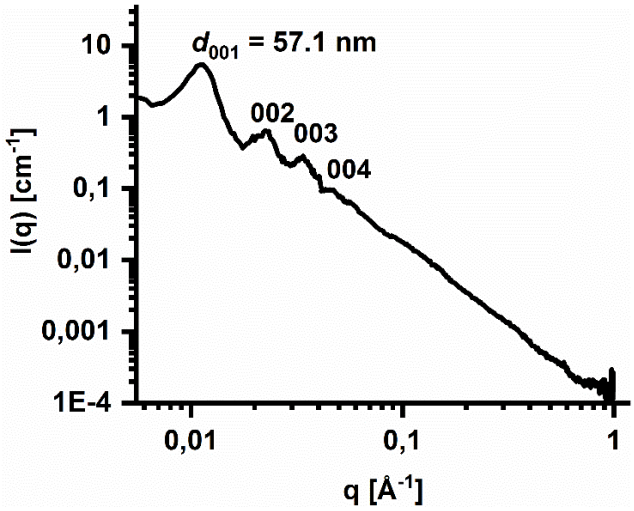


Fig. 14.6.: 1D SAXS pattern of the concentrated gel sample Hec40/PVP60.

The self-supporting films were prepared by spray coating. The fully automatic spray coating system was equipped with a SATA 4000 LAB HVLP 1.0 mm spray gun (SATA GmbH & Co. KG, Germany). Suspensions were sprayed on a corona-treated polyethylene terephthalate (PET) foil (optimont 501, bleher Folientechnik, Germany). The spraying and nozzle pressure were set constant at values of 2 and 4 bar, respectively. The round per flat fan control was set to 6 with a flow speed of 3 mL/s. The distance between the spraying gun and the substrate was 17 cm. The thickness of the suspension layer applied in one spraying step is $\approx$ 2 µm which corresponds to $\approx$ 20 nm dry nanocomposite film thickness. For drying the suspension layer, the sample is stopped under infrared lamps until evaporation of the solvent is complete. After every spraying cycle, a drying cycle of 90 s with a temperature of 55 °C took place. The spraying/drying cycle is repeated until the desired barrier film thickness of 50 µm is obtained. Afterwards, the film was dried at 100 °C for 3 days and peeled off from the PET foil to achieve self-supporting films. For characterization by photoacoustic analysis thinner films on the order of a few µm were spray coated onto clean 1 mm thick microscopy glass slides. In total, we prepared six samples, which are denoted as Hec0/PVP100, Hec23/PVP77, Hec31/PVP69, Hec40/PVP60, Hec51/PVP49, and Hec100/PVP0, respectively (Table 14.1). Throughout the

manuscript, all samples are referred to by the nominal volume fractions (vol%) of Hec and PVP, respectively. To rule out compositional changes during spray coating, these ratios were cross-checked (Table 14.1) by thermogravimetric analysis (TGA), using a Linseis STA PT 1600 (Linseis Messgeräte GmbH, Germany). Changes in mass observed upon heating in synthetic air up to 800 °C were attributed to the combustion of PVP.

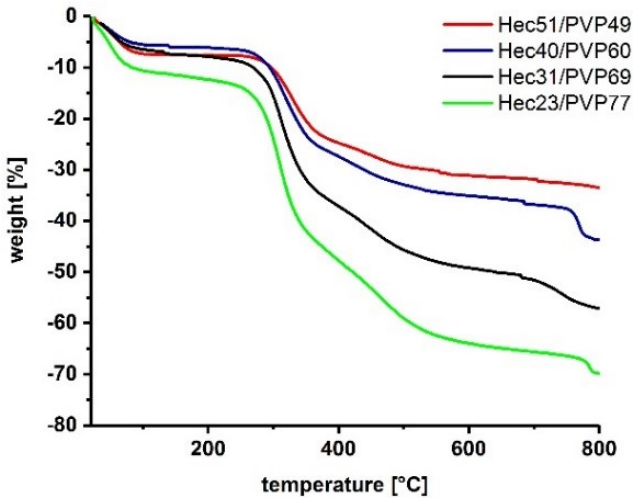


Fig. 14.7.: TGA curves of four hybrid Bragg stacks. The weight loss below 200 °C corresponds to the water.

X-ray diffraction (XRD) patterns for the films were recorded in Bragg-Brentano-geometry on an Empyrean diffractometer (PANalytical B.V.; the Netherlands) using Cu K_α radiation ($\lambda = 1.54187$ Å). The self-supporting films were placed on glass slides (Menzel-Gläser; Thermo Scientific). Prior to the XRD measurements, the samples were dried at 100 °C in a vacuum chamber for one week.

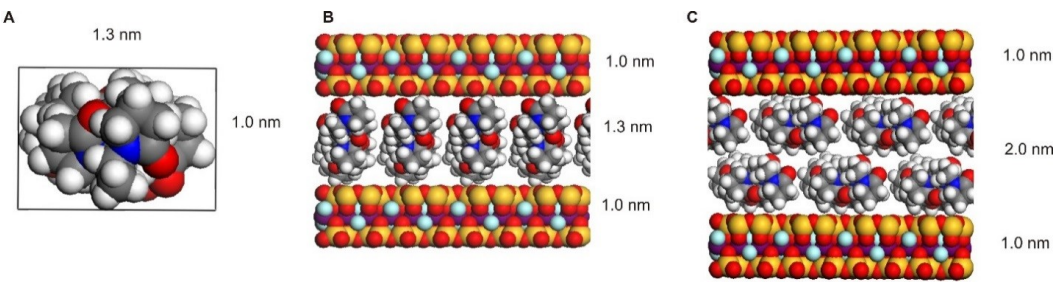


Fig. 14.8.: Space filling models. (A) PVP viewed along the polymer backbone chain. (B) Monolayer of PVP oriented with the longer principal axis perpendicular to the Hec nanosheet. (C) Bilayer of PVP lying in the plane of the Hec nanosheets.

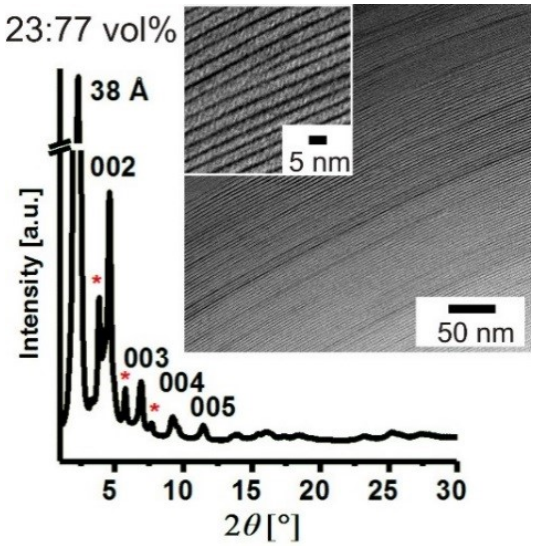


Fig. 14.9.: XRD pattern and TEM image of the nanocomposite film Hec23/PVP77. The red asterisks denote a second series of basal reflections of a minority phase ($d = 2.3\,\mathrm{nm}$).

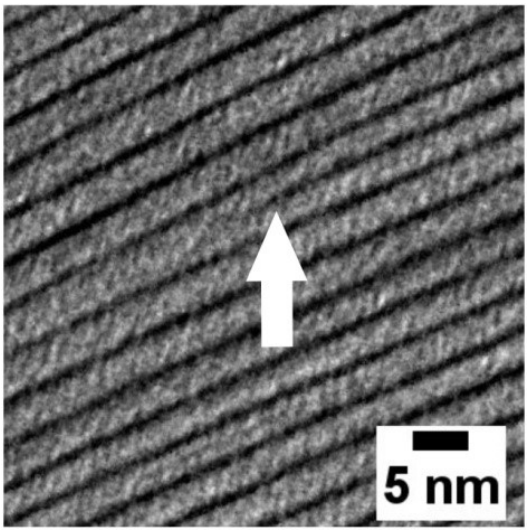


Fig. 14.10.: Lattice plane termination of a single Hec nanosheet.

As a measure of the quality of the one-dimensional crystallinity of the films, the coefficient of variation (CV) and the full width at half maximum (FWHM) were determined (Table 14.1). Large CV-values ($\approx 3\,\%$[3]) and large FWHM indicate non-rationality of the diffraction pattern as caused by a random interstratification of different interlayer heights. Assuming PVP and Hec densities of $1.2\,\mathrm{g/cm}$ and $2.7\,\mathrm{g/cm}$, respectively, nominal d-spacings can be calculated for the various volume ratios (d_{nominal} in Table 14.1).[4] They agree reasonably with those obtained from XRD measurements (d_{XRD} in Table 14.1). In analyzing the cross-plane thermal conductivity using the series resistance model, we used the d_{XRD} values.

Tab. 14.1.: Overview of the structural and chemical characterization.

	nominal Hec:PVP ratio [wt%]	nominal Hec:PVP ratio [vol%]	PVP content [a] [wt%]	PVP content [b][vol%]	d_{nominal} [Å]
Hec100/PVP0	100:0	100:0	-	-	10
Hec51/PVP49	70:30	51:49	27	45	18
Hec40/PVP60	60:40	40:60	38	58	23
Hec31/PVP69	50:50	31:69	49	68	30
Hec23/PVP77	40:60	23:77	59	76	41
Hec0/PVP100	0:100	0:100	100	100	-

	d_{XRD} [Å]	gallery height (PVP) [c][nm]	CV [%]	FWHM [°2θ]
Hec100/PVP0	10	-	-	-
Hec51/PVP49	19	0.9	5.9	0.4
Hec40/PVP60	23	1.3	1.3	0.3
Hec31/PVP69	30	2.0	1.0	0.3
Hec23/PVP77	38	2.8	2.8	0.3
Hec0/PVP100	-	-	-	-

[a] determined by TGA
[b] calculated with the PVP content determined by TGA
[c] gallery height corresponds to the d-spacing minus the Hec platelet height of 1 nm

Scanning electron microscopy (SEM) images were taken with a Zeiss LEO 1530 (Carl Zeiss AG, Germany) at an operating voltage of 3 kV. Cross sections were prepared by cutting with a razor blade. The samples with Hec were sputtered with 10 nm carbon.

Transmission electron microscopy (TEM) images were taken on a JEOL JEM-2200FS (JEOL GmbH, Germany) at an acceleration voltage of 200 kV. Cross-section pictures of the self-supporting films were prepared with a Cryo Ion Slicer IB-09060CIS (JEOL, Germany).

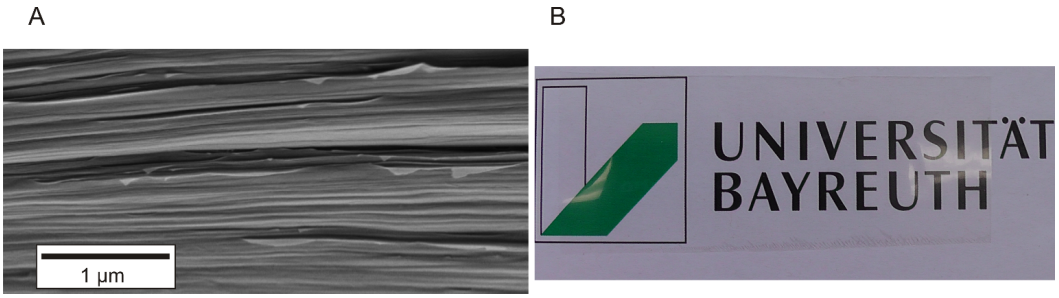


Fig. 14.11.: SEM image and photograph of the hybrid film. A) The SEM image displays the ordered arrangement of hectorite sheets at the macro-scale. B) As the hectorite platelets arrange highly ordered, light scattering is prevented and the films is transparent.

14.7.4 Thermal measurements

For the determination of the in-plane and cross-plane thermal conductivity, the density and the specific heat are needed. Therefore, Helium pycnometry and differential scanning calorimetry (DSC) were used. We determined the in-plane thermal diffusivity by lock-in thermography, and the cross-plane thermal conductivity by the photoacoustic method. Prior to the measurements, the samples were dried at 100 °C in a vacuum chamber for one week.

14.7.5 Helium pycnometry

The density of the samples was measured by helium pycnometry. Therefore, an Ultrapyc 1200e (Quantachrome Instruments) was used. Prior to each measurement the volume of the empty measurement cell was measured. Afterwards, small pieces of the freestanding films were weighed into the sample cell with a nominal volume of 1.8 cm^3. One hundred runs were performed to determine the volume of the films at room temperature. By knowing the mass (measured on a fine balance) and the volume, the density of the samples was calculated.

14.7.6 Differential scanning calorimetry

The specific heat was determined by DSC measurements according to the ASTM E1269 standard. The samples were freeze ground for better processability and contact to the DSC pans. The measurements were performed on a TA instruments Discovery DSC 2500. An isothermal step (1 h, 100 °C) was conducted prior to the measurement to ensure dry conditions. Then two heating cycles were used and only the second cycle evaluated. The temperature profile ranged from -40 to 250 °C using a heating rate of 20 $^{K}/_{min}$ with a nitrogen flow of 50 $^{mL}/_{min}$.

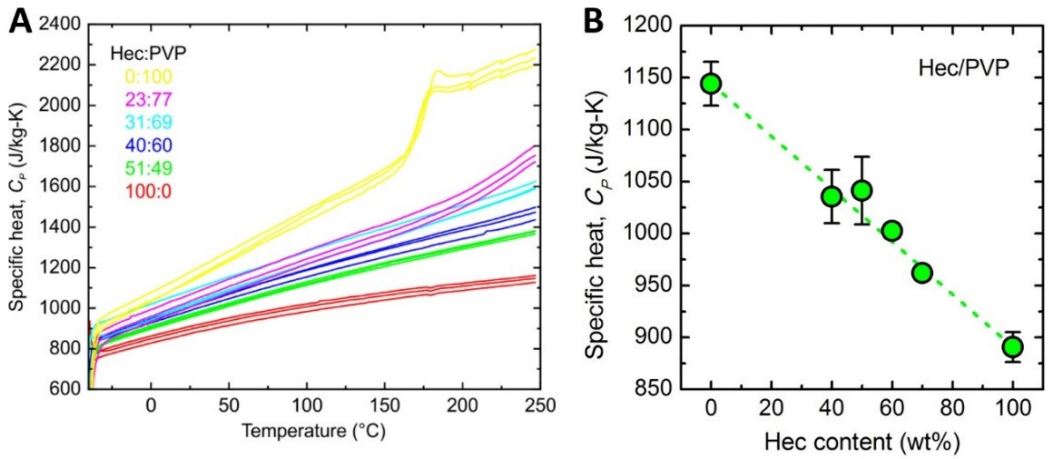


Fig. 14.12.: Temperature and composition dependencies of the specific heat. (A) Three samples per Hec/PVP composition were measured. The average specific heat at 25 °C was used to calculate the thermal conductivity. (B) The green dashed line shows the prediction by a simple mixing model, Cp(x) = (1-x)Cp(0%) + xCp(100%), with x being the hectorite weight fraction

14.7.7 Lock-in thermography

The in-plane thermal diffusivity of free-standing Hec/PVP stack films was obtained by lock-in thermography (LIT). The self-built LIT set-up is shown in Figure 14.13.

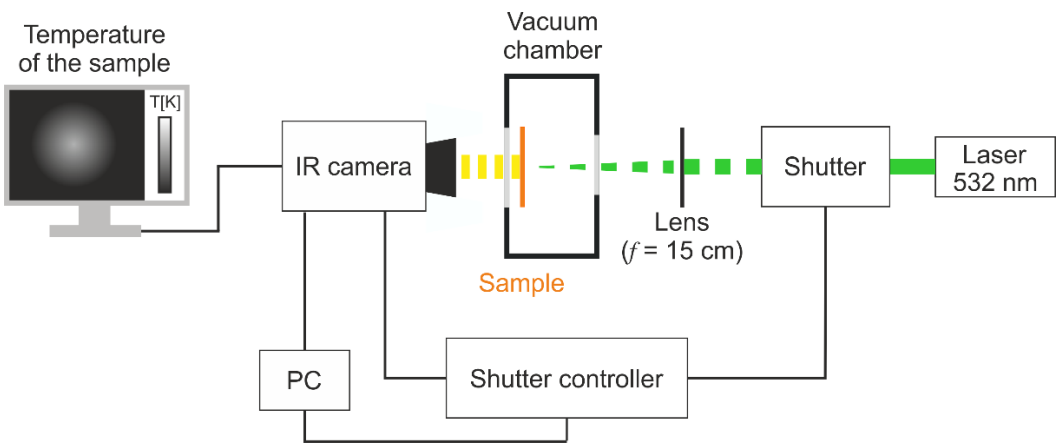


Fig. 14.13.: Scheme of the lock-in thermography set-up. The samples were measured in a vacuum chamber to avoid convective heat losses to the environment.

The sample is heated by a laser beam (Genesis MX 532-1000 SLM OPS, Coherent, $\lambda = 532\,\text{nm}$) focused onto the sample surface by a lens of 150 mm focal length. The intensity of the laser is modulated using a shutter (SH05/M, Thorlabs) controlled by a shutter controller (SC10, Thorlabs). The emitted infrared (IR) radiation of the sample surface is detected by an Infratec VarioCAM HD research IR camera (spectral window: 7.5-14 µm). The camera is equipped with a close-up lense. In

this configuration, the minimum spatial resolution is 29 µm (working distance: 33 mm). Since heat losses to the environment lead to an overestimation of the thermal diffusivity[5],[6] all samples were measured under vacuum conditions ($\approx 3 \times 10^{-3}$ mbar). Furthermore, all samples were coated with a 20 nm carbon layer for enhanced laser absorption. The coating of the sample was facing to the IR camera. LIT measurements were performed using Infratec's IRBISactiveonline software. Measurements were conducted at several lock-in frequencies between 0.219 and 1.765 Hz (depending on the Hec/PVP composition). Furthermore, each measurement was averaged over several (600-2000) lock-in periods to enhance the signal to noise ratio, with the first 60-100 periods being discarded. The software calculates automatically the amplitude and phase of the temperature oscillations (Figure 14.14). The red point marks the midpoint of the laser excitation and thus the midpoint of the radial distribution.

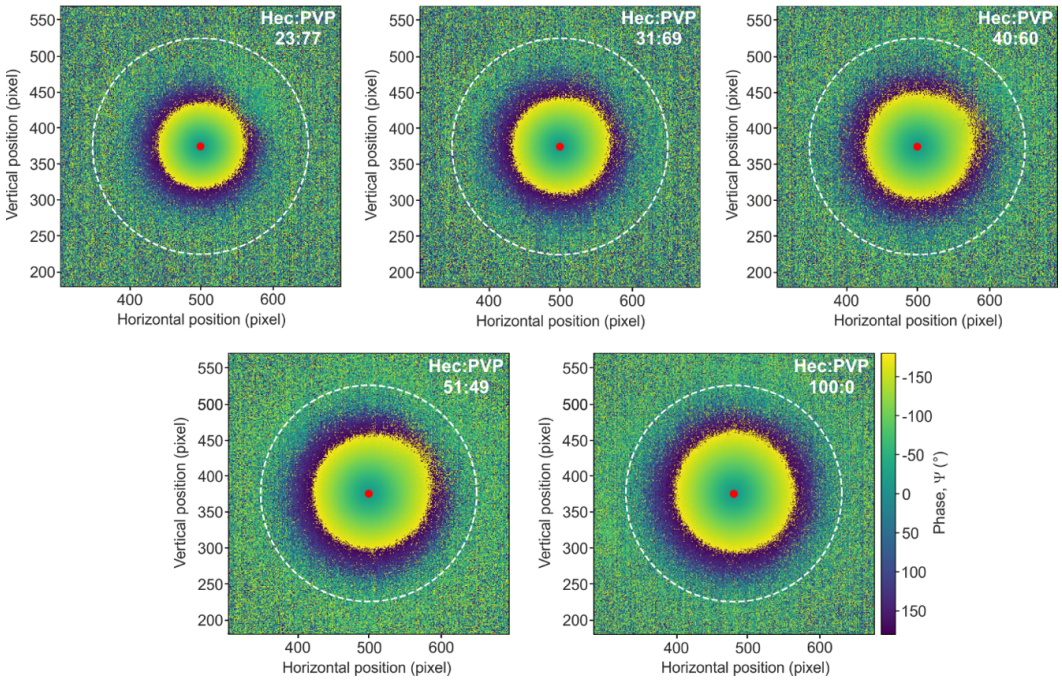


Fig. 14.14.: Exemplary two-dimensional phase images measured at a frequency of 1.111 Hz. The penetration depth of the temperature oscillations increases with increasing thermal diffusivity of the Hec/PVP films. The focal point of the laser is marked by a red point in the center of the IR image

A self-written Python script is used to extract radial profiles for the phase and amplitdue images (Figure 14.15). The thermal diffusivity is calculated from the phase and amplitude slopes according to the slope method of a thermally thin film:[6]

$$m_{\Psi} \cdot m_{\ln(T \cdot \sqrt{r})} = \frac{\pi f_{\text{lock-in}}}{\alpha_{||}} \tag{14.1}$$

Here, m_Ψ, is the slope of the linear relation of the phase and the radial distance, r, $m_{\ln(T\cdot\sqrt{r})}$ is the slope of the linear relation of the natural logarithm of the amplitude T multiplied by the square root of the radial distance r, flock-in is the lock-in frequency, and $\alpha_\parallel$ is the in-plane thermal diffusivity. Three films have been measured for each Hec/PVP composition. An average in-plane thermal diffusivity and a standard deviation were calculated for each composition, as summarized in Table 14.2.

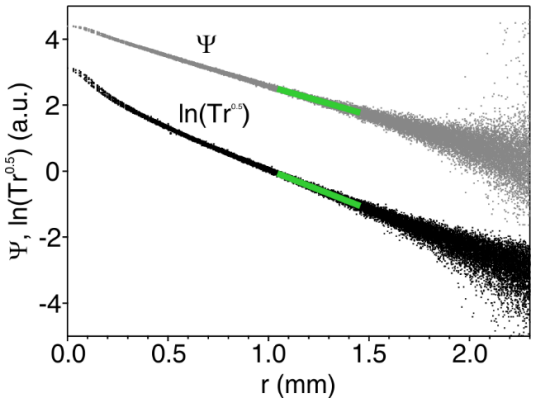


Fig. 14.15.: Exemplary phase Ψ and amplitude profiles. The green line indicates the region, where the linear fit was evaluated. This is sufficiently far away from the central excitation spot.

Tab. 14.2.: In-plane thermal diffusivity values of Hec/PVP stack films. Three films per Hec/PVP composition were measured, based on which an average in-plane thermal diffusivity and a standard deviation were calculated

Sample	In-plane thermal diffusivity (mm^2s^{-1})
Hec23/PVP77	1.02 ± 0.04
Hec31/PVP69	1.28 ± 0.02
Hec40/PVP60	1.52 ± 0.04
Hec51/PVP49	1.69 ± 0.03
Hec100/PVP0	2.35 ± 0.03

The in-plane thermal conductivity was calculated from the in-plane thermal diffusivity ($\alpha_\parallel$), density (ρ), and specific heat (C_p) as

$$k_\parallel = \alpha_\parallel \cdot \rho \cdot C_\mathrm{p} \tag{14.2}$$

14.7.8 Photoacoustic method

The cross-plane thermal conductivity was determined by the photoacoustic method. It uses the photoacoustic effect to determine the thermal properties of a sample.

Therefore, a modulated laser beam ($\lambda = 488\,\text{nm}$) periodically heats the sample. For good absorption of the laser energy a thin Au transducer layer ($\approx 150\,\text{nm}$) was coated on the sample surface. For photoacoustic characterization, the samples were spray-coated on a glass substrate. The layout of the measurement cell above the sample is shown in Figure 14.16A. The gas tight cell is filled with a helium pressure of 20 psi. The microphone (Bruel&Kjaer, 4398-A-011) connected to the cell measures an acoustic wave, which is induced by the periodic heat conduction from the transducer layer surface to the gas phase. As shown in Figure 14.16B, the microphone is linked to a lock-in amplifier with integrated signal generator (Zurich instruments, HF2LI). The signal generator controls the electro-optic modulator (EOM, Conoptics, M25A) and therefore the frequency of the modulated laser beam. The acoustic signal detected by the microphone is then transferred into a phase and amplitude shift in relation to the modulation of the incident laser beam.

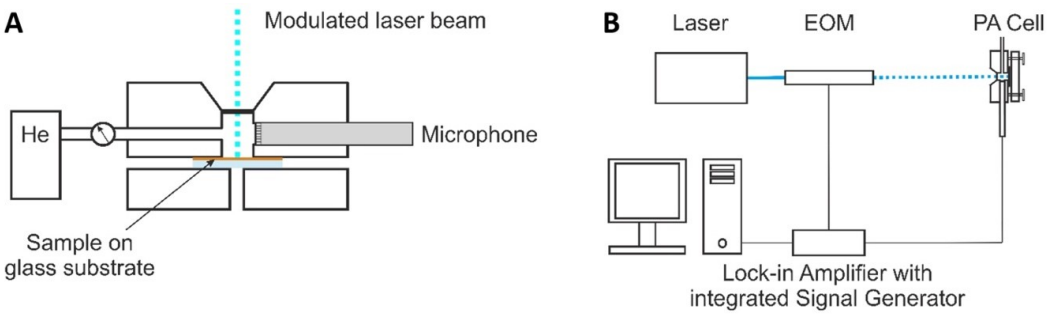


Fig. 14.16.: Scheme of photoacoustic measurements. (A) The photoacoustic cell. (B) The whole setup.

The phase shift is detected as a function of the frequency in a range from 110 Hz to 4000 Hz. The signal is then normalized with the phase shift signal of a thermally thick glass sample (1 mm) with known thermal properties. A representative frequency sweep is shown in Figure 14.17. The red line indicates the best fit according to the fitting procedure presented by Singh *et al.*[7] They used the generalized multilayer model of Hu *et al.*[8] assuming one-dimensional heat transfer. The unknown fitting parameters are the contact resistance between the gold layer and the sample, the thermal diffusivity of the sample, and the contact resistance between sample and substrate. For thin films the fit is not very sensitive to the individual parameters, but to the total layer resistance. Therefore, only the total layer resistance is reported. From the total layer resistance it is possible to calculate the effective thermal conductivity by dividing with the thickness. The thickness of the samples was determined by AFM measurements. The values of the total layer resistance and sample thickness are summarized in Table 14.3. For each Hec/PVP ratio samples with three different thicknesses were measured to exclude influences from the sample thickness.

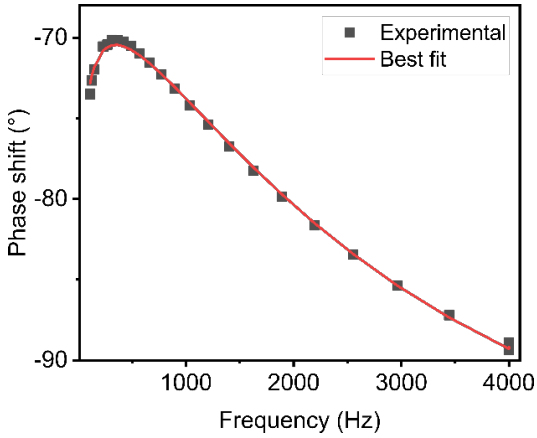


Fig. 14.17.: Normalized photoacoustic phase signal. Representative photoacoustic measurement of the Hec31/PVP69 sample with a thickness of 1.979 µm. The red line shows the best fit.

Tab. 14.3.: Summary of the photoacoustic measurements. The total layer resistance, the thickness, and the resulting effective cross-plane thermal conductivity are given for each sample.

Sample	Total layer resistance [mm^2K/W]	Thickness [µm]	Effective cross-plane thermal conductivity [W/mK]
Hec100/PVP0	1.58	0.28	0.177
Hec100/PVP0	2.55	0.595	0.234
Hec100/PVP0	4.34	0.94	0.217
Hec51/PVP49	3.69	0.255	0.069
Hec51/PVP49	7.74	0.722	0.093
Hec51/PVP49	11.53	0.957	0.083
Hec40/PVP60	6.54	0.463	0.071
Hec40/PVP60	14.03	1.139	0.081
Hec40/PVP60	18.96	1.987	0.105
Hec31/PVP69	8.20	0.744	0.091
Hec31/PVP69	17.58	1.382	0.079
Hec31/PVP69	24.06	1.979	0.082
Hec23/PVP77	8.81	0.568	0.064
Hec23/PVP77	17.45	1.677	0.096
Hec23/PVP77	26.37	2.568	0.097
Hec0/PVP100	2.26	0.351	0.155
Hec0/PVP100	4.50	0.792	0.176

14.7.9 Brillouin light spectroscopy

Brillouin Light Spectroscopy (BLS) measures the phonon dispersion, i.e., the wave vector, q, dependent sound velocity, $\nu_i(q)$, by detecting the Doppler frequency shift, f, in laser light inelastically scattered by sound waves ("phonons"). Since the phonon modes in question are thermally populated, the Bragg condition for light scattering is satisfied independent of the scattering angle, in contrast to the kindred time-domain ("picosecond ultrasonic") techniques that require external phonon injection via special sample preparation. Since the latter typically involves deposition of non-transparent metallic transducer films, they also do not readily allow for transmission and backscattering measurements, further limiting the potential of these techniques for probing anisotropic materials. Assuming a planar sample, BLS can be conducted in three scattering geometries: transmission, reflection, and backscattering. For angle-dependent measurements in the transmission and reflection geometries, the laser source ($\lambda = 532\,\mathrm{nm}$) was mounted on a goniometer and rotated around the sample, similar to a wide-angle X-ray (WAXS) setup. For the transmission geometry (top-right inset to Figure 14.3B), the propagation vector, q, of the sound wave lies in the plane of the sample and its modulus, $|q|$, is independent of the refractive index according to $|q|_{\|} = \left(\frac{4\pi}{\lambda}\sin(\beta)\right)$, with β being the light incident angle. For the reflection geometry, the Bragg condition is fulfilled for sound waves with q along the normal to the sample plane, so that $|q|_{\perp} = \left(\frac{4\pi}{\lambda}\sqrt{n^2 - \sin(\beta)^2}\right)$with n being the refractive index of the sample.[9] In order to find n, several points at different β were acquired, and then fit linearly; n was determined under the constraint that the fit has to pass through the origin (Figure 14.18A), and the obtained values are reported in Table 14.4. In contrast to the transmission geometry, the range of the dispersion relationship that can be scanned by varying $|q|_{\perp}$ is more restricted due to refraction of the laser beam at the air-sample boundary. Finally, backscattering measurements can be conducted, where the incident and scattered light traverse the same path, and the sample is mounted on a rotating stage with angular gradation marks. Only a single dispersion point, corresponding to q directed opposite to k_i in the film and of a magnitude, $|q| = \frac{4\pi n}{\lambda}$ is accessible in the backscattering geometry, but all possible $\frac{q}{|q|}$ directions can be probed.

Because backscattering measurements use the same lens for focusing the incident and collimating scattered light, the alignment of the setup is generally much easier, so that all the backscattering spectra were acquired with microscope objectives (typically 10x) to reduce the spectrum accumulation time. From the given formulas, it is easy to see that the Brillouin frequency shift does not exceed $2n\nu/\lambda$, where v is sound velocity. These frequencies render viscoelasticity effects negligible and are detected using a high-resolution six-pass Tandem Fabry Perot (TFP) interferometer (JRS Instruments, Switzerland) optimized for the 1-50 GHz range. Values still further

out on the dispersion relationship can be accessed using, for example, picosecond ultrasonic interferometry (PUI), but at the expense of great effort and without any gain in information for the problem at hand.[10] Finally, BLS also offers direct access to the shear moduli, simply by analyzing scattered light in different polarizations: VV (i.e., vertical incident and vertical scattered light) corresponds to quasi-longitudinal and quasi-transverse phonon modes, and VH (i.e., vertical incident and horizontal scattered light) to the pure-transverse phonon mode (for transversely isotropic samples, HV does not show a transverse peak). Before the BLS measurements, the samples were dried at 100 °C in a vacuum chamber for one week to remove any residual water content.

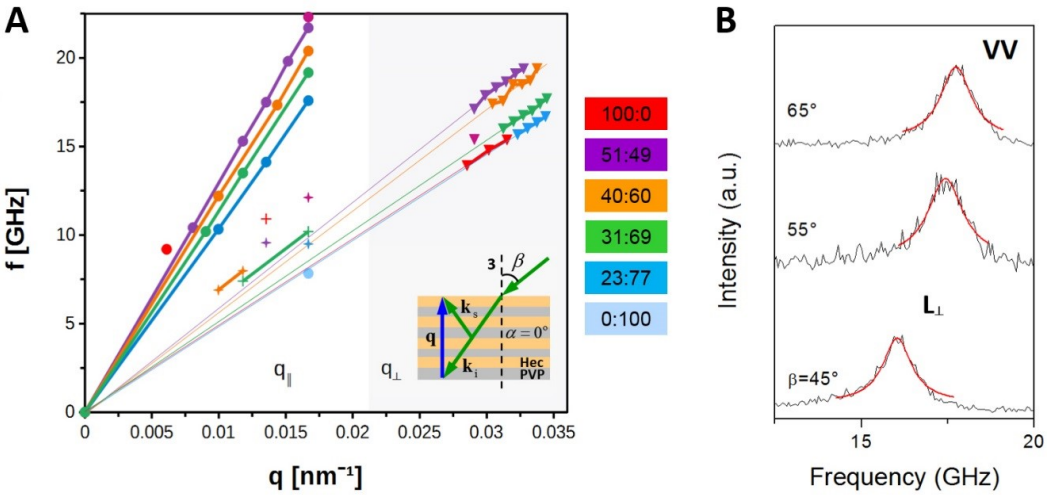


Fig. 14.18.: BLS experiments in the reflection geometry. (A) Dispersion (frequency vs. wave vector) of the longitudinal (L) and transverse (T) phonons in the hybrid stacks and the two constituent components (see the color code) obtained from the polarized (VV) and depolarized (VH) BLS spectra recorded in the reflection geometry (inset). (B) Typical VV spectra of the Hec31/PVP69 hybrid film at three laser incident angles.

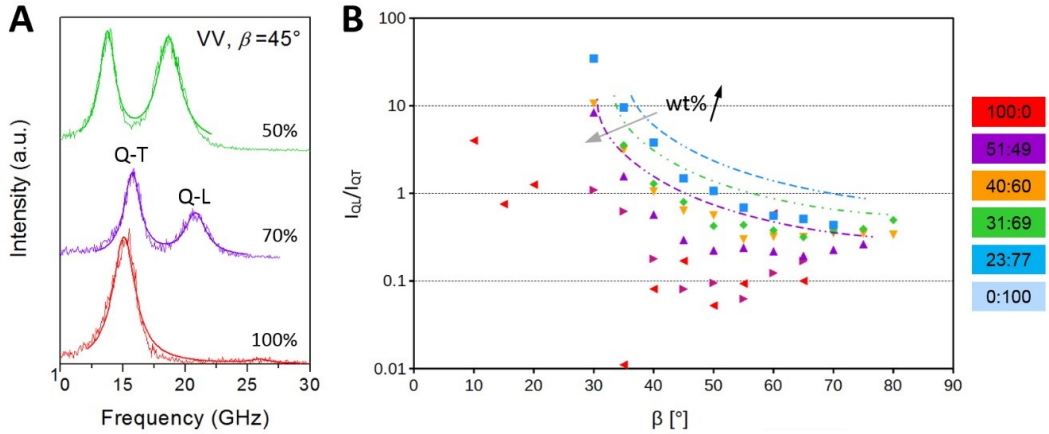


Fig. 14.19.: BLS spectra and peak intensity. (A) Polarized (VV) BLS spectra recorded in the backscattering geometry (inset to Figure 14.3A) for two hybrid stacks and the pure Hec film at an incidence angle of 45°. (B) The intensity ratio of the Q-L to Q-T peaks in (A) vs. the laser incident angle.

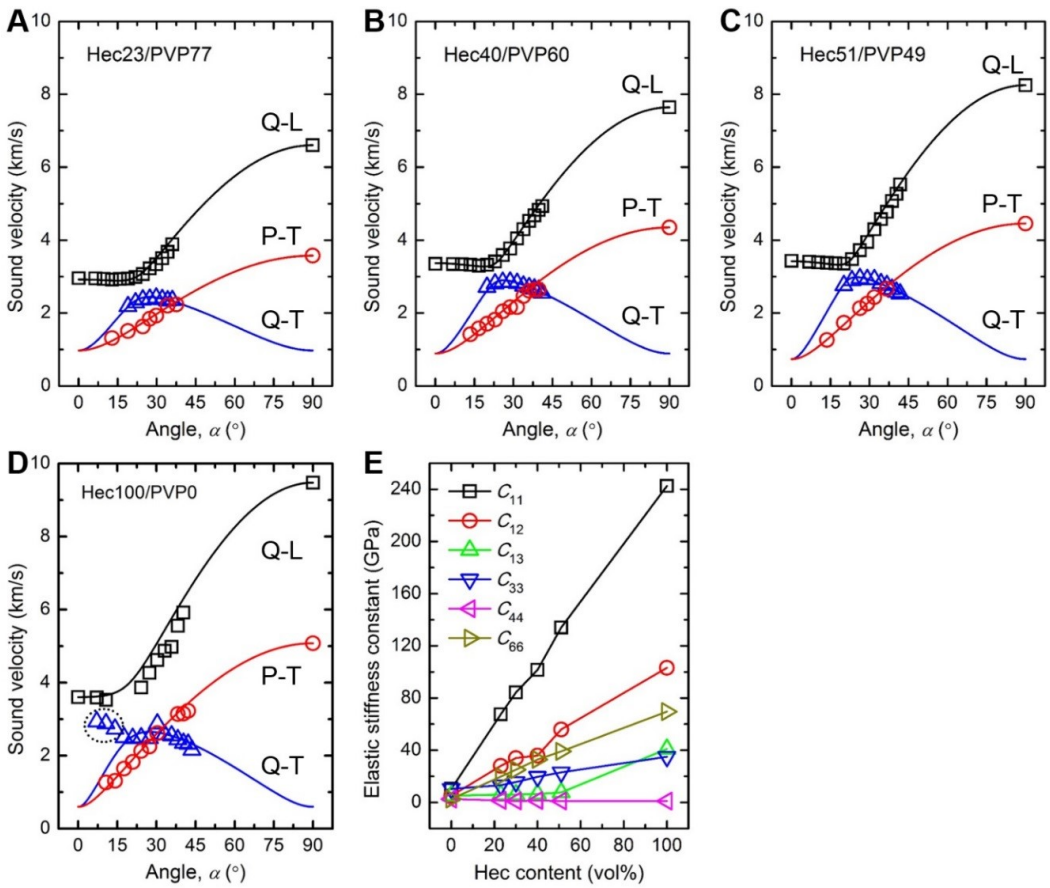


Fig. 14.20.: Direction-dependent sound velocities of hybrid stack films. (A) Hec23/PVP77, (B) Hec40/PVP60, (C) Hec51/PVP49, and (D) Hec100/PVP0 (the data points in the dotted circle were not used in the fitting). Q-L, P-T, and Q-T represent the quasi-longitudinal, pure-transverse, and quasi-transverse phonon modes, respectively. (E) Variation of the elastic stiffness constants with the Hec volume fraction.

Based on the BLS-measured direction-dependent sound velocities, we obtained the elastic tensor within the framework of the Christoffel equation.[11],[12]

$$\begin{bmatrix} \Gamma_{11} - \rho\nu^2 & \Gamma_{12} & \Gamma_{13} \\ \Gamma_{21} & \Gamma_{22} - \rho\nu^2 & \Gamma_{23} \\ \Gamma_{31} & \Gamma_{32} & \Gamma_{33} - \rho\nu^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{14.3}$$

where ρ is the density of the sample, ν is the sound velocity, $\mathbf{u} = [u_1, u_2, u_3]^{\mathrm{T}}$ is the displacement vector, and Γ_{ik} (i, k = 1, 2, 3) is the Christoffel stress, which is defined as

$$\Gamma_{ik} = \sum_{j=1}^{3} \sum_{i=1}^{3} C_{ijkl} n_j n_l. \tag{14.4}$$

Here, C_{ijkl} denotes an element of the 4th rank elastic tensor, and $\mathbf{n} = [n_1, n_2, n_3]^{\mathrm{T}}$ represents the phonon propagation direction.For a transversely isotropic material, the elastic tensor, in the Voigt notation, has the following form.[13]

$$\mathbf{C} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & & & \\ C_{12} & C_{11} & C_{13} & & & \\ C_{13} & C_{13} & C_{33} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ & & & & & C_{44} = \frac{C_{11}-C_{12}}{2} \end{bmatrix}, \tag{14.5}$$

and contains five independent stiffness constants. After some algebra, the sound velocities of the Q-L, Q-T, and P-T modes along a direction defined by α can be represented as follows,

$$\nu_{\text{Q-L}}(\alpha) = \sqrt{\frac{-A_1 + \sqrt{A_1^2 - 4A_2}}{2\rho}} \tag{14.6}$$

$$\nu_{\text{Q-T}}(\alpha) = \sqrt{\frac{-A_1 - \sqrt{A_1^2 - 4A_2}}{2\rho}} \tag{14.7}$$

$$\nu_{\text{P-T}}(\alpha) = \sqrt{\frac{A_3}{\rho}} \tag{14.8}$$

where,

$$A_1 = -\left(\sin(\alpha C_{11})^2 + \cos(\alpha C_{33})^2 + C_{44}\right) \tag{14.9}$$

$$A_2 = \sin(\alpha C_{11} C_{44})^4 + \sin(\alpha)^2 \cos(\alpha)^2 \left(C_{11}C_{33} - C_{13}^2 - 2C_{13}C_{44}\right) + \cos(\alpha C_{33} C_{44})^4 \tag{14.10}$$

$$A_3 = \sin(\alpha C_{66})^2 + \cos(\alpha C_{44})^2 \tag{14.11}$$

Through nonlinear χ^2 fitting of the BLS-measured sound velocities (i.e., ν vs. α) with Equations (14.6)-(14.8), we obtained the elastic stiffness constants as well as their uncertainties.[14] The χ^2 is defined as

$$\chi^2 = \sum_i \frac{[\nu_{i,\text{fit}}(C_{11}, C_{12}, C_{13}, C_{33}, C_{33}, \alpha) - \nu_{i,\text{exp}}(\alpha)]^2}{(\Delta\nu_{i,\text{exp}})^2} \tag{14.12}$$

where $\nu_{i,\text{fit}}$ and $\nu_{i,\text{exp}}$ are the fitted and experimental sound velocities, respectively, $\Delta\nu_{i,\text{exp}}$ is the uncertainty of the measured sound velocity, and the summation is over all experimental sound velocities. By considering the uncertainties of the measured angles, refractive indices, and phonon frequencies, we estimated $\nu_{i,\text{exp}}$ to be 0.02 $\nu_{i,\text{exp}}$. We imposed the following constraints for the elastic stiffness constants:[15]

1. $C_{11} > |C_{12}|$
2. $C_{44} > 0$
3. $C_{33}(C_{11} + C_{12}) > 2C_{13}^2$,

which ensure positive Young's and shear moduli. The availability of experimental data for all the Q-L, Q-T, and P-T modes allows unique determination of C_{11}, C_{12}, C_{13}, C_{33}, and C_{44}.

After that, we calculated the engineering mechanical properties,[13] which include the in-plane Young's modulus ($E_\parallel$), cross-plane Young's modulus ($E_\perp$), sliding shear modulus (G_{13}), torsional shear modulus (G_{12}), and two characteristic Poisson's ratios (ν_{13} and ν_{12}). Note that only five of the engineering mechanical properties are independent (typically, $E_\parallel$, $E_\perp$, G_{13}, G_{12}, and one of ν_{13} and ν_{12} are chosen). Also note that to be consistent with the direction-dependent thermal conductivity results, we used subscripts, "$\parallel$" and "$\perp$", to ,represent directions parallel and perpendicular to the sample film, respectively, rather than directions parallel and perpendicular to the "3"-axis (i.e., the symmetry axis), as seen in typical analysis of transversely isotropic materials. The relevant mechanical properties are summarized in Tables 14.4 and 14.5.

Tab. 14.4.: Summary of elastic stiffness constants. Composition (Hec vol%, Hec wt%), density (ρ), refractive index (n), and elastic stiffness constants (C_{11}, C_{12}, C_{13}, C_{33}, C_{44}, and C_{66}) of the Hec/PVP hybrid Bragg stack films.

Sample ID	Hec vol%	Hec vw%	ρ (g/cm³)	n
Hec0/PVP100	0	0	1.2	1.53*
Hec23/PVP77	23	40	1.55	1.54
Hec31/PVP69	31	50	1.62	1.50
Hec40/PVP60	40	60	1.74	1.47
Hec51/PVP49	51	70	1.97	1.45
Hec100/PVP0	100	100	2.70	1.40

Sample ID	C_{11} (GPa)	C_{12} (GPa)	C_{13} (GPa)	C_{44} (GPa)	C_{44} (GPa)	C_{66} (GPa)
Hec0/PVP100	10.4	5.2	5.2	10.4	2.6	2.6
Hec23/PVP77	67.6±1.7	28.0±1.9	6.0 ± 1.2	13.2±0.3	1.5 ± 0.1	19.8±0.5
Hec31/PVP69	84.3±1.5	34.0±1.8	6.1 ± 1.5	15.7±0.3	1.2 ± 0.2	25.1±0.5
Hec40/PVP60	101.6 ± 1.8	35.9±2.2	6.5 ± 1.8	19.7±0.4	1.4 ± 0.1	32.9±0.6
Hec51/PVP49	134.1 ± 2.2	55.9±2.9	7.7 ± 2.3	23.0±0.5	1.1 ± 0.1	39.1±0.9
Hec100/PVP0	242.5 ± 6.2	103.2 ± 6.7	40.9±5.2	35.0±0.9	1.0 ± 0.1	69.6±1.2

*Source: https://refractiveindex.info/?shelf=organic&book=polyvinylpyrrolidone&page=Konig.

Tab. 14.5.: Summary of engineering mechanical properties. Composition (Hec vol%, Hec wt%) and engineering mechanical properties of the Hec/PVP hybrid Bragg stack films. $E_{\parallel}$: in-plane Young's modulus; $E_{\perp}$: cross-plane Young's modulus; G_{13}, G_{23}: sliding shear moduli; G_{12}: torsional shear modulus; $\nu_{13}, \nu_{23}, \nu_{12}$: Poisson's ratios ($\nu_{13}$ represents the strain response in the j-direction due to a stress in the i-direction).

Sample ID	Hec vol%	Hec vw%	$E_{\parallel}$ (GPa)	$E_{\perp}$ (GPa)	$\frac{E_{\parallel}}{E_{\perp}}$
Hec0/PVP100	0	0	7.0	7.0	1
Hec23/PVP77	23	40	55.0 ± 2.5	12.5 ± 0.4	4.4
Hec31/PVP69	31	50	69.7 ± 2.3	15.1 ± 0.4	4.6
Hec40/PVP60	40	60	88.1 ± 2.6	19.1 ± 0.5	4.6
Hec51/PVP49	51	70	109.9 ± 3.5	22.4 ± 0.6	4.9
Hec100/PVP0	100	100	178.9 ± 9.9	25.4 ± 2.6	7.0

Sample ID	$G_{13} = G_{23}$ (GPa)	G_{12} (GPa)	$\nu_{13} = \nu_{23}$	ν_{12}
Hec0/PVP100	2.6	2.6	0.33 (assumed)	0.33 (assumed)
Hec23/PVP77	1.5 ± 0.1	19.8 ± 0.5	0.06 ± 0.01	0.39 ± 0.03
Hec31/PVP69	1.2 ± 0.2	25.1 ± 0.5	0.05 ± 0.01	0.39 ± 0.2
Hec40/PVP60	1.4 ± 0.1	32.9 ± 0.9	0.05 ± 0.01	0.34 ± 0.02
Hec51/PVP49	1.1 ± 0.1	39.1 ± 0.9	0.04 ± 0.01	0.41 ± 0.02
Hec100/PVP0	1.0 ± 0.1	69.6 ± 1.2	0.12 ± 0.02	0.29 ± 0.06

14.7.10 Evaluation of interfacial thermal conductance

The series resistance model (SRM) shown in Figure 14.4F illustrates a hybrid stack unit cell consisting of one Hec layer and one PVP layer. We describe the corrugation of the Hec nanosheet (Figure 14.1B) by a thermal interface conductance, $G_{\mathrm{Hec/PVP}}$. For the unit cell in Figure 14.4F, the total thermal resistance can be calculated as

$$\frac{d_{\mathrm{XRD}}}{k_{\perp}} = \frac{d_{\mathrm{Hec}}}{k_{\mathrm{Hec}}} + \frac{d_{\mathrm{PVP}}}{k_{\mathrm{PVP}}} + \frac{2}{G_{\mathrm{Hec/PVP}}}, \tag{14.13}$$

where d_{XRD} is the basal spacing as determined by XRD (Table 14.1), is the effective cross-plane thermal conductivity, $d_{\mathrm{Hec}} = 10\,\text{Å}$, and $d_{\mathrm{PVP}} = d_{\mathrm{XRD}} - d_{\mathrm{Hec}}$. For the thermal conductivity of the two components, we used the following values: $k_{\mathrm{Hec}} = 5.71\,\mathrm{W/mK}$, and $k_{\mathrm{PVP}} = 0.17\,\mathrm{W/mK}$. The Hec/PVP interfacial conductance, $G_{\mathrm{Hec/PVP}}$, is determined by least squares fitting of the experimental $k_{\perp}$ data (red solid line in Figure 14.4C) to be $89 \pm 8\,\mathrm{MW/m^2K}$, which falls well into the range of interfacial conductance reported for inorganic/organic interfaces.[16],[17] We also

obtained the Hec/Hec interfacial conductance, $G_{\mathrm{Hec/Hec}} = 219 \pm 28\,\mathrm{MW/m^2K}$, by considering a unit cell consisting of one Hec layer and one Hec/Hec interface i.e.

$$\frac{d_{\mathrm{XRD}}}{k_{\perp}} = \frac{d_{\mathrm{Hec}}}{k_{\mathrm{Hec}}} + \frac{1}{G_{\mathrm{Hec/Hec}}}. \tag{14.14}$$

14.7.11 Uncertainty analysis

The uncertainties (standard deviations) of the data reported in this work were analyzed by taking into account the instrument accuracy, repetitive measurements, and propagation of uncertainties. The sound velocities were estimated to have an error bar of 2 %. The uncertainties of the elastic stiffness constants were determined according to Zgonik *et al.*[14] The uncertainties of the engineering mechanical properties were calculated according to principles of uncertainty propagation. For instance,

$$E_{\perp} = C_{33} - \frac{2C_{13}^2}{C_{11} + C_{12}} \tag{14.15}$$

$$\Delta E_{\perp} = \sqrt{\sum_{ij=11,12,13,33} \left(\frac{\partial E_{\perp}}{\partial C_{ij}}\right)^2 (\Delta C_{ij})^2 + \sum_{ij,kl=11,12,13,33}^{ij<kl} 2\frac{\partial E_{\perp}}{\partial C_{ij}}\frac{\partial E_{\perp}}{\partial C_{kl}}\Delta C_{ij|kl}} \tag{14.16}$$

where

$$\frac{\partial E_{\perp}}{\partial C_{11}} = \frac{2C_{13}^2}{(C_{11} + C_{12})^2}, \tag{14.17}$$

$$\frac{\partial E_{\perp}}{\partial C_{12}} = \frac{2C_{13}^2}{(C_{11} + C_{12})^2}, \tag{14.18}$$

$$\frac{\partial E_{\perp}}{\partial C_{13}} = -\frac{4C_{13}^2}{C_{11} + C_{12}}, \tag{14.19}$$

$$\frac{\partial E_{\perp}}{\partial C_{33}} = 1, \tag{14.20}$$

Similar expressions can be derived for the other engineering moduli and the Poisson's ratios. The uncertainties of the thermal measurement results are analyzed in a similar way, i.e., by considering uncertainty propagation. For example,

$$k_{\parallel} = \alpha_{\parallel}\rho C_{\mathrm{P}} \tag{14.21}$$

$$\Delta k_{\parallel} = \sqrt{\left(\frac{\partial k_{\parallel}}{\partial \alpha_{\parallel}}\right)^2 \left(\Delta\alpha_{\parallel}\right)^2 + \left(\frac{\partial k_{\parallel}}{\partial \rho}\right)^2 (\Delta\rho)^2 + \left(\frac{\partial k_{\parallel}}{\partial C_{\mathrm{P}}}\right)^2 (\Delta C_{\mathrm{P}})^2} \tag{14.22}$$

where

$$\frac{\partial k_{\parallel}}{\partial \alpha_{\parallel}} = \rho C_{\mathrm{P}} \tag{14.23}$$

$$\frac{\partial k_{\parallel}}{\partial \rho} = \alpha_{\parallel} C_{\mathrm{P}} \tag{14.24}$$

$$\frac{\partial k_{\parallel}}{\partial C_{\mathrm{P}}} = \alpha_{\parallel} \rho \tag{14.25}$$

Here we assumed negligible covariance terms. Similar expressions can be derived for $k_{\perp}$, $k_{\parallel}/k_{\perp}$, $\bar{\Lambda}$, etc. The uncertainties of the data are reported as error bars in the figures shown in the main text and Supporting Information. For clarity, error bars smaller than the symbol size are not shown.

14.7.12 References

[1] J. Breu, W. Seidl, A. J. Stoll, K. G. Lange, and T. U. Probst, „Charge homogeneity in synthetic fluorohectorite,“ *Chemistry of Materials*, vol. 13, no. 11, pp. 4213–4220, 2001.

[2] M. Stöter, D. A. Kunz, M. Schmidt, *et al.*, „Nanoplatelets of sodium hectorite showing aspect ratios of 20000 and superior purity,“ *Langmuir*, vol. 29, no. 4, pp. 1280–1285, 2013.

[3] D. M. Moore and R. C. Reynolds, *X-ray Diffraction and the Identification and Analysis of Clay Minerals* -, Second. New York: Oxford University Press, 1997.

[4] E. S. Tsurko, P. Feicht, F. Nehm, *et al.*, „Large scale self-assembly of smectic nanocomposite films by doctor blading versus spray coating: Impact of crystal quality on barrier properties,“ *Macromolecules*, vol. 50, no. 11, pp. 4344–4350, 2017.

[5] A. Salazar, A. Mendioroz, and R. Fuente, „The strong influence of heat losses on the accurate measurement of thermal diffusivity using lock-in thermography,“ *Applied Physics Letters*, vol. 95, no. 12, p. 121 905, 2009.

[6] A. Mendioroz, R. Fuente-Dacal, E. Apiñaniz, and A. Salazar, „Thermal diffusivity measurements of thin plates and filaments using lock-in thermography,“ *Review of Scientific Instruments*, vol. 80, no. 7, p. 074 904, 2009.

[7] V. Singh, T. L. Bougher, A. Weathers, *et al.*, „High thermal conductivity of chain-oriented amorphous polythiophene,“ *Nature Nanotechnology*, vol. 9, no. 5, pp. 384–390, 2014.

[8] H. Hu, X. Wang, and X. Xu, „Generalized theory of the photoacoustic effect in a multilayer material,“ *Journal of Applied Physics*, vol. 86, no. 7, pp. 3953–3958, 1999.

[9] T. Still, *High Frequency Acoustics in Colloid-Based Meso- and Nanostructures by Spontaneous Brillouin Light Scattering*. Springer Berlin Heidelberg, 2010.

[10] C. Klieber, *Ultrafast photo-acoustic spectroscopy of super-cooled liquids*. Massachusetts Institute of Technology, 2010.

[11] S. P. Cheadle, R. J. Brown, and D. C. Lawton, „Orthorhombic anisotropy: A physical seismic modeling study," *Geophysics*, vol. 56, no. 10, pp. 1603–1613, 1991.

[12] M. Mah and D. R. Schmitt, „Experimental determination of the elastic coefficients of an orthorhombic material," *Geophysics*, vol. 66, no. 4, pp. 1217–1225, 2001.

[13] S. Cusack and A. Miller, „Determination of the elastic constants of collagen by brillouin light scattering," *Journal of Molecular Biology*, vol. 135, no. 1, pp. 39–51, 1979.

[14] M. Zgonik, P. Bernasconi, M. Duelli, *et al.*, „Dielectric, elastic, piezoelectric, electro-optic, and elasto-optic tensors of $BaTiO_3$ crystals," *Physical Review B*, vol. 50, no. 9, pp. 5941–5949, 1994.

[15] F. Mouhat and F.-X. Coudert, „Necessary and sufficient elastic stability conditions in various crystal systems," *Physical Review B*, vol. 90, no. 22, 2014.

[16] M. D. Losego, M. E. Grady, N. R. Sottos, D. G. Cahill, and P. V. Braun, „Effects of chemical bonding on heat transport across interfaces," *Nature Materials*, vol. 11, no. 6, pp. 502–506, 2012.

[17] W.-L. Ong, S. M. Rupich, D. V. Talapin, A. J. H. McGaughey, and J. A. Malen, „Surface chemistry mediates thermal transport in three-dimensional nanocrystal arrays," *Nature Materials*, vol. 12, no. 5, pp. 410–415, 2013.

15

Breathable and Flexible Polymer Membranes with Mechanoresponsive Electric Resistance

Qiang Gao,[1] Bernd A. F. Kopera,[2] Jian Zhu,[1] Xiaojian Liao,[1], Chao Gao,[3,4,5,6] Markus Retsch,[2] Seema Agarwal[1] and Andreas Greiner[1]

[1] Macromolecular Chemistry and Bavarian Polymer Institute, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

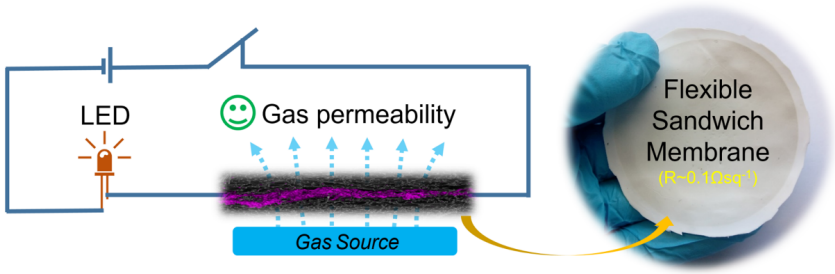
[2] Department of Chemistry, Physical Chemistry I, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

[3] MOE Key Laboratory of Macromolecular Synthesis and Functionalization, Department of Polymer Science and Engineering, Zhejiang University, Hangzhou 310027, P. R. China

[4] Institute of Applied Mechanics

[5] State Key Laboratory of Fluid Power and Mechatronic Systems, Zhejiang University, Hangzhou 310027, China

[6] Hangzhou Gaoxi Technology Co., Ltd., Hangzhou 310027, P. R. China



Accepted for publication in Advanced Functional Materials (adfm.201907555)

15.1 Abstract

Flexible low-resistance membranes play an important role in soft electronics as sensors for robotics, body movement monitoring, nanogenerators to collect kinetic energy from body movements, and flexible batteries. Despite great efforts, low-resistance mechanically stable large-dimensional membranes that tolerate very high deformability without sacrificing resistance, produce low joule heating, and allow passage of gases for human comfort are still looked for. Here, we provide one of the solutions by sandwiching a network of silver nanowires (AgNWs) between two highly porous electrospun thermoplastic polyurethane (TPU) membranes. The membranes are mechanically robust (both for bending and stretching) with a strong interface and large strain before breakage (more than 700 %). The sheet resistance is as low as $< 0.1(\pm 0.01)\,\Omega/\mathrm{sq}$, and changed to only $1.6(\pm 0.43)\,\Omega/\mathrm{sq}$ upon stretching to 100 % strain. The combination of polymer elasticity and the AgNW network structure provides a reversible change in resistance beyond 100 % strain. We employ a detailed thermographic analysis to in-situ image and characterize the AgNW network morphology during various stretched conditions. We believe that this flexible sandwich-like electrically conductive membrane is a good candidate for smart wearable devices and soft robots.

15.2 Introduction

Traditional electronics are typically composed of intrinsically heavy and rigid materials, like silicon, metals, and glass, that have extremely limited flexibility, stretchability, bendability, twistability, and impact resistance. There is great need for flexible and deformable circuits and electrodes with low electric resistance that maintain performance even at strained state because of the current rapid development of lightweight, wearable, and flexible electronic devices.[1]–[3] Efforts to construct flexible circuits and electrodes with low resistance focused on elastomeric electrically conductive materials. Examples include electrically conductive sponges,[4] hydrogels,[5] nonwovens,[6] films,[7] and fibers.[8]–[10] These materials are promising for energy production and storage,[11] actuators,[12] sensors,[13] electronic skin[14]–[16] and soft robotics.[17] However, obtaining flexible conductors with both high strain properties and low resistance is a challenge for design and production of flexible electronic devices.[18] Further, porosity is an important requirement in case electronic devices are meant for contact with skin. Air permeability is very important for the comfort of wearable devices. Achieving a combination of flexibility, elasticity, high conductivity, in porous membranes with good mechanical integrity is a continued search.

Presently, typical flexible electrodes are composed of an elastic substrate as the elastomer to support bending, stretching and twisting. The most commonly used materials are: polydimethylsiloxane (PDMS),[19],[20] thermoplastic polyurethane (TPU),[21] electrically conductive polymers,[22] carbon materials,[23],[24] metal nanoparticles[25] and metal nanowires.[26] Technically, low-resistance electrodes provide the advantages of less energy loss and joule heating. Silver is an excellent material for electrodes and is often used for flexible electrodes with low resistance.[27] In our previous work, we prepared electrospun polymer membranes with a metal-like conductivity of 7.5×10^5 S/m using a very low content of silver nanowires (AgNWs: 3.35 vol%) as an additive.[6] However, this kind of electrically conductive membrane is bendable but not sufficiently stretchable.

To prepare flexible membranes, printing fabrication techniques have been developed by transferring electrically conductive materials from a working substrate to another target substrate, such as PDMS, TPU and polyethylene terephthalate (PET).[28] Additionally, printing fabrication techniques provide a simple and versatile way to design and print patterns on substrate materials using computer control.[21],[29] In addition, microfluidic processing,[30] microchannel wetting patterning,[31] direct laser patterning,[32] shadow mask patterning[33] and photolithographic[34] methods have also been developed to fabricate flexible conductors. Other methods, such as filling of a AgNW network prepared by spray drying of a AgNW dispersion with PDMS was also used for making stretchable, conductive electrodes.[20] Nevertheless, the stretchable conductors prepared with these methods are all airtight due to their nonporous elastic substrate.

Alternatively, electrospinning is a promising technique for fabricating porous substrates with high conductivity, flexibility and air permeability. The good electropsinnability of many different polymer types also provides the advantage of tuning the membrane mechanical characteristics. For example, electrospun polyamide nanofiber nonwovens embedded with AgNW networks showed 50 % stretchability and a sheet resistance of 8.2 Ω/sq.[35] Electrospun poylurethane nonwoven porous substrate coated with AgNW provided highly stretchable membrane (more than 300 %) with high conductivity.[36]

Here, we present a flexible and breathable polymer membrane with bending/stretching stability and very low electrical resistance as a new electrode material. Also, we provide important fundamental studies regarding analysis on island formation, evolution of anisotropy and hot-spots, and thermal transport in such structures.

Our study shows

1. a simple preparation method for making such membranes which is applicable to large-dimension samples. We use a network of AgNWs as conductive layer sandwiched between two porous TPU fibrous nonwovens made by electrospinning. We improved the interface stability with polycaprolactone (PCL) hot glue in the form of short fibers dispersed in between the AgNW network.

2. Our membranes show a sheet resistance as low as $< 0.1\,\Omega/\mathrm{sq}$. The resistance does not significantly increase upon deformation (twisting, bending and stretching to approximately 100 % strain).

3. We also show a reversible change in the resistance upon stretching or bending with more than 100 % strain for several cycles. The AgNW networks reversibly breaks and reforms during stretching and releasing, respectively.

4. Our membranes are breathable. This allows the exchange of gases, which is important for human comfort.

5. Our membranes are thermal insulators comparable to porous polymers, despite their electrical conductivity.

We think that our membranes are a promising material for electrodes in smart textiles and other wearable devices.

15.3 Results and Discussion

The concept for the preparation of flexible and breathable electrically conductive membranes is illustrated in Figure 15.1. First, we prepared TPU nanofiber nonwovens via electrospinning. We then used the porous TPU membranes as a filter for the filtration of AgNWs ($17\,\mathrm{mg/mL}$, $126 \pm 10\,\mathrm{nm}$ in diameter and $18 \pm 4\,\mu\mathrm{m}$ in length, Figure 15.6) mixed with a PCL short-fiber dispersion ($1\,\mathrm{mg/mL}$, average aspect ratio $\approx$ 1000, Figure 15.7). The TPU nonwovens consist of randomly oriented fibers (the 2D order parameter is around 0.1) with an average diameter of $1.6 \pm 0.5\,\mu\mathrm{m}$ (Figure 15.2a and b). After filtration, the AgNWs and short PCL fibers generated a double network-type structure due to percolation (Figure 15.2c and d). Afterwards, we covered the AgNW/PCL layer with another layer of TPU nonwoven by electrospinning. Finally, we hot pressed the 3 layer stack at 75 °C for 30 min to melt the PCL and increase the adhesion of the layers (Figure 15.2e). The resulting three layer membranes are designated as TPU-AgNW/PCL-TPU. We

checked the presence of an AgNW layer sandwiched between the two TPU layers with PCL short fibers by EDX (Figure 15.2f). The cross-sectional SEM images of TPU-AgNW/PCL-TPU and TPU-AgNW-TPU (without the use of PCL short fibers for comparison) are shown in Figure 15.2g and h, respectively. The PCL short-fiber dispersion acts as a glue between the AgNW network and the TPU. The importance of providing a good adhesion between the fiber support and the conducting nanowires was also investigated by Jiang *et al.*[36]

Omitting the PCL left us with a loose interface between the layers. Figure 15.8 shows the morphology of the middle layer composed only of PCL short fibers after hot pressing at 75 °C for 30 min. Patches of molten fibers can be observed.

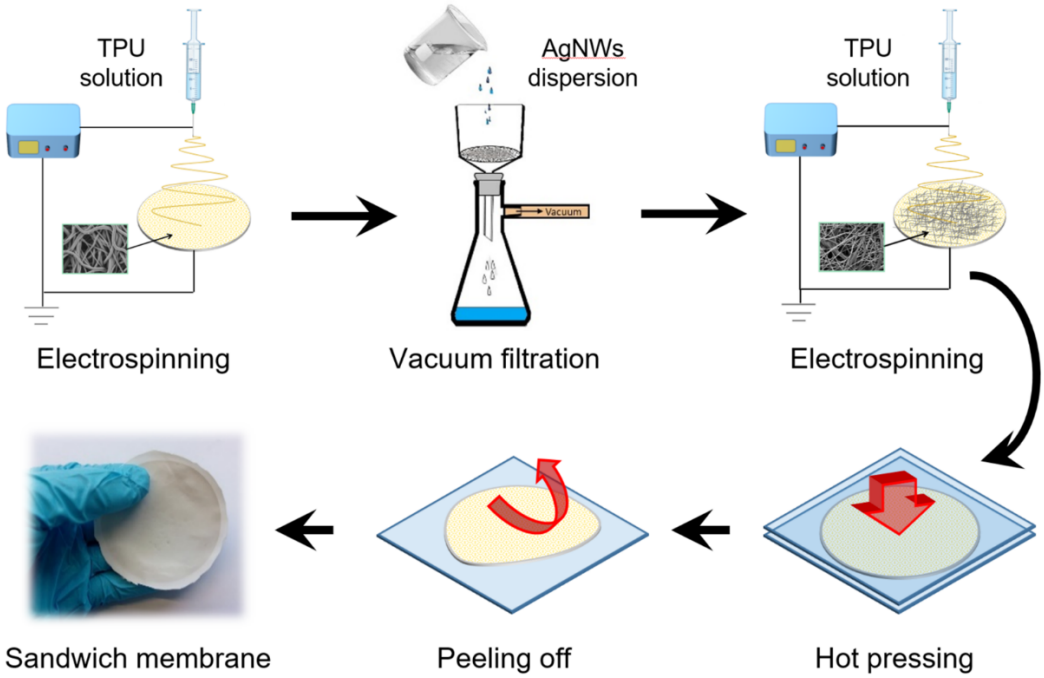


Fig. 15.1.: Schematic of preparing the TPU-AgNW/PCL-TPU membrane by sandwiching a network of AgNWs between TPU layers.

The pore size distribution (Figure 15.9) of the sandwich membrane without AgNWs is mainly distributed at 2.8 µm with 80 % porosity. Upon formation of the AgNW network layer, the average pore size was 1.5 µm to 0.9 µm. Additionally, the thickness of the sandwich membrane increased slightly with increasing volume of the AgNW solution, from 0.18 ± 0.4 mm to 0.27 ± 0.02 mm, as shown in Figure 15.10.

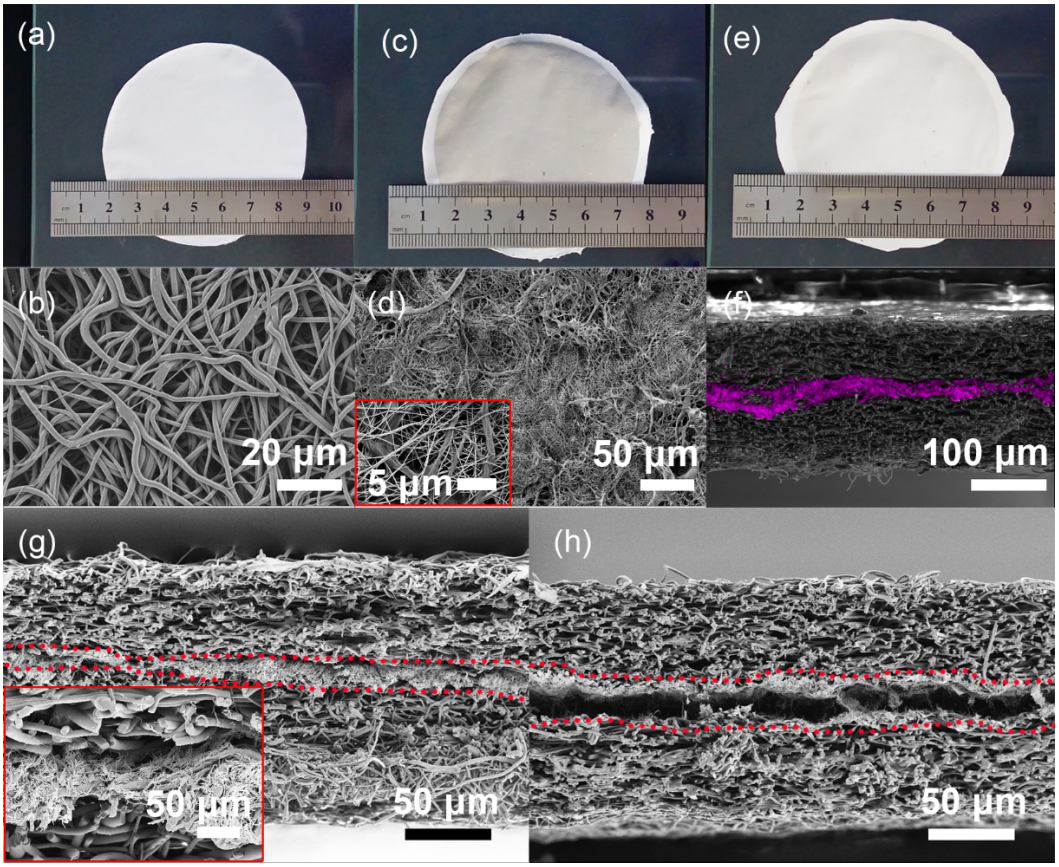


Fig. 15.2.: Photograph (a) and SEM image (b) of the electrospun TPU nonwoven. Photograph (c) and SEM image (d) of the electrospun TPU nonwoven layered with AgNWs and PCL after heating. Photograph of as-prepared sandwich membrane of TPU-AgNW/PCL-TPU (e). Cross-section of EDX overlapped SEM image of as-prepared sandwich membrane of TPU-AgNW/PCL-TPU with PCL short fibers (f). Cross-section of TPU-AgNW/PCL-TPU (g) and TPU-AgNW-TPU (h).

Different amounts of the AgNW dispersion were used to investigate the influence of the AgNW concentration on the air permeability and electrical properties. The relevant parameters are illustrated in Table 15.1. The sheet resistance was as high as $6.1 \pm 0.3 \times 10^8\,\Omega/\mathrm{sq}$ without an AgNW layer. The sheet resistance decreased to as low as $0.09\,\Omega/\mathrm{sq}$ upon increasing the amount of AgNWs because of the formation of a dense network of AgNWs. Even the use of a small amount of AgNWs (1.2 wt.%) significantly decreased the resistance to $\approx 1\,\Omega/\mathrm{sq}$. Further reduction in the amount of AgNW to ≈ 0.5 wt.% led to a sharp increase in the sheet resistance ($2.4 \pm 3.1 \times 10^6\,\Omega/\mathrm{sq}$) indicating the amount of AgNW required for making a network by percolation somewhere between 0.5–1.2 wt.%. In general, the sheet resistance of TPU-AgNW/PCL-TPU remains almost unchanged at 100 °C in air for more than 60 h (Figure 15.11).

The TPU-AgNW/PCL-TPU membranes also possess good gas permeability (Figure 15.12), as demonstrated by the penetration of carbon dioxide (CO_2) gas, visualized by the pink base solution containing phenolphthalein indicator (Video S1).

Tab. 15.1.: The relevant parameters of flexible sandwich-like electrically conductive membranes with different contents of AgNWs.

Sample	TPU spinning solution (16 wt.%) [ml]	PCL dispersion (1 g/l) [ml]	AgNW dispersion (17 mg/ml) [ml]
TPU-PCL-TPU	1	10	0
TPU-0.1AgNW/PCL-TPU	1	10	0.1
TPU-0.25AgNW/PCL-TPU	1	10	0.25
TPU-0.5AgNW/PCL-TPU	1	10	0.5
TPU-1.0AgNW/PCL-TPU	1	10	1.0
TPU-1.5AgNW/PCL-TPU	1	10	1.5
TPU-2.0AgNW/PCL-TPU	1	10	2.0

Sample	Density [g/m²]	Content of AgNWs [wt.%]	Sheet resistance of the AgNW layer [Ω/sq]
TPU-PCL-TPU	-	-	$6.1 \pm 0.3 \times 10^8$
TPU-0.1AgNW/PCL-TPU	0.51	0.5 ± 0.6	$2.4 \pm 3.1 \times 10^6$
TPU-0.25AgNW/PCL-TPU	1.28	1.2 ± 0.5	1.14 ± 0.15
TPU-0.5AgNW/PCL-TPU	2.56	2.0 ± 0.6	0.39 ± 0.07
TPU-1.0AgNW/PCL-TPU	5.12	3.5 ± 0.4	0.20 ± 0.02
TPU-1.5AgNW/PCL-TPU	7.68	7.9 ± 1.2	0.16 ± 0.04
TPU-2.0AgNW/PCL-TPU	10.24	8.5 ± 1.7	0.09 ± 0.002

The sandwich membranes exhibited stress at break of more than 8 MPa. The TPU without AgNWs had a breaking stress of ≈ 10 MPa. We observed excellent stretchability for the TPU-AgNW/PCL-TPU membranes. The membranes with different amounts of AgNWs can be stretched to over 700 % (Figure 15.3a). The membranes showed 10 % and 30 % creep after stretching for 100 cycles to 50 % and 100 % strain, respectively (Figure 15.13). We measured the resistance as a function of % strain (Figure 15.3b). TPU-AgNW/PCL-TPU can tolerate considerable strains with a moderate increase in resistance depending upon the amount of AgNWs. The sample with the highest content of AgNWs (TPU-2.0AgNW/PCL-TPU; 8.5 wt.%)

changed its resistance by only two orders of magnitude at a strain of ≈ 150 %. In contrast, the sample with the fewest AgNWs (TPU-0.25AgNW/PCL-TPU; 1.2 wt.%) showed a comparable increase in resistance already at a strain of only 60 %. The network was sparse due to fewer AgNWs in TPU-0.25AgNW/PCL, which led to a higher initial sheet resistance and stronger increase in resistance during stretching.

Figure 15.3c shows the morphologies of the AgNW layer after 0 %, 100 %, and 150 % stretching and recovering to 0 % again for the TPU-2.0AgNW/PCL sample (8.5 wt.% AgNWs). When the membrane was stretched up to 100 %, hardly any cracks could be observed in the AgNW layer. However, upon further stretching to approximately 150 %, large cracks (bright area) occurred, causing a significant increase in resistance. The cracks visualize that the AgNW breaks up in an island-like fashion. The formation of island-like structures is confirmed by EDX measurement. Ag-rich patches are indicated in purple in Figure 15.3d. The separated islands and cracks restricted the flow of electricity and in turn created hot spots (Figure 15.14) by joule heating at the bottle necks. Such hotspots can be imaged by infrared thermography when applying a DC current (Figure 15.3e). The hotspots shift with increasing strain and highlight the dynamic changes within the AgNW network upon stretching (see video in the S2). Depending on the overall resistance of the composite sample and local bottle necks this can lead to substantial Joule heating resulting in a fatal failure of the sandwich structure at high strains. For low power applications and in the light of practical strains for the case of wearable electronics our sandwich structure will operate in a safe window with low Joule heating.

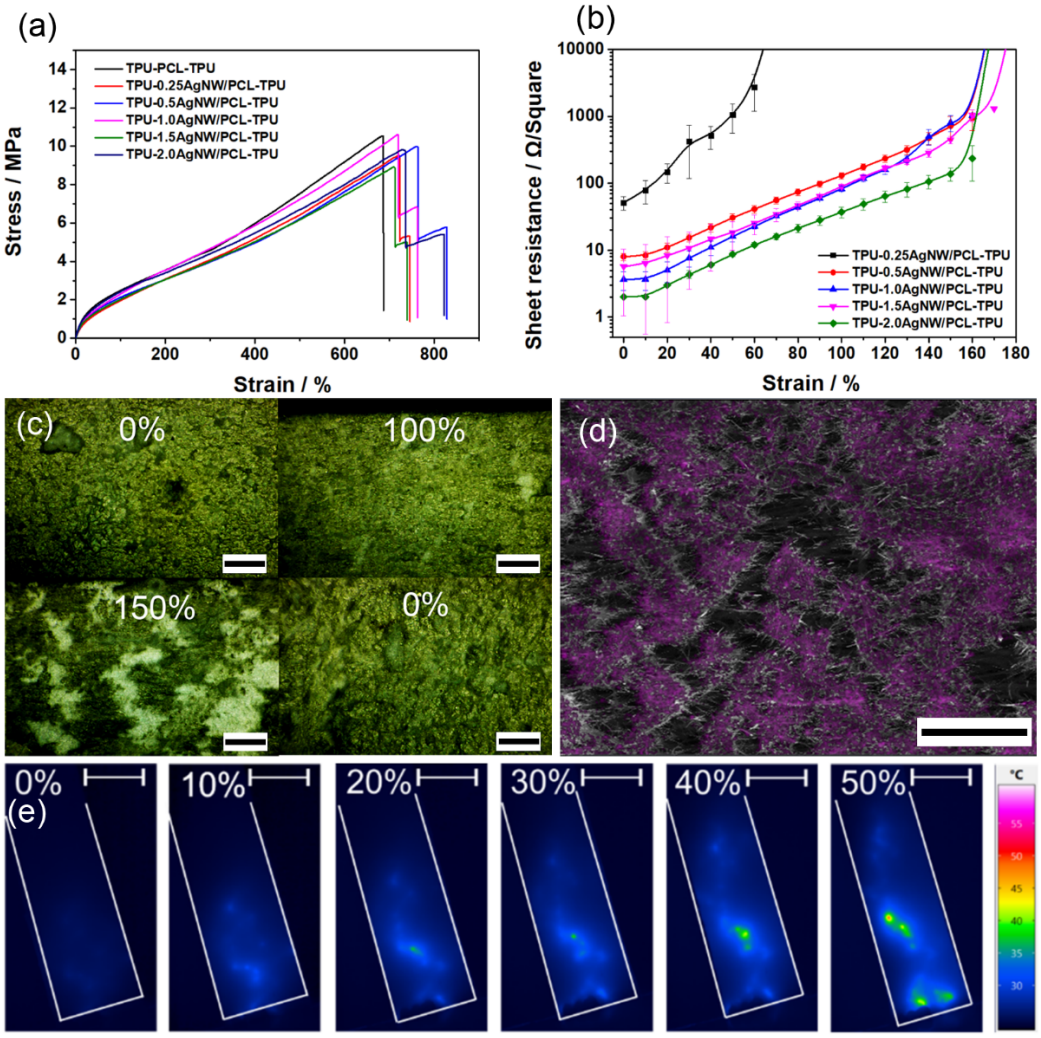


Fig. 15.3.: Stress-strain curves of sandwich membranes with different amounts of AgNWs (a). sheet resistance of sandwich membranes with different amounts of AgNWs with different strains (b). Optical images of TPU-2.0AgNW/PCL under different stains (c). Note the appearance of white spots at 150 % strain due to cracks in the AgNW layer. (scale bar: 500 µm) EDX-SEM images of TPU-2.0AgNW/PCL at 100 % strain (d). (scale bar: 50 µm) Ag is indicated in purple. Thermography images of the TPU sandwich sample upon stretching (e). White lines indicate the film boundary. A constant current of 2 mA passes through the sample. Note the appearance and shift of hotspots due to local bottlenecks for the electric current created by the cracks in the AgNW layer. (scale bars: 3 mm)

The morphological changes of the AgNW network upon stretching cannot be directly assessed, since they are embedded between two layers of TPU fibers. We, therefore, employ direction dependent lock-in thermography to gain a better insight on the composite morphology. Using a line laser as a heat source, we are able to probe the overall thermal diffusivity in such a composite sample along different orientations relative to the stretching direction. We see in Figure 15.4a–c that the TPU fibers align expectedly parallel to the stretching direction. Image analysis of these laser scanning microscopy images reveal that the fiber orientation is already complete at a strain of ≈ 100 %.

The 2D order parameter (S_{2D}) measures how well the fibers are aligned. The order parameter is given by

$$S_{2D} = \frac{2}{N} \cdot \left[\sum_{i=1}^{N} \left(\cos\left(\alpha_i - \bar{\alpha}\right) \right)^2 \right] - 1 \tag{15.1}$$

where N is the number of stretched fiber orientations in the SEM image, α_i is the angle between the fiber and the horizontal axis, and ($\bar{\alpha}$) is the average angle.[37] The 2D order parameter of pure TPU remains constant at a value close to one after 100 % strain(Figure 15.4d).

For pure TPU the thermal diffusivity increases along the stretching direction and decreases perpendicular to it (Figure 15.4e and 15.4f). Since TPU is a thermal insulator the absolute thermal diffusivity remains at a low level. The evolution of anisotropy can be explained by the geometric changes, where the fibers start to align, forming an anisotropic fiber network (Figure 15.15). Heat travels along these oriented fibers faster than perpendicular to it, resulting in a high anisotropy ratio (Figure 15.4g). The geometric alignment can be accompanied by a molecular orientation of the polymer chains within the fibers, leading to an additional increase in the thermal anisotropy.[38]

The thermal diffusivity at 0 % strain is higher for the AgNW-TPU sample (TPU-2.0AgNW/PCL-TPU) than for the pure TPU support structure. This can be understood by the additional thermal transport via electronic conduction through the AgNW network. The angular dispersion is rather shallow demonstrating an isotropic temperature spreading in the unstretched and disordered fiber network. Stretching of the fiber network immediately introduces anisotropy to the thermal diffusivity. For AgNW-TPU also some degree of anisotropy evolves (Figure 15.4h and 15.4i), but the thermal diffusivity overall decreases compared to the unstretched state (Figure 15.16). This can be understood by a decrease in electronic thermal conductivity in the AgNW layer. The overall thermal diffusivity remains, nevertheless, higher compared to pure TPU. Simultaneously, also for AgNW-TPU samples, we find an evolution of anisotropic heat spreading (Figure 15.4g). Overall, stretching of the AgNW-TPU sample leads to a smaller amount of anisotropy compared to the pure TPU network structure (Figure 15.4g). This corroborates our previous findings that the AgNW network is torn into isolated island structures upon stretching. Thermal transport within these island-like structures happens in an isotropic fashion, which shows that the anisotropic orientation of the TPU fiber network is only partially transferred to the AgNW network and counteracted by the evolution of gaps between such islands parallel to the stretching direction. We corroborate this interpretation with additional orientation analysis of AgNW-TPU samples upon stretching, summarized in Figures 15.17 and 15.18.

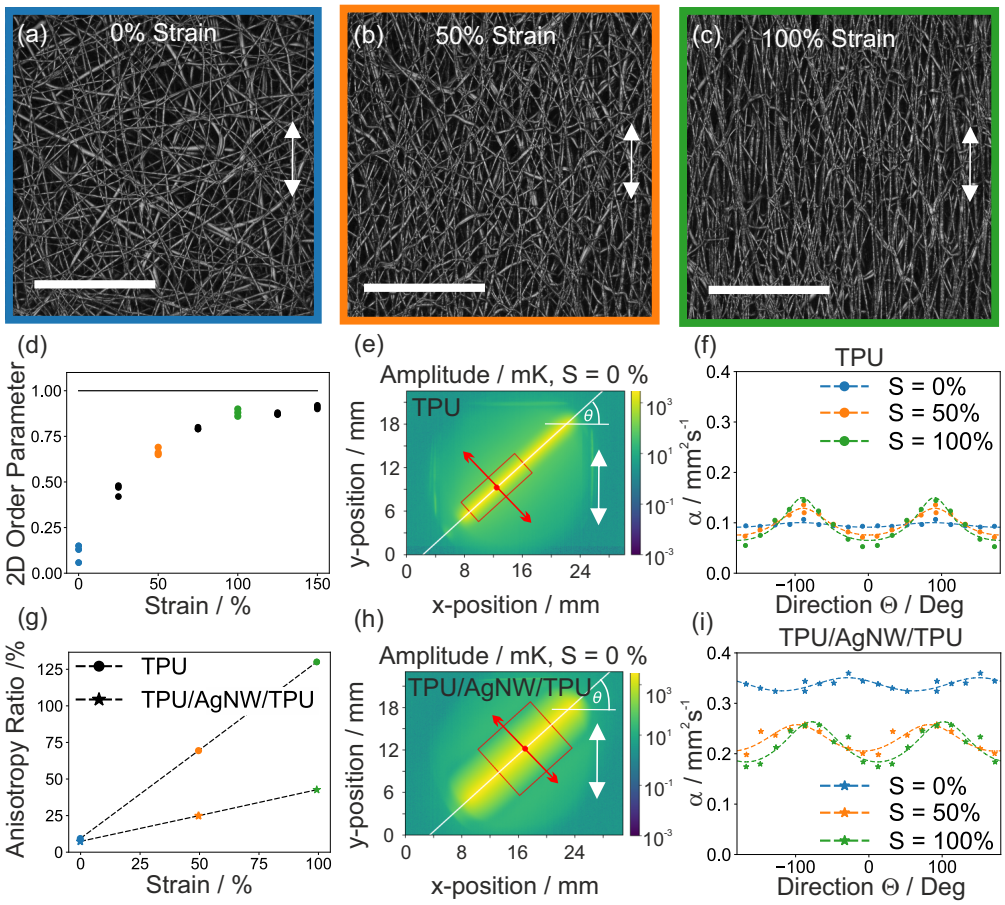


Fig. 15.4.: Laser scanning microscopy images of TPU fibers on the surface of a TPU nonwoven. (a-c) (scale bars: 50 µm). The fibers align in the stretching direction (white arrows) with increasing strain, S. The 2D order parameter for the fiber orientation, derived from laser scanning microscopy images, increases from 0.1 to 0.9 with increasing strain (d). Temperature amplitude as a function of position on the surface of a pure TPU (e). Thermal diffusivity as a function of direction (0°is horizontal) and strain for a pure TPU (f). The TPU nonwovens are stretched vertically (+/- 90° direction). Anisotropy ratios for the thermal diffusivity. (g) The pure TPU shows a stronger anisotropy than the TPU-2.0AgNW/PCL-TPU. Temperature amplitude as a function of position on the surface of TPU-2.0AgNW/PCL-TPU (h) film with 0 % strain. Thermal diffusivity as a function of direction (0°is horizontal) and strain for TPU-2.0AgNW/PCL-TPU (i) film. The films are stretched vertically (+/- 90°direction)

We now focus on the reversibility of the stretching/compression process and its implications for electromobility measurements. The membranes, even with very high amounts of AgNWs (TPU-2.0AgNW/PCL-TPU), showed excellent stretchability and bending stability. The samples were subjected to 100 cycles of stretching (Figure 15.5a) and bending (Figure 15.5b). The resistance change can be described by:

$$RC = \frac{R}{R_0} \tag{15.2}$$

where R is the time dependent resistance and R_0 is the initial resistance. Both R and R_0 were measured after stretching and bending tests to prove the stability after deformation. The RC of TPU-0.25AgNW/PCL-TPU with only 1.2 wt.% of AgNWs significantly increased after only 60 stretching cycles, the RC was > 20000 under 50 % strain. However, the same sample showed a much better bending stability (bending curvature from 0° to 150°) without a significant change in resistance. The RC of TPU-0.25AgNW/PCL-TPU after 100 bending cycles was merely 1.32. The sample TPU-2.0AgNW/PCL-TPU with a dense network of AgNWs showed both excellent bending and stretching stability. No significant change in the RC was observed for bending tests. During strain testing RC increased by only a factor of ≈ 20 after 100 cycles. It is important to note that our method of fabrication is particularly suitable to adjust the AgNW concentration in a simple manner. This can even be achieved locally by filtration through a mask. In Figure 15.19, we demonstrate, that the local variation of the AgNW concentrations leads to locally confined heating powers. Hence the AgNW concentration, sheet resistance and response to stretching or bending can be laterally adjusted and optimized for a given application.

We also investigated the changes to the AgNW upon repeated stretching and unstretching with thermography. This measurement was conducted using a sample of intermediate AgNW concentration (TPU-1.0AgNW/PCL-TPU, 3.5 wt.%. Upon stretching (50 % strain) we measured an increase in electrical resistance, which translated into hot spot areas. Consistent with our previous analysis such islands of AgNW network (Figure 15.3d) almost completely disappear upon release of the strain. In the following stretching cycles these hot-spots reform and increase in number and evolved heat. With this intermediate AgNW concentration, a disconnection between the well conducting AgNW islands happens easier and the re-formation of the intact network becomes less complete, in contrast to the better performing samples with a higher amount of AgNWs.

An LED was utilized to demonstrate the flexibility of the prepared sandwich membrane under various mechanical loads during working conditions. TPU-0.25AgNW/PCL-TPU was connected to a circuit, as shown in Figure 15.5e, and was bent and twisted. The LED maintained almost the same brightness. However, after 50 % strain stretching, the brightness of the LED could hardly be observed. This behavior corroborates the formation of a very high RC of TPU-0.25AgNW/PCL-TPU after stretching. When using a more highly concentrated AgNW network (TPU-2.0AgNW/PCL-TPU) twisting, bending, and stretching (50 % strain) are tolerated and the brightness of the LED remained almost unchanged. In addition, detailed information is shown in Video S3.

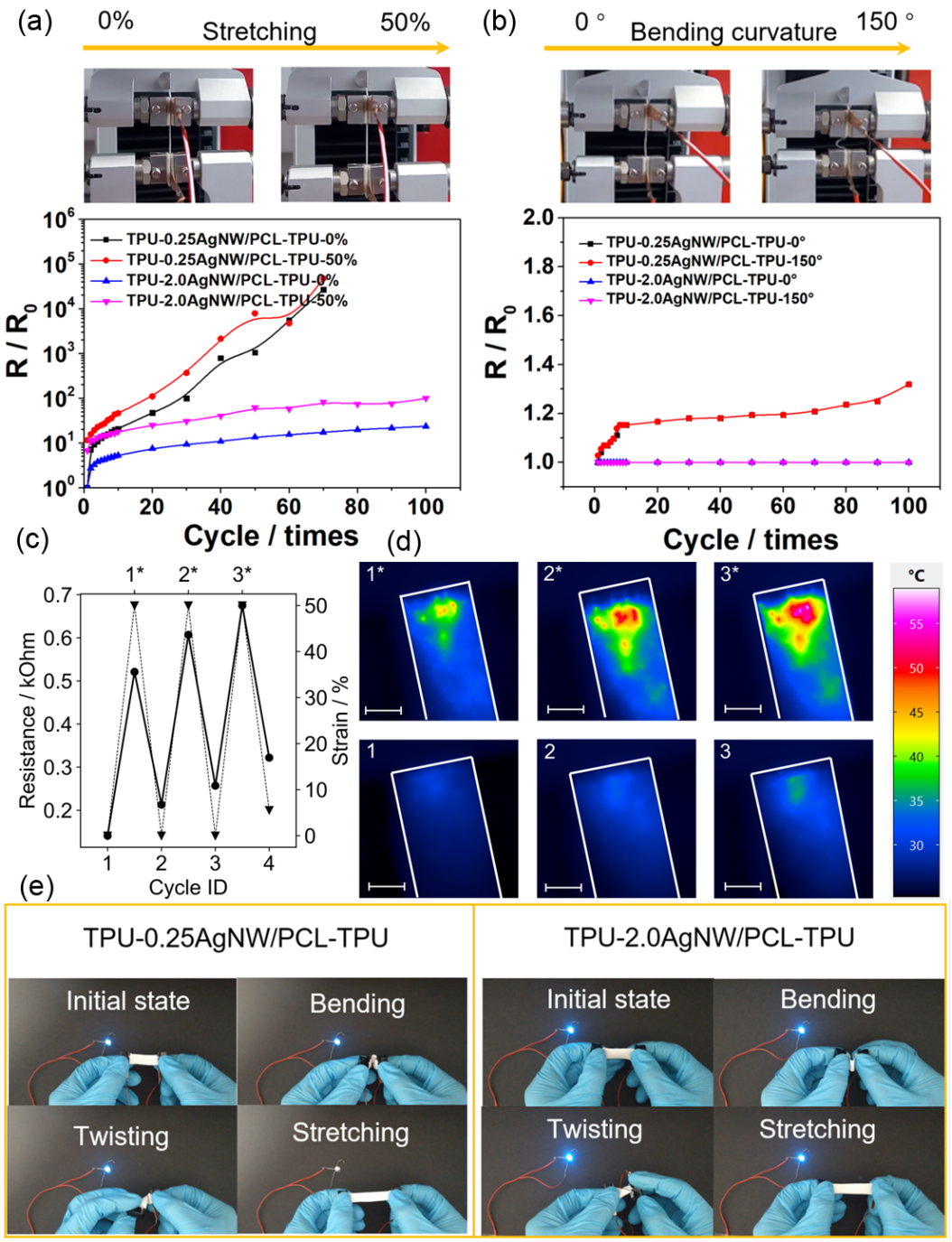


Fig. 15.5.: Resistance change after 100-cycle stretching test (50 % strain) (a). Resistance change ratio after 100-cycle bending test (bending curvature from 0° to 150°) (b). Noticing that the bending angle of samples was close to 150° when distance between two clips decreased to 50 %. Electric resistance (dots) and strain (triangles) during the cycling for the thermographic imaging. (c) Thermographic images of a TPU-1.0AgNW/PCL-TPU film during cycling to 50 % strain. (d) (scale bars: 5 mm); A constant current of 10 mA passes through the film. The hotspots occur mainly in the stretched state and at different positions from cycle to cycle. Digital image of LED brightness by different operation of nonwoven TPU-0.25AgNW/PCL-TPU and TPU-2.0AgNW/PCL-TPU (e).

The use of TPU-2.0AgNW/PCL-TPU as resistive-type strain sensor in a preliminary test for monitoring finger and wrist bending and unbending movements for several times (Figure 15.20) worked very well. The sensitivity and precision would be further investigated and other application areas will also be explored.

15.4 Conclusion

In conclusion, we proposed a new strategy to fabricate sandwich-like electrically conductive membranes with very low electric resistance. Our material consists of an AgNW network sandwiched between two porous electrospun TPU nonwovens. PCL short fibers act as glue and provide a strong interface between the three layers. Our membranes show good bending and stretching stability, high stretchability (breaking elongation more than 700 %) and very low initial electric resistance, as low as $< 0.1\,\Omega/\mathrm{sq}$ depending upon the amount of AgNWs. This composite structure shows potential for use as a strain sensor to monitor body movement. Additionally, the membrane possesses gas permeability and low thermal diffusivity. Furthermore, our material has the potential to be integrated into smart wearable devices to collect body movement signals using a connected PC or mobile phone in the future.

15.5 Experimental Section

15.5.1 Materials

In this study, TPU (Desmopann DP 2590, Bayer Materials Science, M_n 88900, M_w 145000), PCL (Capa 6800, M_w 120000, Perstorp UK Ltd.), poly(vinylpyrrolidone) (PVP K30, M_w 40000, Sigma-Aldrich), ethylene glycol (p.a. $\geq$ 99.5%, Fluka), silver nitrate ($AgNO_3$, p.a. 99.9999%, Sigma-Aldrich), iron chloride ($FeCl_3$, p.a. 98.0 %, Sigma-Aldrich), sodium chloride (NaCl, p.a. 99.0 %), and dimethylformamide (DMF, p.a. 99.8 %) were used as obtained. Other solvents were distilled before use. The AgNWs were synthesized using a previous method.[6]

15.5.2 Analytical instruments and methods

A Zeiss LEO 1530 (Jena, Germany; Schottky field emission cathode) was used for SEM characterization of the AgNWs and their corresponding networks. EDX was performed using a Zeiss Ultra Plus (Jena, Germany; acceleration voltage of 10 kV). TEM measurements were performed using an elastic bright-field transmission electron microscope (TEM, Zeiss 922 Omega EFTEM) at a voltage of 200 kV. ImageJ

software was used to determine the AgNW diameter. An average of 100 AgNWs was taken for the diameter calculation.

The AgNW content was determined from thermogravimetric analysis (TGA Libra F1, Netzsch, Selb, Germany). A heating rate of 20 °C/min was used. The measurements were performed in synthetic air.

Pore size measurements were performed with a PSM 165/H (Dresden, Germany) to determine the pore size. Topor was used as the standard test liquid (surface tension = 16.0 mN/m). The sample holder had a diameter of 11 mm, and a flow rate of up to 70 L/min was applied. At least three measurements were carried out for each sample, and an average was taken.

The gas permeability test was performed with a homemade unit as described in our previous work in reference [5]. The membrane was fixed at the bottleneck with the cap. A rubber tube (airtight) was used to connect to another glass bottle containing solid carbon dioxide. The glass bottle fixing the nonwovens at the bottleneck was immersed in a phenolphthalein/limewater mixture. The carbon dioxide permeated through the membrane and discolored the aqueous basic solution containing phenolphthalein indicator.

Tensile tests were carried out (sample length 2 cm, width 2 mm) using a tensile tester (ZwickiLine Z0.5; BT1-FR0.5TN. D14; Zwick/Roell, Germany). The following parameters were used: clamping length 1 cm, crosshead speed 10 mm/min, temperature 20 °C and pretension 0.01 N/mm. The thickness of the samples was measured with a screw micrometer. The 100-cycle stretching test with 50 % and 100 % strain was also measured with the abovementioned measuring conditions.

Sheet resistance measurements (four-point measurements) were performed using a Keithley 2420 high-current source meter coupled with a Signatone SYS-301. The sheet resistance was measured ten times for each sample.

A tensile tester (ZwickiLine Z0.5; BT1-FR0.5TN. D14; Zwick/Roell, Germany) was employed to control the stretching distance and bending degree, and a digital multimeter (EMOS Multimeter EM391) connected to measure the sample with a bronze conductor was employed to measure the stretching resistance and bending resistance. The size of the sample and the measuring parameters used here were the same as those used in the tensile test.

Thermography imaging was performed with an Infratec VarioCAM HD research IR camera (7.5 – 14 µm). The spatial resolution is 29 µm at closest proximity. All measurements were performed in air. The resistance was measured simultaneously

with a Keithley 2400 Sourcemeter. The Sourcemeter acted as a constant current source during the four wire resistance measurements. A home built stretching apparatus was used to elongate the samples. The strain was determined from the length change.

Thermal diffusivity was measured with our own, self-built Lock-In thermography setup.[39],[40] Briefly, the films are mounted on a stretchable holder inside a vacuum chamber. An intensity modulated line laser is focused onto the back side of the sample. The intensity modulation creates temperature oscillations inside the thin film. We then detect the surface temperature with a Infratec VarioCAM HD research IR camera. The samples were coated with 25 nm of carbon on both sides to enhance laser absorption and IR emission. The IRBIS active online software from the IR Camera detects amplitude and phase of the temperature fluctuations on the sample surface. The thermal diffusivity is extracted from the material by fitting the linearized amplitude and phase signal, which spreads perpendicular to the line laser excitation.

Optical microscopy was performed on a LEXT Olympus OLS 5000 confocal laser scanning microscope. A 100x magnification lense with a working distance of 300 µm was used. Image analysis was conducted with the Matlab code published by Persson *et al.*[37]

15.5.3 Preparation of TPU nonwoven and PCL short fibers

Electrospinning of 16 wt% TPU in DMF/THF (80/20 wt.%) was carried out to obtain TPU nonwovens with randomly oriented fibers. A round 325 mesh stain-less sieve (70 mm in diameter) at an electrode distance of 15 cm was used as the collector. Other spinning conditions were as follows: positive voltage = 15 kV; temperature = 25 °C; relative humidity = 35 %. For spinning PCL, a 15 wt.% solution in a mixture of THF/DMF (70/30 wt.%) was used. Other electrospinning conditions were the same as those used for TPU spinning. The dispersion (1 g/L) of short PCL nanofibers was obtained as reported previously by us.[5]

15.5.4 Preparation of sandwich TPU-AgNW/PCL-TPU membranes

The preparation procedure for the sandwich membranes is shown schematically in Figure 15.1. First, a TPU nonwoven prepared according to the procedure described above via electrospinning was used as a filter. Then, a dispersion of AgNWs and PCL short fibers was filtered. The AgNWs and PCL short fibers formed a conductive

layer on top of the TPU nanofiber nonwoven. The fibers were dried under vacuum for 10 min at a pressure of 0.1 mbar. Later, another layer of TPU nanofibers was electrospun on top of the nonwoven with AgNWs and PCL short fibers to construct a flexible, sandwich-like, electrically conductive nonwoven. To induce thermal annealing, the achieved sandwich-like, electrically conductive nonwovens were pressed between two glass plates and heated at a temperature of 75 °C for 30 min to melt the PCL fibers and bond the two layers of TPU nonwovens and AgNWs together.

Acknowledgment

Qiang Gao would like to thank the China Scholarship Council for awarding a fellowship for carrying out PhD research in Germany in the lab of Prof. Andreas Greiner. Bernd A. F. Kopera acknowledges support from the Elite Network of Bavaria. Financial support by DFG (project GR 972/46-2) and Volkswagen Foundation is acknowledged.

15.6 References

[1] E. J. Markvicka, M. D. Bartlett, X. Huang, and C. Majidi, „An autonomously electrically self-healing liquid metal–elastomer composite for robust soft-matter robotics and electronics,“ *Nature Materials*, vol. 17, no. 7, pp. 618–624, 2018.

[2] Y. J. Hong, H. Jeong, K. W. Cho, N. Lu, and D.-H. Kim, „Wearable and implantable devices for cardiovascular healthcare: From monitoring to therapy based on flexible and stretchable electronics,“ *Advanced Functional Materials*, vol. 29, no. 19, p. 1 808 247, 2019.

[3] S. Huang, Y. Liu, Y. Zhao, Z. Ren, and C. F. Guo, „Flexible electronics: Stretchable electrodes and their future,“ *Advanced Functional Materials*, vol. 29, no. 6, p. 1 805 924, 2018.

[4] S. Jiang, B. Uch, S. Agarwal, and A. Greiner, „Ultralight, thermally insulating, compressible polyimide fiber assembled sponges,“ *ACS Applied Materials & Interfaces*, vol. 9, no. 37, pp. 32 308–32 315, 2017.

[5] J. Liao, C. Zhu, B. Gao, *et al.*, „Multiresponsive nanoparticles: Multiresponsive elastic colloidal crystals for reversible structural color patterns,“ *Advanced Functional Materials*, vol. 29, no. 39, p. 1 970 271, 2019.

[6] S. Reich, M. Burgard, M. Langner, *et al.*, „Polymer nanofibre composite nonwovens with metal-like electrical conductivity,“ *npj Flexible Electronics*, vol. 2, no. 1, 2018.

[7] Y. Zeng, T. Li, Y. Yao, *et al.*, „Thermally conductive reduced graphene oxide thin films for extreme temperature sensors,“ *Advanced Functional Materials*, p. 1 901 388, 2019.

[8] W. Weng, P. Chen, S. He, X. Sun, and H. Peng, „Smart electronic textiles,“ *Angewandte Chemie International Edition*, vol. 55, no. 21, pp. 6140–6169, 2016.

[9] S. Lee, S. Shin, S. Lee, *et al.*, „Ag nanowire reinforced highly stretchable conductive fibers for wearable electronics,“ *Advanced Functional Materials*, vol. 25, no. 21, pp. 3114–3121, 2015.

[10] J. Lee, B. L. Zambrano, J. Woo, K. Yoon, and T. Lee, „Recent advances in 1d stretchable electrodes and devices for textile and wearable electronics: Materials, fabrications, and applications,“ *Advanced Materials*, p. 1 902 532, 2019.

[11] R. Cao, X. Pu, X. Du, *et al.*, „Screen-printed washable electronic textiles as self-powered touch/gesture tribo-sensors for intelligent human–machine interaction,“ *ACS Nano*, vol. 12, no. 6, pp. 5190–5196, 2018.

[12] M. Amjadi and M. Sitti, „Self-sensing paper actuators based on graphite-carbon nanotube hybrid films,“ *Advanced Science*, vol. 5, no. 7, p. 1 800 239, 2018.

[13] R. Tong, G. Chen, D. Pan, *et al.*, „Highly stretchable and compressible cellulose ionic hydrogels for flexible strain sensors,“ *Biomacromolecules*, vol. 20, no. 5, pp. 2096–2104, 2019.

[14] A. Chortos, J. Liu, and Z. Bao, „Pursuing prosthetic electronic skin,“ *Nature Materials*, vol. 15, no. 9, pp. 937–950, 2016.

[15] S. Wang, J. Y. Oh, J. Xu, H. Tran, and Z. Bao, „Skin-inspired electronics: An emerging paradigm,“ *Accounts of Chemical Research*, vol. 51, no. 5, pp. 1033–1045, 2018.

[16] Y. Lee, J. Park, A. Choe, *et al.*, „Mimicking human and biological skins for multifunctional skin electronics,“ *Advanced Functional Materials*, p. 1 904 523, 2019.

[17] C. Majidi, „Soft-matter engineering for soft robotics,“ *Advanced Materials Technologies*, p. 1 800 477, 2018.

[18] G. Chen, N. Matsuhisa, Z. Liu, *et al.*, „Plasticizing silk protein for on-skin stretchable electrodes,“ *Advanced Materials*, vol. 30, no. 21, p. 1 800 129, 2018.

[19] T.-H. Kim, C.-S. Lee, S. Kim, *et al.*, „Fully stretchable optoelectronic sensors based on colloidal quantum dots for sensing photoplethysmographic signals," *ACS Nano*, vol. 11, no. 6, pp. 5992–6003, 2017.

[20] H.-S. Liu, B.-C. Pan, and G.-S. Liou, „Highly transparent AgNW/PDMS stretchable electrodes for elastomeric electrochromic devices," *Nanoscale*, vol. 9, no. 7, pp. 2633–2639, 2017.

[21] A. D. Valentine, T. A. Busbee, J. W. Boley, *et al.*, „Hybrid 3D printing of soft electronics," *Advanced Materials*, vol. 29, no. 40, p. 1 703 817, 2017.

[22] H. H. Shi, N. Khalili, T. Morrison, and H. E. Naguib, „Self-assembled nanorod structures on nanofibers for textile electrochemical capacitor electrodes with intrinsic tactile sensing capabilities," *ACS Applied Materials & Interfaces*, vol. 10, no. 22, pp. 19 037–19 046, 2018.

[23] F. Sarker, P. Potluri, S. Afroj, *et al.*, „Ultrahigh performance of nanoengineered graphene-based natural jute fiber composites," *ACS Applied Materials & Interfaces*, vol. 11, no. 23, pp. 21 166–21 176, 2019.

[24] C. Zhu, A. Chortos, Y. Wang, *et al.*, „Stretchable temperature-sensing circuits with strain suppression based on carbon nanotube transistors," *Nature Electronics*, vol. 1, no. 3, pp. 183–190, 2018.

[25] M. Tavakoli, M. H. Malakooti, H. Paisana, *et al.*, „EGaIn-assisted room-temperature sintering of silver nanoparticles for stretchable, inkjet-printed, thin-film electronics," *Advanced Materials*, vol. 30, no. 29, p. 1 801 852, 2018.

[26] H. Lee, M. Kim, I. Kim, and H. Lee, „Flexible and stretchable optoelectronic devices using silver nanowires and graphene," *Advanced Materials*, vol. 28, no. 22, pp. 4541–4548, 2016.

[27] P.-C. Hsu, X. Liu, C. Liu, *et al.*, „Personal thermal management by metallic nanowire-coated textile," *Nano Letters*, vol. 15, no. 1, pp. 365–371, 2014.

[28] S. J. Benight, C. Wang, J. B. Tok, and Z. Bao, „Stretchable and self-healing polymers and devices for electronic skin," *Progress in Polymer Science*, vol. 38, no. 12, pp. 1961–1977, 2013.

[29] K. Arapov, E. Rubingh, R. Abbel, *et al.*, „Conductive screen printing inks by gelation of graphene dispersions," *Advanced Functional Materials*, vol. 26, no. 4, pp. 586–593, 2015.

[30] S. Kustra, H. Wu, S. Basu, G. K. Rohde, and C. J. Bettinger, „High-throughput arrays for rapid characterization of solution-processable transparent conducting electrodes," *Small*, vol. 8, no. 24, pp. 3746–3751, 2012.

[31] B.-R. Yang, W. Cao, G.-S. Liu, *et al.*, „Microchannel wetting for controllable patterning and alignment of silver nanowire with high resolution," *ACS Applied Materials & Interfaces*, vol. 7, no. 38, pp. 21 433–21 441, 2015.

[32] J.-W. Yoon, W. S. Chang, and S. H. Cho, „Laser direct patterning of AgNW CNT hybrid thin films,“ *Optics and Lasers in Engineering*, vol. 73, pp. 40–45, 2015.

[33] F. Xu and Y. Zhu, „Highly conductive and stretchable silver nanowire conductors,“ *Advanced Materials*, vol. 24, no. 37, pp. 5117–5122, 2012.

[34] V. Martinez, F. Stauffer, M. O. Adagunodo, *et al.*, „Stretchable silver nanowire elastomer composite microelectrodes with tailored electrical properties,“ *ACS Applied Materials & Interfaces*, vol. 7, no. 24, pp. 13 467–13 475, 2015.

[35] Y. J. Fan, X. Li, S. Y. Kuang, *et al.*, „Highly robust, transparent, and breathable epidermal electrode,“ *ACS Nano*, vol. 12, no. 9, pp. 9326–9332, 2018.

[36] Z. Jiang, M. O. G. Nayeem, K. Fukuda, *et al.*, „Highly stretchable metallic nanowire networks reinforced by the underlying randomly distributed elastic polymer nanofibers via interfacial adhesion improvement,“ *Advanced Materials*, vol. 31, no. 37, p. 1 903 446, 2019.

[37] N. E. Persson, M. A. McBride, M. A. Grover, and E. Reichmanis, „Automated analysis of orientational order in images of fibrillar materials,“ *Chemistry of Materials*, vol. 29, no. 1, pp. 3–14, 2016.

[38] J. Liu and R. Yang, „Tuning the thermal conductivity of polymers with mechanical strains,“ *Physical Review B*, vol. 81, no. 17, 2010.

[39] A. Philipp, N. W. Pech-May, B. A. F. Kopera, *et al.*, „Direct measurement of the in-plane thermal diffusivity of semitransparent thin films by lock-in thermography: An extension of the slopes method,“ *Analytical Chemistry*, vol. 91, no. 13, pp. 8476–8483, 2019.

[40] H. M. Akram, M. Maqsood, and H. Rashid, „Development and performance analysis of a standard orifice flow calibration system,“ *Review of Scientific Instruments*, vol. 80, no. 7, p. 075 103, 2009.

15.7 Supporting Information

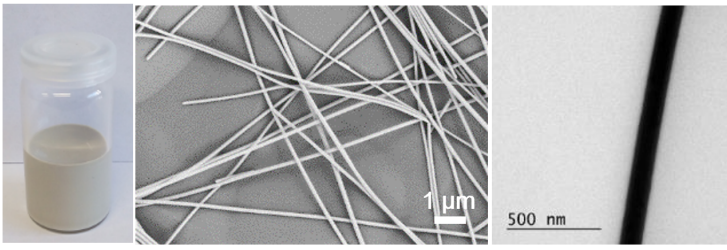


Fig. 15.6.: Digital image of the AgNW dispersion (17 $^{mg}/_{mL}$) (left), SEM (middle) and TEM (right) images of AgNWs.

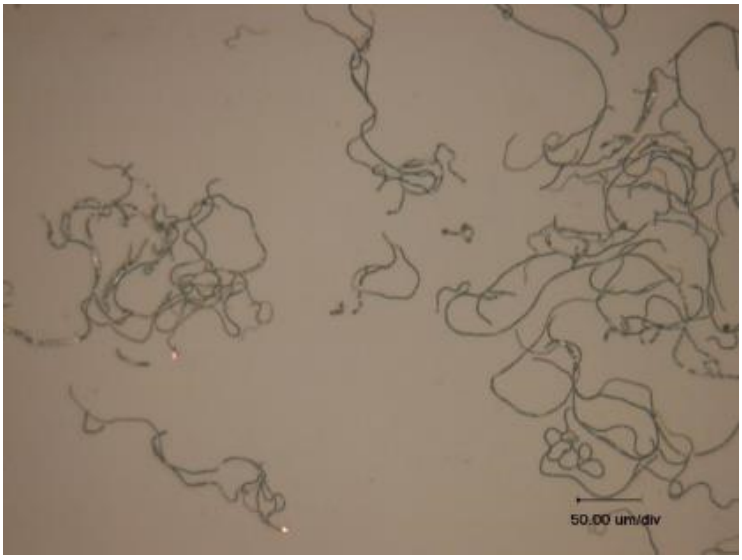


Fig. 15.7.: Optical microscopy image of short PCL fibers.

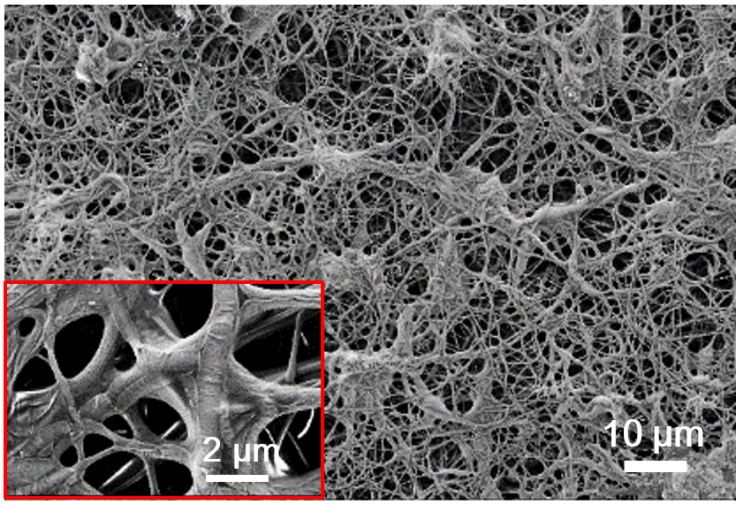


Fig. 15.8.: SEM image of TPU-PCL-TPU

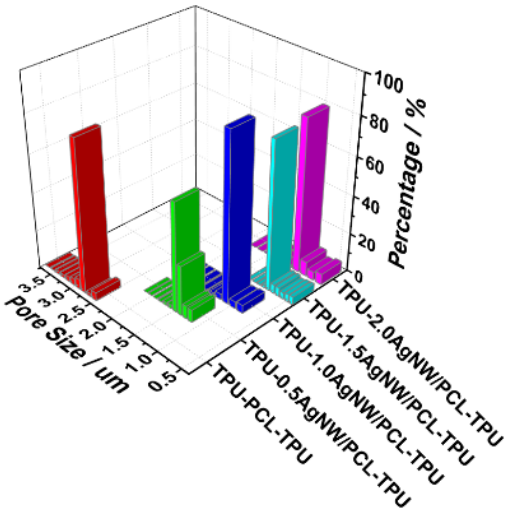


Fig. 15.9.: Pore size distribution of the sandwich membranes with different amounts of AgNWs.

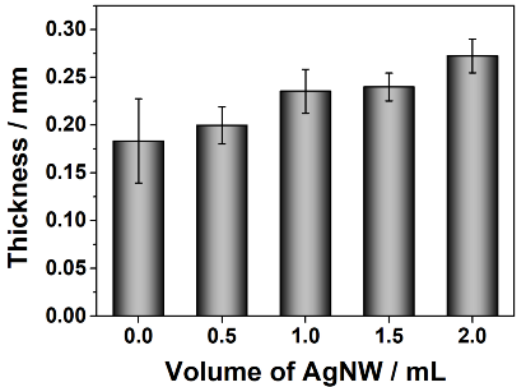


Fig. 15.10.: Thickness of sandwich membranes with different amounts of AgNWs.

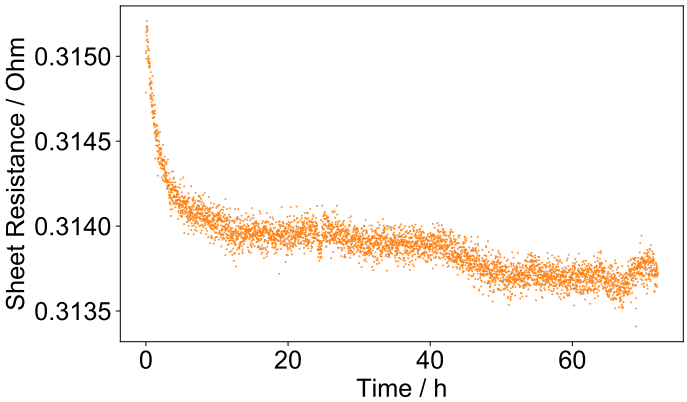


Fig. 15.11.: Sheet resistance stability of AgNW in air at 100 °C. The samples composed by shorter AgNW (here used ≈ 10 µm in length) showed higher resistance compared to those samples with longer AgNW (17 µm, 0.1 Ω), which was demonstrated in our previous work.[1]

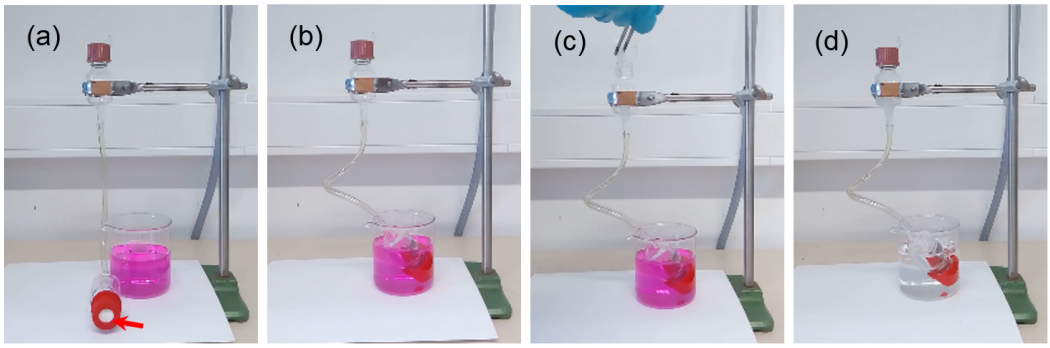


Fig. 15.12.: Digital images of the color change of an aqueous basic solution with phenolphthalein indicator; sandwich membrane (indicated by a red arrow) and aqueous basic solution with phenolphthalein indicator (a); the color of the aqueous basic solution with phenolphthalein indicator before (b), in (c) and after (d) adding dry ice.

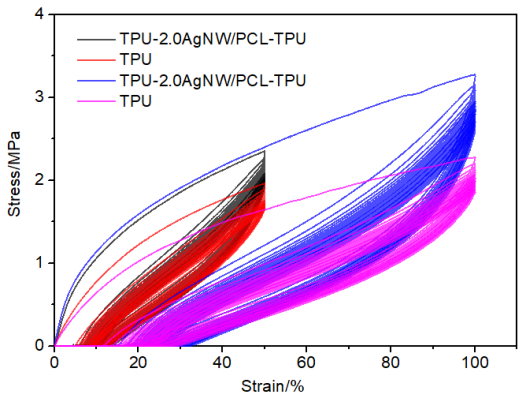


Fig. 15.13.: 100-cycle stress-strain curves of TPU-2.0AgNW/PCL-TPU. TPU-2.0AgNW/PCL-TPU (black): 50 % strain; TPU (red): 50 % strain; TPU-2.0AgNW/PCL-TPU (blue): 100 % strain; TPU (pink): 100 % strain.

15.7.1 Mechanism of hotspot formation

After stretching, seen in the below diagram (Fig 15.14 left), island parts composed by dense AgNWs and bridge parts composed by sparse AgNWs form respectively (Fig 15.14 right), where presents different resistance of smaller resistance in R1 and far larger resistance in R2. The total resistance of system is determined by the resistance of bridge parts, which has been reported by Someya (Adv. Mater. 2019, 1903446). Hence hotspots appear on the bridge parts, according to the joule heating power, P, is given by

$$P = R \cdot I^2 \tag{15.3}$$

where R is the electric resistance and I the DC current. TPU is an electric insulator. Therefore, all the current flows through the AgNW network. We used a current source and applied a constant current during our measurements. The local heating

power, and to some extent, the temperature, is then proportional to the local resistance. The resistance depends on the AgNW cross sectional area perpendicular to the current. Hotspots are therefore indicative of local constrictions (bottlenecks) in the AgNW network.

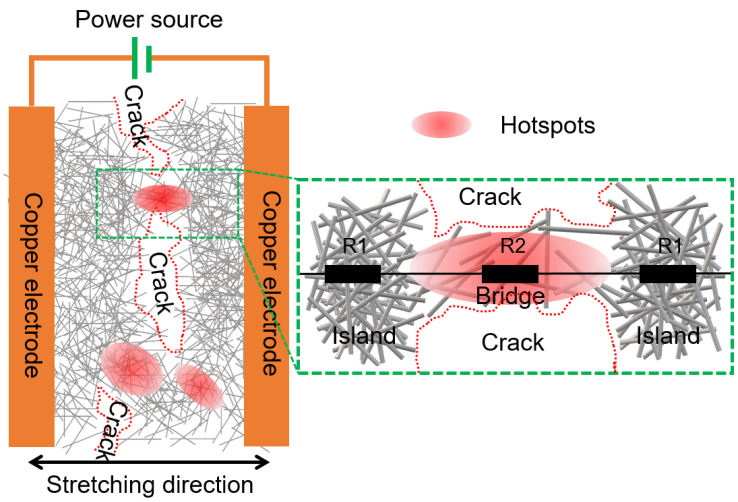


Fig. 15.14.: Schematic diagram of hotspots appearance by stretching. (R1≪R2)

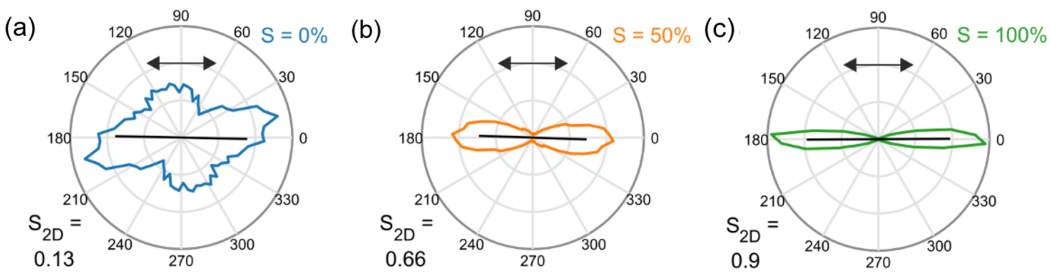


Fig. 15.15.: Polar orientation maps for the TPU surface fibers derived from optical microscopy images in different strain of 0 % (a), 50 % (b) and 100 % (c).

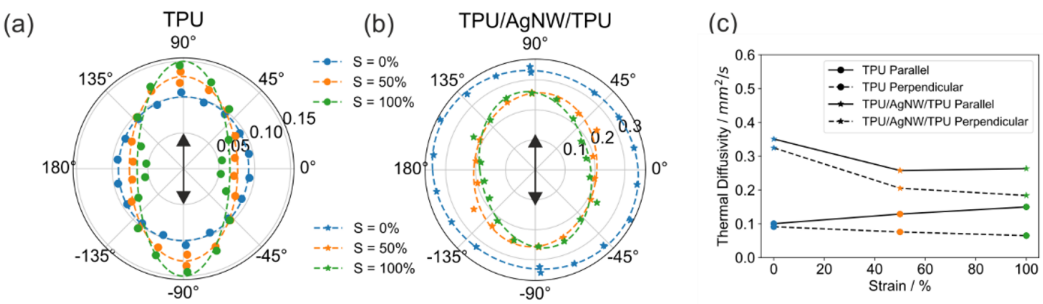


Fig. 15.16.: (a, b) Thermal diffusivity as a function of direction for pure TPU and the TPU/ AgNW /TPU sandwich membrane. Black arrows indicate the stretching direction. Dashed lines are ellipses fitted to the data. (c) Thermal diffusivity parallel and perpendicular to the stretching direction as a function of strain.

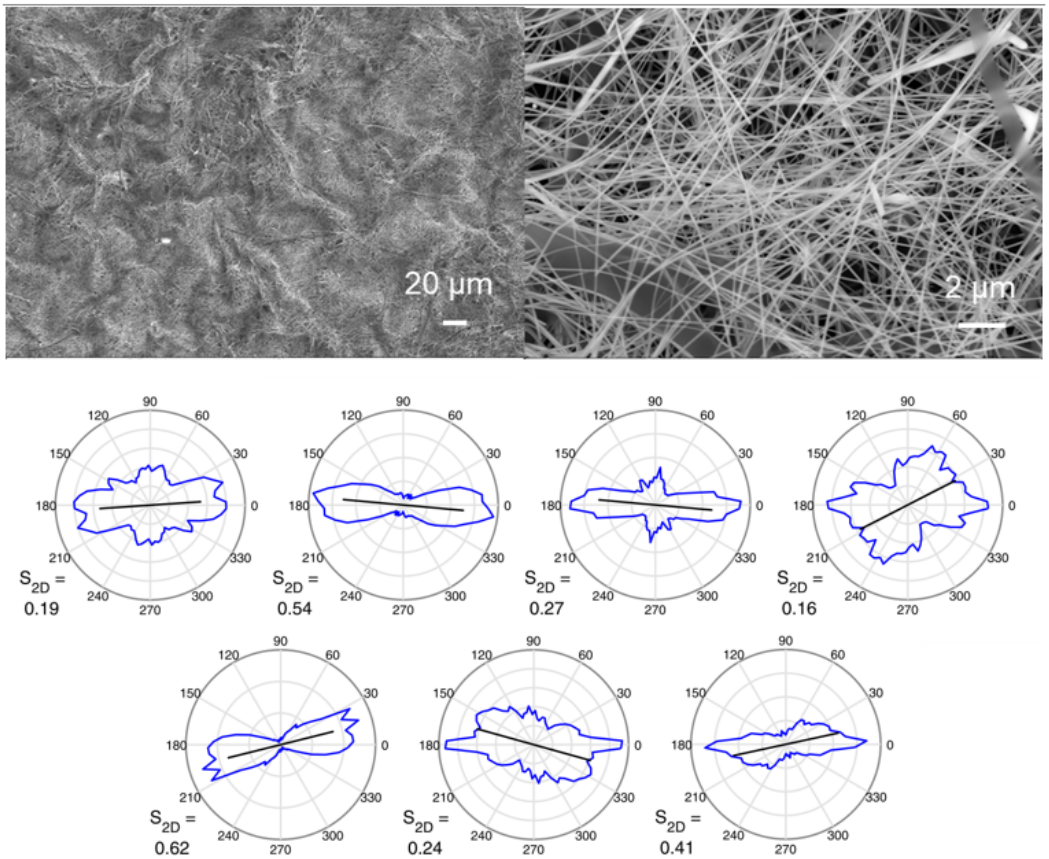


Fig. 15.17.: SEM images (top) of AgNWs mesh without stretching (0 % strain) and orientation plots from different areas (bottom). The order parameter varies between 0.16 and 0.62. Each polar plot was calculated from a different SEM image. The average of S_{2D} is < 0.4, presenting merely partial alignment at 0 % strain.

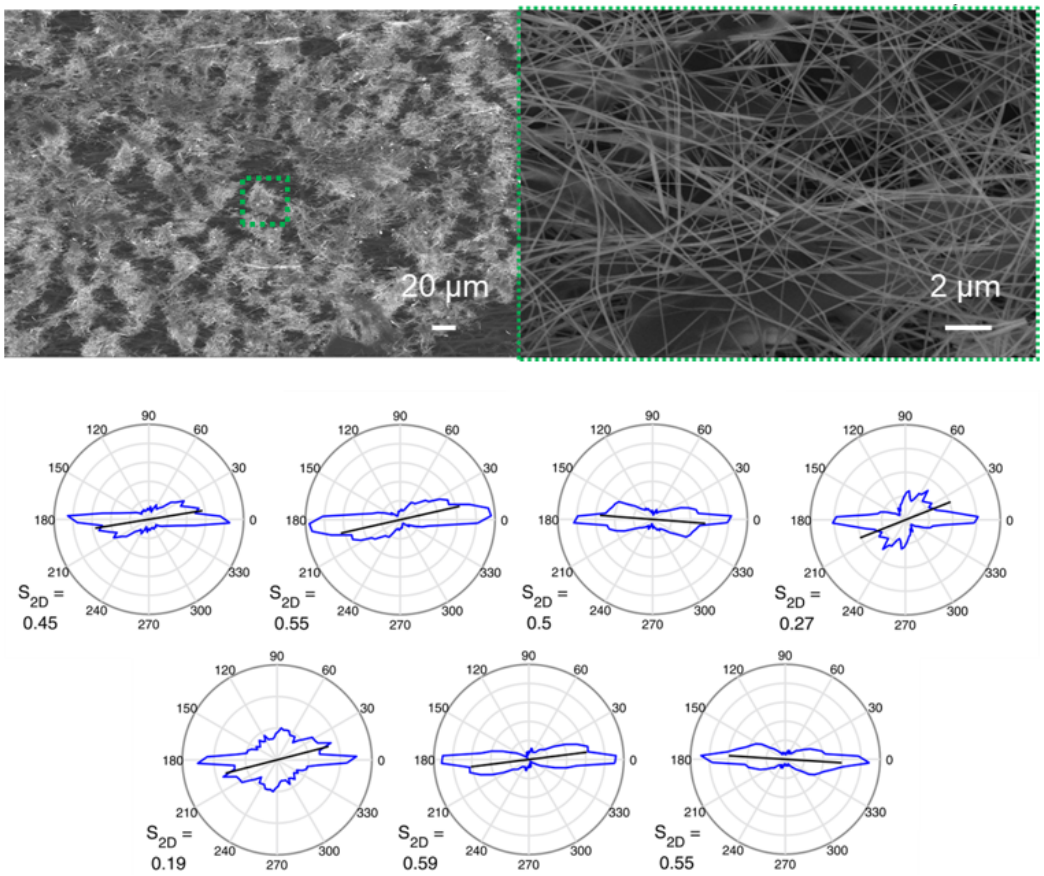


Fig. 15.18.: SEM images (top) of AgNWs mesh under 100 % strain and orientation plots from different areas (bottom). The order parameter varies between 0.19 and 0.59. Each plot was calculated from a different AgNW island. The average of S_{2D} is ≈ 0.4, at 100 % strain, which is close to the order parameter at 0 % strain, demonstrating that stretching has no significant influence on the alignment of AgNWs.

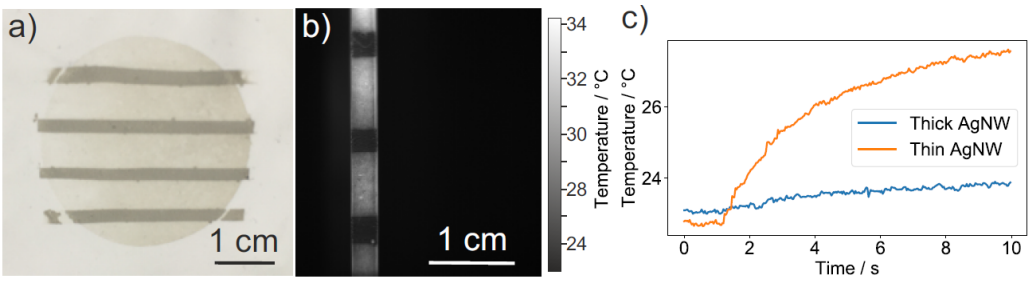


Fig. 15.19.: a) Dedicated adjustment of the electrical resistance by local filtration of various concentrations. b) The lateral change in resistance of a thin strip from the sample in (a) is imaged by thermography. A constant current of 54 mA is run through the sample from top to bottom. c) Surface temperature change vs. time. The areas of low AgNW content (wide, bright stripes) become significantly warmer than the dark areas with high AgNW content.

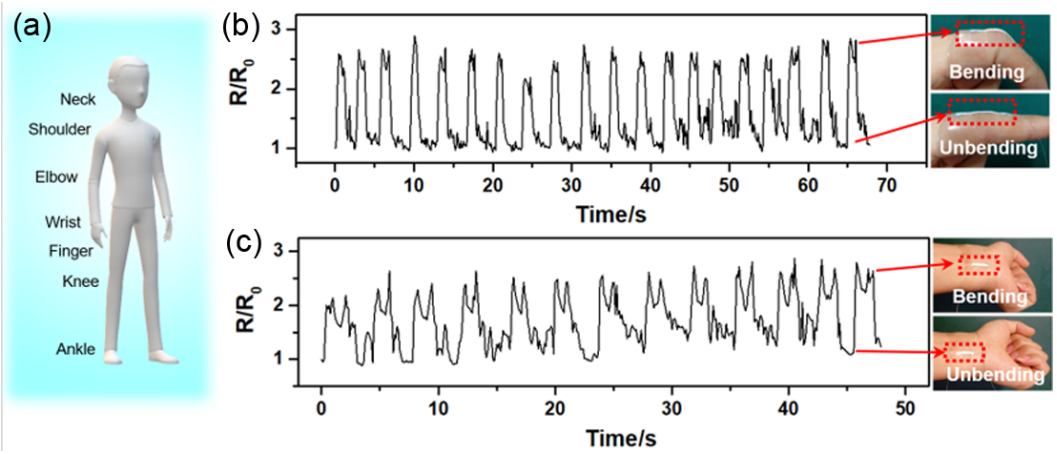


Fig. 15.20.: Sandwich membrane utilizing as a strain sensor for monitoring the movement of potential body parts (a); Sheet resistance change ratio for monitoring the movement of finger (b) and wrist (c).

References

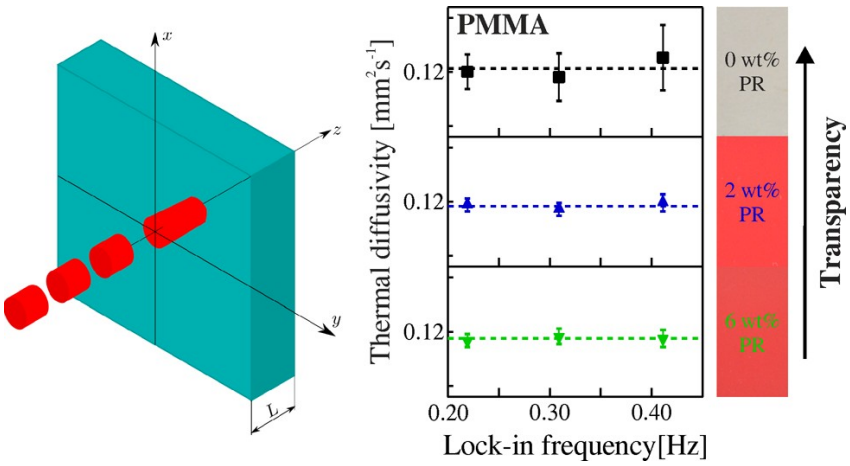
[1] S. Reich, M. Burgard, M. Langner, *et al.*, „Polymer nanofibre composite nonwovens with metal-like electrical conductivity,“ *npj Flexible Electronics*, vol. 2, no. 1, 2018.

16

Direct Measurement of the In-Plane Thermal Diffusivity of Semitransparent Thin Films by Lock-In Thermography: An Extension of the Slopes Method

Alexandra Philipp, Nelson W. Pech-May, Bernd A. F. Kopera, Anna M. Lechner, Sabine Rosenfeldt, and Markus Retsch

Department of Chemistry, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany



Published in *Analytical Chemistry*, **2019**, 91, 8476-8483
Reproduced with permission from ACS Publications

16.1 Abstract

We present an extension of the well-known slopes method for characterization of the in-plane thermal diffusivity of semitransparent polymer films. We introduce a theoretical model which considers heat losses due to convection and radiation mechanisms, as well as semitransparency of the material to the exciting laser heat source (visible range) and multiple reflections at the film surfaces. Most importantly, a potential semitransparency of the material in the IR detection range is also considered. We prove by numerical simulations and by an asymptotic expansion of the surface temperature that the slopes method is also valid for any semitransparent film in the thermally thin regime. Measurements of the in-plane thermal diffusivity performed on semitransparent polymer films covering a wide range of absorption coefficients (to the exciting wavelength and in the IR detection range of our IR camera) validate our theoretical findings.

16.2 Introduction

Lock-in thermography is a well-known technique for accurate determination of the in-plane thermal dffusivity of solids.[1]–[4] In recent decades, it has attracted the attention of scientists due to its non contact and noninvasive attributes. Moreover, lock-in thermography has shown versatility for materials characterization, in nondestructive testing and evaluation as well as in biomedical applications.[5],[6] In particular, for thermal characterization of isotropic solids, the slopes method[2],[7] has been widely used. Therefore, a freestanding sample is periodically heated by a focused optical source. The temperature oscillations as a function of the radial coordinate are detected, typically using an infrared (IR) camera. The thermographic signal (amplitude T and phase Ψ) is either derived from the front-face[2] or the rear-face configuration,[1] i.e., from the non illuminated face. The thermographic signals (ln(amplitude) and phase) far from the heating spot vary linearly with radial distance under the precondition of isotropic opaque thin films in the thermally thin regime (film thickness $\ll$ thermal diffusion length). Moreover, the product between the amplitude slope and the phase slope is independent of convective and radiative heat losses. Consequently, it can be used for the determination of the thermal diffusivity of a film if heat conduction to the gas is negligible.[1],[8] Thus, the slopes method is well-suited for measurement of the in-plane thermal diffusivity of high thermal conductors,[4],[9]–[11] whereas it overestimates, the in-plane thermal diffusivity of low thermal conductors.[1] This overestimate is produced by the large conductive heat losses from the low conductor surface to the surrounding gas. However, the overestimation can be avoided by measuring the sample under vacuum conditions. Even though the characterization of opaque samples using lock-in thermography

has been widely explored, there is almost no attempt in the literature considering the study of semitransparent samples without a coating.[2] Instead, for measuring the in-plane thermal diffusivity of semitransparent films, it is recommended to coat the sample with a thin opaque layer to be able to use the opaque lock-in models. However, there are some cases in which coating the semitransparent material is not an option. For these applications, such as in situ monitoring of the thermal properties evolution of a polymer film or fiber under stretching, a complete model is required which takes into account the effects of the film semitransparency (to the excitation wavelength and in the infrared range of the IR detector or camera) on the thermographic signal.

In this work, we extend the use of lock-in thermography to the measurement of the in-plane thermal diffusivity of semitransparent thin films of isotropic low thermally conductive materials (such as polymers) without coating. A 2D heat conduction model is considered, which includes heat losses to the surrounding atmosphere, semitransparency of the sample to the exciting wavelength, multiple reflections at the sample surfaces, as well as the effect of its semitransparency through the IR window of the IR camera used. In addition to our theoretical work, we performed thermal diffusivity measurements with a home-built lock-in thermography setup. We used three different polymers covering a wide range of absorption properties: poly(methyl methacrylate) (PMMA), low-density polyethylene (LDPE), and polyether ether ketone (PEEK). We chose PMMA as the material because of its low absorption of visible light and high absorption of IR radiation. Thin LDPE films show semitransparency in both the visible and relevant IR wavelength range. We tuned the optical properties of these two materials with the addition of a red dye. The purchased PEEK films are also low absorbers of the visible light. Since their absorption of IR radiation depends on the sample thickness, we measured PEEK films with four different thicknesses.

16.3 Modeling and Experimental Section

Modeling:Heat Conduction through a Semitransparent Film

Consider a semitransparent film of thickness L, heated by a focused Gaussian laser of wavelength λ and power P_0 modulated at frequency f. The focused beam has the radius a (measured at $1/e^2$). In addition, this film is thermally isotropic and is in a vacuum. Under this circumstance, heat conduction to the surrounding air can be neglected in the model. Figure 16.1 shows the described situation.

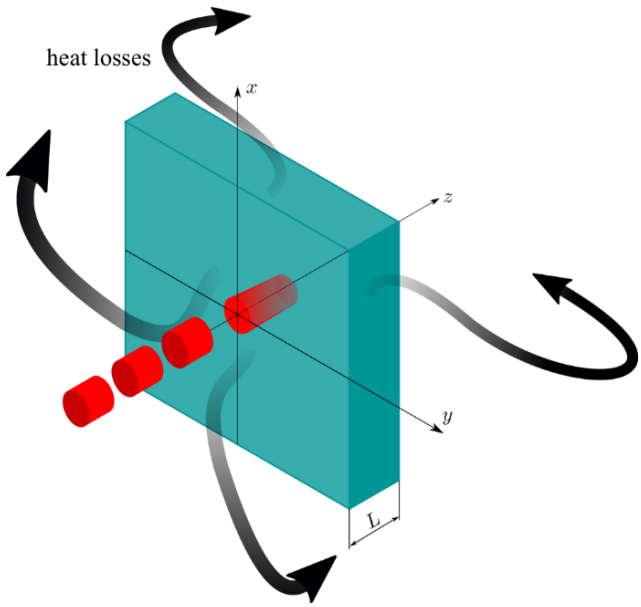


Fig. 16.1.: Semitransparent film heated by a focused, modulated Gaussian laser beam. The black arrows indicate heat losses to the environment.

The surface temperature at $z = 0$ (front-face temperature) can be expressed as (a detailed derivation is provided in the Supporting Information)

$$S(r,0,\omega) = F\frac{P_0}{2\pi K}\frac{(1-R)\alpha\gamma}{1-R^2\exp(-2\alpha L)}\int_0^\infty \delta J_0(\delta r)\frac{\exp\left(-\frac{(\delta a)^2}{8}\right)}{(\beta^2-\alpha^2)E_0}\cdot \tag{16.1}$$
$$\Bigg[-\frac{A_0}{\beta-\gamma}\left(1-\exp\left((\beta-\gamma)L\right)\right)+\frac{B_0}{\beta+\gamma}\left(1-\exp\left(-(\beta+\gamma)L\right)\right)+$$
$$\frac{E_0}{\alpha+\gamma}\left(1-\exp\left(-\left(\alpha+\gamma\right)L\right)\right)-R\exp(-2\alpha L)\cdot\Bigg[-\frac{A_1}{\beta-\gamma}\left(1-\exp((\beta-\gamma)L)\right)+$$
$$\frac{B_1}{\beta+\gamma}\left(1-\exp(-(\beta+\gamma)L)\right)-\frac{E_0}{\alpha-\gamma}\left(1-\exp((\alpha-\gamma)L)\right)\Bigg]\Bigg]\,\mathrm{d}\delta$$

where $\omega = 2\pi f$, r is the radial coordinate, and K is the thermal conductivity of the film. F includes the effect of the IR emissivity, sensor area, IR detection range, and the derivative of the Planck distribution at room temperature. R is the reflectance, and α is the absorption coefficient of the film, both taken at the laser wavelength. γ is an effective IR absorption coefficient, which averages the IR absorption over the detection range of our IR camera. δ is the Hankel space variable; $J_0(\cdot)$ is the Bessel function of zero order, and $\beta^2 = \delta^2 + \frac{i\omega}{D}$. D is the thermal diffusivity of the film. All coefficients A_0, A_1, B_0, B_1, and E_0 are computed explicitly in the Supporting Information, in terms of the thermal diffusivity, modulation frequency, heat convective-radiative losses, and optical absorption.

The surface temperature at $z = L$ (rear-face temperature) reads

$$S(r, L, \omega) = F \frac{P_0}{2\pi K} \frac{(1-R)\alpha\gamma \exp(-\gamma L)}{1 - R^2 \exp(-2\alpha L)} \int_0^\infty \delta J_0(\delta r) \frac{\exp\left(-\frac{(\delta a)^2}{8}\right)}{(\beta^2 - \alpha^2) E_0} \cdot \tag{16.2}$$
$$\left[-\frac{A_0}{\beta+\gamma}\left(1 - \exp\left((\beta+\gamma)L\right)\right) + \frac{B_0}{\beta-\gamma}\left(1 - \exp\left(-(\beta-\gamma)L\right)\right) - \right.$$
$$\frac{E_0}{\alpha-\gamma}\left(1 - \exp\left((\gamma-\alpha)L\right)\right) - R\exp(-2\alpha L) \cdot \left[-\frac{A_1}{\beta+\gamma}\left(1 - \exp((\beta+\gamma)L)\right) + \right.$$
$$\left.\left. \frac{B_1}{\beta-\gamma}\left(1 - \exp(-(\beta-\gamma)L)\right) - \frac{E_0}{\alpha+\gamma}\left(1 - \exp((\alpha+\gamma)L)\right) \right]\right] \mathrm{d}\delta$$

The integrals in eqs. (16.1) and (16.2) cannot be solved in a closed form, not even in the thermally thin regime ($\mu \gg L$), i.e., when the thermal diffusion length $\mu = \sqrt{D/(\pi f)}$ is much larger than the actual thickness of the film (L). Thus, it is hard to prove the validity of the slopes method for semitransparent films. Up to now, it has been confirmed for the optically opaque case. However, it is reliable to do numerical simulations to show the validity of the slopes method for semitransparent films in the thermally thin regime. In this work, we perform simulations for the rear-face configuration, which corresponds to our experimental setup. Moreover, in the thermally thin regime, we have shown the numerical equivalence between the front-face and rear-face temperatures (see Figure 16.6 of the Supporting Information).

Mendioroz *et al.*[1] showed that the radial temperature profile for an optically opaque film in the thermally thin regime far from the punctual heating spot ($a = 0$) can be written as

$$T(r \to \infty, 0, \omega) \approx \sqrt{\frac{\pi}{2}} \frac{\exp(-\sigma r)}{\sqrt{\sigma r}} \tag{16.3}$$

where $\sigma^2 = \frac{2h}{KL} + \frac{i\omega}{D}$ and combined coefficient due to convection and radiation losses. From eq. (16.3), it can be shown that the slopes method (product of the amplitude and phase slopes) gives the thermal diffusivity of the film, overcoming the effect of the heat losses. In general, for a given complex number $\sigma^2 = u + \frac{i\omega}{D}$, such that $u \in \mathbb{R}$, ω and D are positive real numbers. It is straightforward to show that the product of its real part $\Re\{\sigma\}$ and its imaginary part $\Im\{\sigma\}$ is independent of u, i.e., $\Re\{\sigma\} \cdot \Im\{\sigma\} = \frac{\omega}{2D}$.

On the other hand, the asymptotic behavior of the Hankel integrals involved in eqs. (16.1) and (16.2) can be explored.[12],[13]

Consider a Hankel integral of zero order

$$I_0(b) = \int_0^\infty \phi(x) J_0(bx)\, \mathrm{d}x \quad b \to \infty \tag{16.4}$$

This is an asymptotic expansion that when $b \to \infty$ can be expressed as[12]

$$I_0(b \to \infty) = \frac{1}{b} \sum_{s=0}^{\infty} \frac{(-1)^{1+s}\phi^{(s)}(0)}{s!} \frac{\Gamma\left[\left(\frac{1}{2}\right)(1+s)\right]}{\Gamma\left[\left(\frac{1}{2}\right)(1-s)\right]} \left(\frac{2}{b}\right)^s + \Delta_0(b) \tag{16.5}$$

where the first term (infinite summation) is the Poincaré asymptotic expansion (PAE) of the integral in eq. (16.4) and is a series expansion containing integer powers of $1/b$. $\phi^s(0)$ represents the sth derivative of $\phi(x)$ evaluated at $x = 0$, and $\Gamma(\cdot)$ is the gamma function. The second term takes into account the exponentially small decaying terms of the integral. For odd functions $\phi(-x) = -\phi(x)$, the PAE equals zero, and only the exponentially small terms are relevant for the asymptotic expansion. The $\Delta_0(b)$ term can be obtained using the McClure-Wong distributional method.[13] For meromorphic functions, as is our case, it reads

$$\Delta_0(b) = i\pi \sum_{j=0}^{m} \mathrm{Res}\left\{H_0^{(1)}(bz)\phi(z^2)z;\ z = a_j\right\} \tag{16.6}$$

where $H_0^{(1)}(\cdot)$ is the Hankel function of first class and zero order. $\mathrm{Res}\{f(z); z = a_j\}$ stands for the residues of $f(z)$ evaluated at $z = a_j$, and a_j as the jth pole of $\phi(z^2)$ located at the upper half of the complex plane.

Consider a semitransparent film, in the thermally thin regime, illuminated by a punctual laser source ($a = 0$). The asymptotic expansion of the front-face thermographic signal $S(r \to \infty, 0, \omega)$ can be obtained by computation of the residues appearing the integral in eq. (16.1). The terms which contain poles in eq. (16.1) are linear combinations of

$$S(r \to \infty, 0, \omega) \sim \int_0^{\infty} \frac{\delta J_0(\delta r)}{\left(\delta^2 - \alpha^2 + i\frac{\omega}{D}\right)\left(\delta^2 - \gamma^2 + i\frac{\omega}{D}\right)\left(\delta^2 + \frac{2h}{KL} + i\frac{\omega}{D}\right)} \mathrm{d}\delta \tag{16.7}$$

Following eq. (16.6) we obtain

$$S(r \to \infty, 0, \omega) \sim \frac{\exp(-\sigma_\alpha r)}{\sqrt{\sigma_\alpha r}} + \frac{\exp(-\sigma_\gamma r)}{\sqrt{\sigma_\gamma r}} + \frac{\exp(-\sigma_h r)}{\sqrt{\sigma_h r}} \tag{16.8}$$

where $\sigma_\alpha^2 = -\alpha^2 + \frac{i\omega}{D}$, $\sigma_\gamma^2 = -\gamma^2 + \frac{i\omega}{D}$ and $\sigma_h^2 = \frac{2h}{KL} + \frac{i\omega}{D}$. Note that if the three real parts are of the same order of magnitude, then the expression in eq. (16.8) reduces to a single exponential decaying term

$$S(r \to \infty, 0, \omega) \approx \frac{\exp(-\sigma r)}{\sqrt{\sigma r}} \tag{16.9}$$

where $\sigma^2 = u + \frac{i\omega}{D}$ and u is a combination of α, γ, and $2h/(KL)$. This means that the slopes method holds for this particular case. On the other hand, if any of the real parts are larger than the others, the corresponding term in eq. (16.8) can be dropped,

because its amplitude is exponentially small with respect to the others. Accordingly, eq. (16.8) always can be reduced to eq. (16.9) for a suitable value of u. As mentioned before, the value of u does not influence the product of the real and imaginary parts of σ. Thus, we have shown that the slopes method $m_{\ln(|T\sqrt{r}|)} \cdot m_{\Psi} = {}^{\pi f}/_{D}$ is also applicable for semitransparent films in the thermally thin regime, using a punctual excitation source.

Experimental Section.

Materials. Poly(methyl methacrylate) Plexiglas 7N (PMMA; Evonik), low-density polyethylene Purell PE 1840H (LDPE; LyondellBasell), tetrahydrofuran (THF; > 99.9%; Sigma-Aldrich), and phenol red (PR; Alfa Aesar) were used as received. Amorphous polyether ether ketone (PEEK) films with a thickness of 25, 50, 75, and 250 µm were purchased from www.goodfellow.com; potassium bromide (KBr) round cell windows were ordered from Sigma-Aldrich.

Preparation of Thin, Freestanding Films for Lock-In Thermography. Freestanding PMMA films were prepared from solution. Therefore, 30 wt% PMMA was dissolved in THF under magnetic stirring. To obtain red PMMA films 2 and 6 wt% PR (with regard to the amount of PMMA), respectively, were added to PMMA before dissolution in THF. Then, the PMMA solution was cast on a glass substrate using the doctorblade method. After drying for 48 h, the film was removed from the glass substrate and cut into pieces of appropriate sizes. The thickness of the PMMA films was around 200 µm. The fabrication of the freestanding LDPE films is composed of the following steps: compounding, injection molding, and hot pressing. First, LDPE pellets and PR powder were mixed under N_2 flow in a twin-screw compounder with a stirring speed of 40 rpm and at a temperature of 200 °C. Second, the compounded material was directly filled into the injection unit. Discs with a diameter of $\sim$ 27 mm and a thickness of $\sim$ 1 mm were fabricated using an injection force of 6 kN and a tool temperature of 20 °C. Finally, thin, freestanding LDPE films were obtained by hot pressing of the discs at a temperature of 200 °C and subsequent cooling to room temperature. In this way, LDPE films with 0, 2, and 6 wt% PR, respectively, were prepared. The thickness of the LDPE films was around 200 µm. Photographs of the PMMA and LDPE thin films with various contents of PR are depicted in Figures 16.8 and 16.9. PEEK films were cleaned and cut into pieces.

Characterization Methods. In-plane thermal diffusivity measurements on thin, freestanding films (200 µm) were conducted using a home-built lock-in thermography (LIT) setup (Figure 16.9). The sample was heated by a laser beam (Genesis MX 488-1000 SLM OPS, Coherent, $\lambda = 488.1$ nm) focused onto the sample surface by a lens of 150 mm focal length. The amplitude of the laser was modulated in sine

waveform using a Rigol waveform generator DG1022A. The emitted infrared (IR) radiation of the sample surface is detected by an Infratec VarioCAM HD research IR camera (7.5-14 µm). The camera is equipped with a close-up lens. In this configuration, the minimum spatial resolution is 29 µm at a working distance of 33 mm. To avoid heat losses due to conduction and convection into the environment, all samples were measured under vacuum conditions ($\sim 3 \times 10^{-3}$ mbar). However, heat losses due to radiation still remain. LIT measurements were performed using Infratecs IRBISactiveonline software. Transparent samples (PMMA and LDPE without phenol red as well as all PEEK films) were coated with a 20 nm carbon layer for enhanced laser absorption. The coating of the sample was facing the infrared camera. The influence of the carbon coating on the absorption of light in the IR and UV-vis range is shown in Figure 16.11.

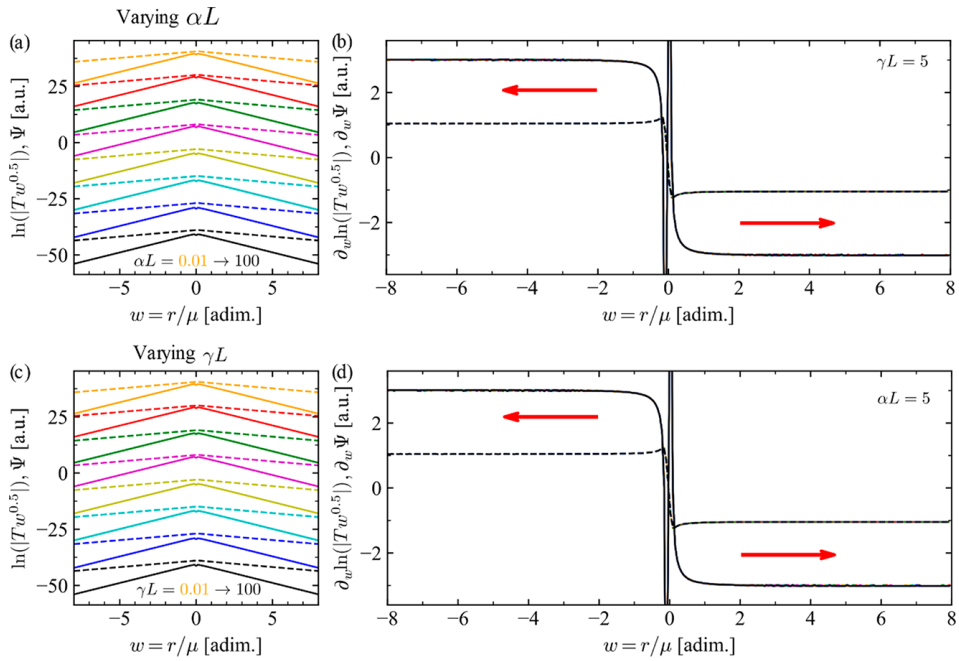


Fig. 16.2.: (a) Plots of $\ln(|T\sqrt{w}|)$ (continuous lines) and Ψ (dashed lines) as a function of the normalized radial profile $w = {}^{r}/_{\mu}$. A wide range of absorption coefficients is explored at a fixed IR absorption: $\gamma L = 5$. (b) Derivatives of amplitudes (continuous lines) and phases (dashed lines) with respect to w, for different absorption coefficients, are explored. Note that results overlap. (c) Plots of $\ln(|T\sqrt{w}|)$ (continuous lines) and Ψ (dashed lines) as a function of the normalized radial profile$w = {}^{r}/_{\mu}$. A wide range of IR absorption coefficients is explored at a fixed optical absorption: $\alpha L = 5$. (d) Derivatives of amplitudes (continuous lines) and phases (dashed lines) with respect to w, for the different IR absorption coefficients, are explored. Note that the results overlap.

In the case of the fabricated polymeric films (PMMA and LDPE), we measured three different films per sample; in the case of the purchased PEEK films, we measured only one film per sample.

UV-vis measurements were conducted on an Agilent Cary 5000 spectrometer in the transmission mode. Absorption spectra were recorded from 380 to 800 nm. The data interval was fixed to 1 nm, with averaging for 100 ms. Each measured sample was normalized with a suitable reference (control): For the freestanding polymers, the direct lamp spectrum in air was used. For the measurement of the carbon coating, an uncoated KBr disc was used. The phenol red powder was measured by applying a thin layer on transparent adhesive tape with the same tape as reference. The same polymeric samples as characterized by lock-in thermography were investigated.

A Bruker Vertex 70 FT-IR spectrometer was used for absorption spectra acquisition. The polymeric samples, as well as a carbon-coated KBr window, were measured in transmission mode. To get the IR absorbance of the pure carbon layer, an uncoated KBr window with the same thickness as the uncoated one was used as a reference. Furthermore, an ATR-IR spectrum of phenol red powder was recorded. All samples were measured in the wavenumber range of the spectral range of the infrared camera (data interval, 4 cm^{-1}; averaging, 32 measurements).

16.4 Results and Discussion

Numerical Simulations

Here we present numerical simulations of the radial temperature profiles based on eq. (16.2). A thin polymeric film (K = 0.15 $Wm^{-1}K^{-1}$ with thickness L = 25 µm is considered. The laser power is set to $P_0 = 50$ mW at a modulation frequency $f = 0.10$ Hz. This gives a thermal diffusion length µ = 564 µm; i.e., the film is in the thermally thin regime ($\mu \gg L$). A typical value is used for the combined heat transfer coefficient h = 15 $Wm^{-2}K^{-1}$.[8],[14] Figure 16.2a shows simulations of a surface temperature radial profile for a semitransparent film, considering a fixed IR absorption coefficient such that $\gamma L = 5$. The absorption coefficient to the incident laser wavelength is varied over a wide range: $0.01 \leq \alpha L \leq 100$. The laser beam is focused on the film surface with a radius $a = 0.1\mu$. The temperature profiles for the "amplitude" $\ln(|T\sqrt{w}|)$ and phase ψ are presented with a vertical shift and in ascending order of αL from top to bottom.

The derivatives of the "amplitude" $\partial_w \ln(|T\sqrt{w}|)$ and phase $\partial_w \psi$ with respect to the normalized radial profile are antisymmetric around the beam spot position (r = 0), as depicted in Figure 16.2b. Both derivatives reach a constant value above $r = 2\mu$ (and below $r = -2\mu$), as indicated by the red arrows. Thus, $|r| \geq 2\mu$ corresponds to the linear part of the temperature profiles shown in Figure 16.2b. Two red arrows mark these intervals. Within the region $0 \leq r \leq 2\mu$, the amplitude derivative shows

a concave upward behavior and is discontinuous at $r = 0$. This discontinuity is due to the logarithmic function. In contrast, the phase derivative is concave downward away from the beam spot; it shows an abrupt change to concave upward near $r = 0$ and is continuous at the beam spot position. Moreover, the values of the derivatives for the different absorption coefficients to the exciting wavelength are equal, as they are all superimposed in Figure 16.2b. This means that the difference between the slopes of the amplitude $\ln(|T\sqrt{w}|)$ and phase Ψ profiles (for $|r| \geq 2\mu$) is due to the heat losses independent of αL.

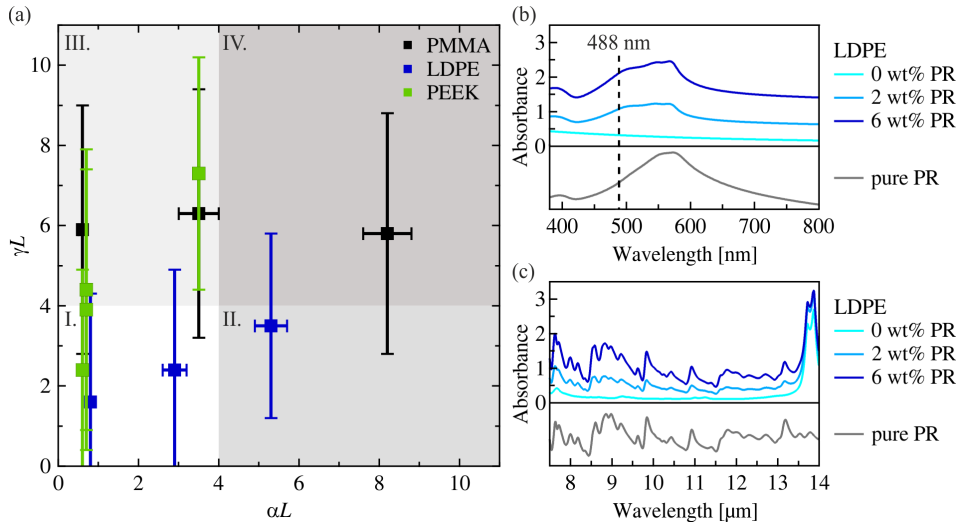


Fig. 16.3.: (a) Plot of A_{IR} versus A_{Vis} of all polymeric thin films. The error bars of A_{IR} arise from measuring three films per sample (except for PEEK) and from averaging over the spectral wavelength range of the infrared camera (7.5-14 µm); the error bars of A_{Vis} arise from measuring three films per sample (except for PEEK). Area I marks the (semi)transparent A_{Vis} and A_{IR} range and area II marks the opaque A_{Vis} range and (semi)transparent A_{IR} range. Area III marks the (semi)transparent A_{Vis} range and opaque A_{IR} range, and area IV marks the opaque A_{Vis} and A_{IR} range. Exemplary (b) IR and (c) UV-vis absorbance spectra of LDPE with 0, 2, and 6 wt% phenol red (PR) as well as the spectra of pure PR powder are shown. The dashed line at 488 nm marks the wavelength of the incident laser used in lock-in measurements.

Simulations presented in Figure 16.2c are similar to those in Figure 16.2a, but in this case, the optical absorption coefficient to the exciting wavelength is fixed ($\alpha L = 5$), and a wide range of IR absorption coefficients are explored: $0.01 \leq \gamma L \leq 100$. The same radius $a = 0.1\mu$ is used for the focused beam. The simulations are vertically shifted in ascending order of γL from top to bottom.

The corresponding "amplitude" and phase derivatives are shown in Figure 16.2d. The same characteristics as described in Figure 16.2b are found. In this case, the values of the derivatives for the different IR absorption coefficients are equal, as they are all superimposed. This means that the difference between the slopes of the

Tab. 16.1.: Summary of A_{Vis} and A_{IR} values of PMMA and LDPE Films with Various PR Contents and PEEK Samples with Different Film Thicknesses.[a]

		A^c_{Vis}	A^d_{IR}	exciting laser light	IR radiation
PMMA	0 wt % PR[b]	0.6 ± 0.1	$> 3.5^e$	semitransparent	opaque
	2 wt % PR	3.5 ± 0.5	$> 3.5^e$	semitransparent	opaque
	6 wt % PR	8.2 ± 0.6	$> 3.5^e$	opaque	opaque
LDPE	0 wt % PR[b]	0.8 ± 0.1	1.6 ± 2.7	semitransparent	semitransparent
	2 wt % PR	2.9 ± 0.3	2.4 ± 2.5	semitransparent	semitransparent
	6 wt % PR	5.3 ± 0.4	3.5 ± 2.3	opaque	semitransparent
PEEK	25 μm film[b]	0.6	2.4 ± 2.5	semitransparent	semitransparent
	50 μm film[b]	0.7	$> 3.5^e$	semitransparent	opaque
	75 μm film[b]	0.7	$> 3.5^e$	semitransparent	opaque
	250 μm film[b]	3.5	$> 3.5^e$	semitransparent	opaque

[a]Classification of the optical properties regarding the exciting laser light and IR radiation in semitransparent or opaque. [b]With 20 nm carbon coating. [c]Value at 488 nm. [d]Average value for 7.5-14 µm; error is the standard deviation of the average value. [e]Above the detection limit. In the case of the PMMA and LDPE samples, three films were measured, and an average value of A_{Vis} and A_{IR} was calculated.

amplitude $\ln(|T\sqrt{w}|)$ and phase Ψ profiles is also due to the heat losses, independent of γL. To validate the above results, further simulations for a wide range of both absorption coefficients were performed (see Supporting Information).

We have found that the product between the amplitude derivative $\mathrm{d}_w \ln(|T\sqrt{w}|)$ and phase derivative $\mathrm{d}_w\psi$, for $|r| \geq \mu$,is constant and independent of the optical absorption to the exciting wavelength and to the optical absorption in the IR detection range; i.e., this result is also correct in the optically opaque limit. Consequently, on the basis of the slopes method for optically opaque films in the thermally thin regime,[1] we conclude that, for any semitransparent or opaque film in the thermally thin regime and far from the excitation beam spot, $|r| \geq \mu$:

$$\partial_r \ln(|T\sqrt{r}|) \cdot \partial_r \Psi = m_{\ln(|T\sqrt{r}|)} \cdot m_\Psi = \frac{\pi f}{D} \tag{16.10}$$

This confirms our theoretical expectations as outlined above.

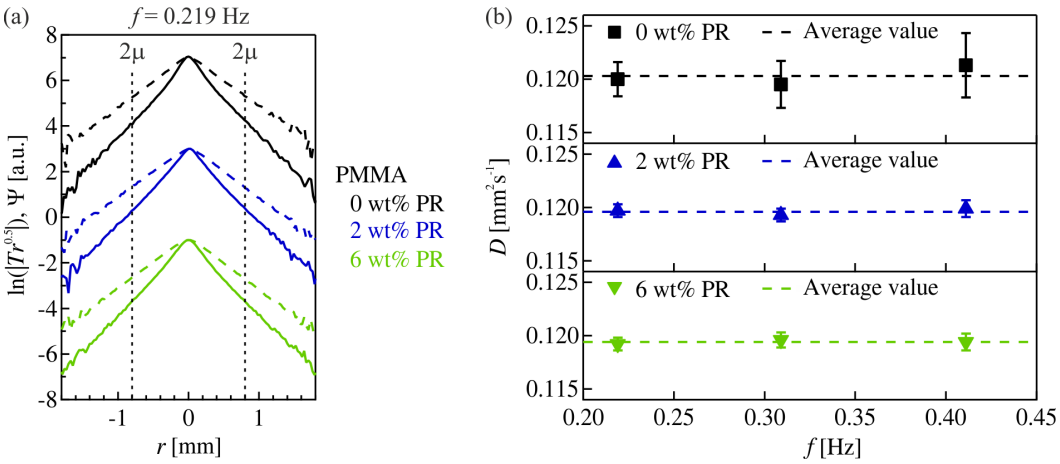


Fig. 16.4.: Exemplary measurement data of a PMMA film with 0, 2, and 6 wt% phenol red (PR), respectively. (a) Plots of $\ln(|T\sqrt{w}|)$ (continuous lines) and Ψ (dashed lines) as a function of the radial profile r. The vertical dashed lines at 2μ ($\sim 800\,\mu$m represent the lower fitting boundary used for evaluation of the phase and amplitude profiles. (b) Plots of the thermal diffusivity D against the lock-in frequency f. Error bars are the standard error arising from the fitting of the radial profiles. The dashed lines represent the respective average value.

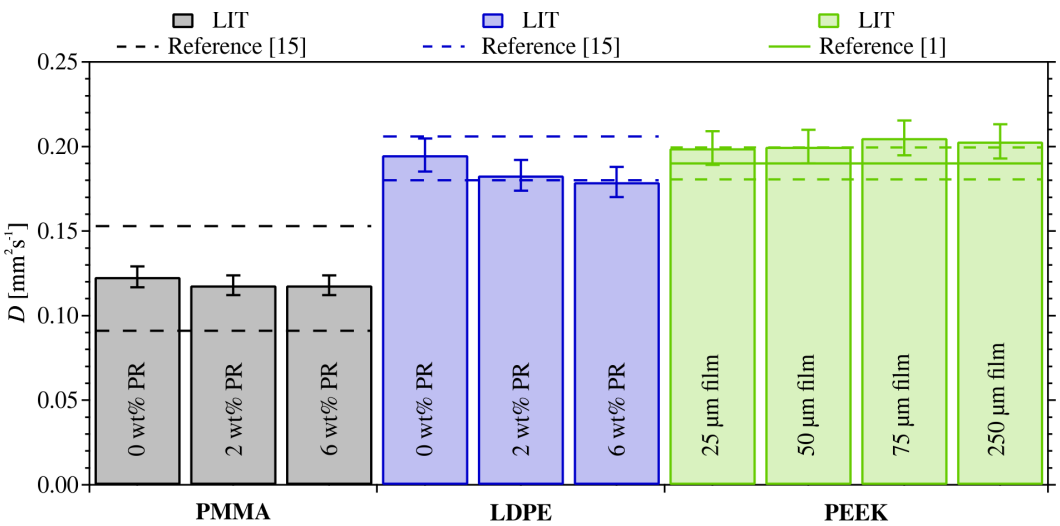


Fig. 16.5.: Thermal diffusivity D of PMMA (black bars) and LDPE (blue bars) with 0, 2, and 6 wt% phenol red (PR), respectively. Error bars are 5 % uncertainty. The dashed lines represent the lower and upper limits of the reference values for unmodified PMMA and LDPE.[15] The thermal diffusivities of PEEK films with a thickness of 25, 50, 75, and 250 µm are plotted as green bars. Error bars are 5 % uncertainty. The corresponding solid line represents the literature value also measured with lock-in thermography. The dashed lines are the 5 % uncertainty range given by Mendioro *et al.*[1]

Lock-In Thermography Measurements.

To validate our theoretical results experimentally, we measured three different polymers covering a wide range of IR absorption coefficients and optical absorption

coefficients to the exciting wavelength: poly(methyl methacrylate) (PMMA), low-density polyethylene (LDPE), and polyether ether ketone (PEEK). To tune the optical properties of PMMA and LDPE, we added 2 and 6 wt% phenol red (PR), respectively (Figure 16.8). Furthermore, PEEK films with different thicknesses, i.e., 25, 50, 75, and 250 µm, were investigated.

In Figure 16.3a, a plot of the IR absorption ($A_{\mathrm{IR}} = {}^{\gamma L}/_{\ln(10)}$) as a function of the optical absorption ($A_{\mathrm{Vis}} = {}^{\alpha L}/_{\ln(10)}$)to the exciting wavelength of all samples is shown. In Table 16.1, the corresponding values are listed. Furthermore, the samples are classified in "semitransparent" or "opaque" regarding their optical properties to the exciting laser light and IR radiation.

Figure 16.3a is divided into four areas dependent on the optical properties of the material:

I (semi)transparent to both IR radiation in the wavelength range of the IR camera (7.5-14 µm) and the exciting wavelength of the laser (488 nm),

II (semi)transparent to the IR radiation and opaque to the exciting wavelength,

III opaque to the IR radiation and (semi)transparent to the exciting wavelength, and

IV opaque to both IR radiation and exciting wavelength.

As can be seen from this plot, PMMA is opaque to IR radiation and semitransparent (III) or opaque (IV) to the exciting laser light depending on the dye concentration. LDPE, on the other hand, is semitransparent to IR radiation and semitransparent (I) or opaque, due to a higher PR content, (II) to the exciting laser light. Here, the addition of PR enhances not only the absorbance of visible light (Figure 16.3b) but also the absorbance of IR light (Figure 16.3c). All PEEK films are semitransparent to the exciting laser light, but either semitransparent (I) or opaque (III) to the IR radiation depending on the sample thickness. IR and UV-vis absorbance spectra of all samples are summarized in Figure 16.12.

Exemplary lock-in thermography (LIT) measurement data of PMMA with various PR contents (0, 2, 6 wt% PR) are depicted in Figure 16.4a. Here, temperature profiles for the "amplitude" $\ln(|T\sqrt{w}|)$ and phase Ψ are shown for a lock-in frequency f of 0.219 Hz. To be independent of the optical properties of the thin films, we extracted the phase and "amplitude" slopes above a radial distance of 2μ (vertical dashed lines in Figure 16.4a). Subsequently, we calculated the thermal diffusivity D for each frequency using the equation $m_{\ln(|T\sqrt{r}|)} \cdot m_{\Psi} = {}^{\pi f}/_{D}$. Fits were done on radial

averaged profiles for each phase and amplitude image. In this way, the whole phase and amplitude distribution are evaluated rather than single lines. Figure 16.4b shows the individual thermal diffusivities with dependence on the used lock-in frequency. The error bars arise from the fitting of the radial profiles.

From this data, an average thermal diffusivity is calculated (dashed line). The data in Figure 16.4b indicates that the PMMA films have similar thermal diffusivities independent of the PR content. Therefore, the slopes method is valid independent of the optical absorption at the exciting wavelength. However, the standard error of the individual thermal diffusivities is larger for the sample without PR due to the lower signal-to-noise ratio of the radial profiles. The thermal diffusivities determined by LIT are shown in Figure 16.5. We measured PMMA and LDPE films with 0, 2, and 6 wt% of PR content. Additionally, four PEEK samples with different thicknesses were measured. For all samples, we obtained rather small standard deviations of the thermal diffusivity. In Figure 16.5, we plot an uncertainty of 5 % as error bars.[1],[15]

From our lock-in measurements, we determined an in-plane thermal diffusivity of $\sim 0.12\,\mathrm{mm^2s^{-1}}$ for PMMA independent of the PR content. Thus, the result does not depend on the optical absorption coefficient to the exciting wavelength that changes from semitransparent to opaque with an increasing amount of red dye (Table 1). Similarly, for LDPE with 0, 2, and 6 wt % PR, we found a thermal diffusivity of $\sim 0.19\,\mathrm{mm^2s^{-1}}$. Here, too, we observe no dependence on the optical absorption coefficient for the exciting laser light (Table 16.1).

We obtained an average thermal diffusivity value of $\sim 0.20\,\mathrm{mm^2s^{-1}}$ for all PEEK samples independent of the film thickness. Consequently, the result is not affected by the IR absorption coefficients that change from semitransparent to opaque with increasing film thickness (Table 16.1). The PEEK films also serve as reference material for our LIT setup, since literature values are available that were also measured with thermographic methods.[1],[16] The literature value of $\sim 0.19\,\mathrm{mm^2s^{-1}}$ from Mendioroz *et al.*[1] is plotted as a solid green line in Figure 16.5; the dashed green lines represent the 5 % uncertainty range of this value. We found a low relative deviation of $\sim$ 5 % between our average value and the literature value. This good agreement proves that correct thermal diffusivity values are obtained even for samples with low optical absorption coefficients for the exciting wavelength.

We note the fundamental difficulty in quantitatively comparing the thermal conductivity of polymer samples reported from different groups. This is an inherent problem owing to the variability of the sample microstructure, which can be strongly altered on the basis of the fabrication and processing conditions. This is particularly true for semicrystalline samples because not only the polymer chain orientation but

also the degree of crystallinity can vary significantly. Furthermore, a quantitative characterization of the exact polymer microstructure is difficult to achieve and often not reported along the measured thermal conductivity values. In addition to the microstructure, the thermal transport characterization technique may also result in systematically differing thermal conductivity values. Consequently, a broad range of typical thermal conductivities are generally accepted for many polymer materials, which lead to the dashed lines for PMMA and LDPE in Figure 16.5.[15] We report the influence of both issues (sample microstructure and measurement method) on the thermal conductivity value for our samples to provide a more reasonable classification of the absolute thermal conductivity values that we offer (see Supporting Information for further information). We, therefore, compare LIT to xenon flash analysis (XFA) for polymer samples stemming from various preparation conditions. We found for the amorphous PMMA samples no signs of polymer crystallinity (Figure 16.13) and a comparable polymer microstructure. This led to a good agreement between LIT and XFA (which measure the cross-plane thermal diffusivity) measurements. For the semicrystalline LDPE samples, we did not observe any influence of the PR dye on the degree of crystallinity (see Supporting Information). Yet, the orientation of the crystalline domains is strongly anisotropic and depends on the processing conditions such as hot-pressing and thermal annealing (Figure 16.15). The XFA measurement results in lower cross-plane thermal diffusivities compared to LIT, which may be caused by the in-plane orientation of the LDPE crystallites. Thermal annealing further alters the LDPE microstructure, as can be seen for the thick XFA samples, which leads to a concomitant increase in thermal diffusivity. Due to the much lower degree of anisotropy upon hot pressing, the thin samples for LIT are less sensitive to the thermal annealing step and, consequently, do not change significantly.

Considering these uncertainties, it becomes clear that we obtained accurate values of thermal diffusivity for uncoated semitransparent samples using the slopes method. These experimental results validate our theoretical expectations discussed in the Numerical Simulations section. Lock-in thermography is, consequently, a powerful and versatile characterization technique to measure thermal transport in thin, freestanding films and fibers, independent of their optical properties.

16.5 Conclusion

In this paper, we developed a two-dimensional heat conduction model which includes, in addition to heat losses to the surrounding atmosphere, also semitransparency of the sample to the exciting wavelength, multiple reflections at the sample surfaces, and the effect of its semitransparency through the infrared (IR) sensitivity of the IR camera used. On the basis of this model, we conducted numerical simulations to investigate

separately the effect of semitransparency on the exciting laser light and on the IR wavelength of the IR camera. We found that the well-known slope method, which does not consider semitransparency, is still valid as long as the phase and amplitude slopes are extracted far from the heating spot center ($\approx 2\mu$). We translate our theoretical findings into the measurement of three different polymers, poly(methyl methacrylate) (PMMA), low-density polyethylene (LDPE), and polyether ether ketone (PEEK). The optical properties in the visible and IR wavelength ranges of these polymers were varied by their film thickness (PEEK) or by the addition of a red dye (PMMA and LDPE). We obtained thermal diffusivity values which are in good agreement with literature values and independent of the optical absorption properties of the samples. We could, therefore, show that, in the thermally thin regime, the slopes method holds (far from the heating spot center) independent of semitransparency (to the exciting wavelength and in the IR range of the camera). Consequently, the in-plane thermal diffusivity can be measured accurately.

Associated Content

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.analchem.9b01583. Details of the analytical model for a semitransparent film, photographs of polymer samples, UV-vis and IR spectra of all samples as well as the influence of carbon coating, XRD and SAXS data, and reference thermal diffusivity measurements using XFA.

Author Information

Corresponding Authors

* E-mail: markus.retsch@uni-bayreuth.de
* E-mail: nelson.pech@uni-bayreuth.de

ORCID

- Nelson W. Pech-May : 0000-0002-5822-482X
- Markus Retsch: 0000-0003-2629-8450

Notes

The authors declare no competing financial interest.

Acknowledgments

The authors thank Rainer Giesa and Sandra Ganzleben for help with injection molding and hot-pressing. Argyrios Georgiadis is acknowledged for XFA measurements of the LDPE samples after thermal annealing. B. A. F. K. acknowledges support from the Elite Network of Bavaria (ENB). The Bavarian Polymer Institute is acknowledged for access to polymer processing and small angle scattering facilities. The Volkswagen foundation funded this work through a Lichtenberg professorship. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714968).

16.6 References

[1] A. Mendioroz, R. Fuente-Dacal, E. Apiñaniz, and A. Salazar, „Thermal diffusivity measurements of thin plates and filaments using lock-in thermography," *Review of Scientific Instruments*, vol. 80, no. 7, p. 074 904, 2009.

[2] B. Zhang and R. E. Imhof, „Theoretical analysis of the surface thermal wave technique for measuring the thermal diffusivity of thin slabs," *Applied Physics A Materials Science & Processing*, vol. 62, no. 4, pp. 323–334, 1996.

[3] A. Salazar, A. Mendioroz, R. Fuente, and R. Celorrio, „Accurate measurements of the thermal diffusivity of thin filaments by lock-in thermography," *Journal of Applied Physics*, vol. 107, no. 4, p. 043 508, 2010.

[4] A. Wolf, P. Pohl, and R. Brendel, „Thermophysical analysis of thin films by lock-in thermography," *Journal of Applied Physics*, vol. 96, no. 11, pp. 6306–6312, 2004.

[5] C. Boué, F. Cassagne, C. Massoud, and D. Fournier, „Thermal imaging of a vein of the forearm: Analysis and thermal modelling," *Infrared Physics & Technology*, vol. 51, no. 1, pp. 13–20, 2007.

[6] J. Rantala, D. Wu, and G. Busse, „Amplitude-modulated lock-in vibrothermography for NDE of polymers and composites," *Research in Nondestructive Evaluation*, vol. 7, no. 4, pp. 215–228, 1996.

[7] L. Fabbri and P. Fenici, „Three-dimensional photothermal radiometry for the determination of the thermal diffusivity of solids," *Review of Scientific Instruments*, vol. 66, no. 6, pp. 3593–3600, 1995.

[8] A. Salazar, A. Mendioroz, and R. Fuente, „The strong influence of heat losses on the accurate measurement of thermal diffusivity using lock-in thermography," *Applied Physics Letters*, vol. 95, no. 12, p. 121 905, 2009.

[9] A. Muscio, P. G. Bison, S. Marinetti, and E. Grinzato, „Thermal diffusivity measurement in slabs using harmonic and one-dimensional propagation of thermal waves,“ *International Journal of Thermal Sciences*, vol. 43, no. 5, pp. 453–463, 2004.

[10] C. Pradère, J. Goyhénèche, J. Batsale, S. Dilhaire, and R. Pailler, „Thermal diffusivity measurements on a single fiber with microscale diameter at very high temperature,“ *International Journal of Thermal Sciences*, vol. 45, no. 5, pp. 443–451, 2006.

[11] M. Oksanen, R. Sscholz, and L. Fabbri, *Journal of Materials Science Letters*, vol. 16, no. 13, pp. 1092–1094, 1997.

[12] E. A. Galapon and K. M. L. Martinez, „Exactification of the poincaré asymptotic expansion of the hankel integral: Spectacularly accurate asymptotic expansions and non-asymptotic scales,“ *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 470, no. 2162, p. 20 130 529, 2014.

[13] C. L. Frenzen and R. Wong, „A note on asymptotic evaluation of some hankel transforms,“ *Mathematics of Computation*, vol. 45, no. 172, pp. 537–537, 1985.

[14] S. O. Kanstad and P.-E. Nordal, „Experimental aspects of photothermal radiometry,“ *Canadian Journal of Physics*, vol. 64, no. 9, pp. 1155–1164, 1986.

[15] Granta Design, *Ces edupack*, 2018.

[16] N. W. Pech-May, A. Mendioroz, and A. Salazar, „Simultaneous measurement of the in-plane and in-depth thermal diffusivity of solids using pulsed infrared thermography with focused illumination,“ *NDT & E International*, vol. 77, pp. 28–34, 2016.

16.7 Supporting Information

Modeling

Consider the geometry shown in Figure 16.1 of the main text. The power distribution $P(z)$ inside the semitransparent thin film can be calculated using the Beer-Lambert law taking into account multiple reflections of the laser beam at the film surfaces,[1]

$$P(z) = \frac{I_0\,(1-R)\,(\exp(-\alpha z) + R\exp(-2\alpha L)\exp(\alpha z))}{1 - R^2\exp(-2\alpha L)}, \tag{16.11}$$

where R is the reflectance and α is the absorption coefficient of the film, both taken at the laser wavelength. A fraction of this power is converted into heat $Q(z) = \chi P(z)$ and is the heating source of the film. Where χ is the efficiency of the light to heat conversion. We consider heat conduction by diffusion in the film. Accordingly, the heat diffusion equation for the presented configuration reads

$$\nabla^2 T - \frac{i\omega}{D}T = -\frac{\chi P_0}{2K}\frac{\exp(-\frac{2r^2}{a^2})}{\pi a^2}\frac{(1-R)\,\alpha\,(\exp(-\alpha z) - R\exp(-2\alpha L)\exp(\alpha z))}{(1 - R^2\exp(-2\alpha L))} \tag{16.12}$$

where $T = T(\vec{r}, \omega)$ is the temperature field inside the sample at the angular frequency $\omega = 2\pi f$, D and K is the thermal diffusivity and thermal conductivity of the film, respectively. Because the film is thermally isotropic, Equation (16.12) has cylindrical symmetry. Thus, we can express the temperature field as a Hankel transform:

$$T(r, z, \omega) = \int_0^\infty \delta J_0(\delta r)\,[A\exp(\beta z) + B\exp(-\beta z) + C\exp(-\alpha z) + E\exp(\alpha z)]\,\mathrm{d}\delta, \tag{16.13}$$

where δ is the Hankel space variable, $J_0(\cdot)$ is the Bessel function of zero order, $\beta^2 = \delta^2 + \frac{i\omega}{D}$ and the coefficients C and E are obtained from a particular solution of Equation (16.12):[1]

$$C = \frac{\chi P_0}{2\pi K}\frac{(1-R)\alpha}{(\beta^2 - \alpha^2)}\frac{\exp(-\frac{(\delta a)^2}{8})}{1 - R^2\exp(-2\alpha L)} \tag{16.14}$$

$$E = -\frac{\chi P_0}{2\pi K}\frac{(1-R)\alpha R\exp(-2\alpha L)}{(\beta^2 - \alpha^2)}\frac{\exp(-\frac{(\delta a)^2}{8})}{1 - R^2\exp(-2\alpha L)} \tag{16.15}$$

The solution of the homogeneous part of Equation (16.12) gives the other two coefficients in Equation (16.13) by considering the following boundary conditions:

$$K\left.\frac{\mathrm{d}T}{\mathrm{d}z}\right|_{z=0} = hT(z=0) \tag{16.16}$$

$$K\frac{\mathrm{d}T}{\mathrm{d}z}\bigg|_{z=L} = hT(z=L) \tag{16.17}$$

where h is the linearized combined coefficient of convective and radiative heat transfer.[2],[3] Accordingly, the temperature field given in Equation (16.13) can be written as

$$T(r,z,\omega) = \frac{\chi P_0}{2\pi K}\frac{(1-R)\alpha}{1-R^2\exp(-2\alpha L)}\cdot \tag{16.18}$$

$$\int_0^\infty \delta J_0(\delta r)\frac{\exp(-\frac{(\delta a)^2}{8})}{(\beta^2-\alpha^2)}\left[\frac{A_0\exp(\beta z)+B_0\exp(-\beta z)}{E_0}+\exp(-\alpha z)- \tag{16.19}$$

$$R\exp(-\alpha L)\cdot\left(\frac{A_1\exp(\beta z)+B_1\exp(-\beta z)}{E_0}+\exp(\alpha z)\right)\right]\mathrm{d}\delta$$

Where

$$A_0 = \exp(-\beta L)\left(\beta-h'\right)\left(\alpha+h'\right)+\exp(-\alpha L)\left(\beta+h'\right)\left(-\alpha+h'\right), \tag{16.20}$$

$$B_0 = \exp(\beta L)\left(\beta+h'\right)\left(\alpha+h'\right)+\exp(-\alpha L)\left(\beta-h'\right)\left(-\alpha+h'\right), \tag{16.21}$$

$$A_1 = \exp(-\beta L)\left(\beta-h'\right)\left(-\alpha+h'\right)+\exp(\alpha L)\left(\beta+h'\right)\left(\alpha+h'\right), \tag{16.22}$$

$$B_1 = \exp(\beta L)\left(\beta+h'\right)\left(-\alpha+h'\right)+\exp(\alpha L)\left(\beta-h'\right)\left(\alpha+h'\right), \tag{16.23}$$

$$E_0 = \exp(-\beta L)\left(\beta-h'\right)^2-\exp(\beta L)\left(\beta+h'\right)^2 \tag{16.24}$$

and $h' = {}^{h}/_{k}$.

The semi-transparency of the film in the infrared (IR) region is also considered. In particular, for the detection range of our IR camera (7.5 – 14 µm), the signal is collected not only from the sample surface but also from the inside.

Let us consider an effective IR absorption coefficient γ, which averages the IR absorption over the detection range of our IR camera. This effective coefficient is a good approximation for materials with smoothly varying IR absorption spectra in the corresponding detection range.[4] Accordingly, the signal recorded by the IR camera can be written as,

$$S(r,z_0,\omega) = F\int_0^L \gamma\exp(-\gamma|z-z_0|)T(r,z,\omega)\,\mathrm{d}z, \tag{16.25}$$

where $z_0 = \{0, L\}$ for the IR camera placed in the same side (front-face configuration) or in the backside (rear-face configuration) of the illuminated surface, respectively. The proportionality constant F includes the effect of the IR emissivity, sensor area, IR detection range and the derivative of the Planck distribution at room temperature.

Setting $z_0 = 0$ in Equation (16.25) allows us to evaluate the resulting integral and obtain an expression for the signal recorded by the IR camera, in the front-face configuration (see Equation (16.1) in main text). Similarly, to the previous case, we set $z_0 = L$ and evaluate the integral in Equation (16.25). We obtain the following expression for the signal recorded by the IR camera, in the rear-face configuration (see Equation (16.2) in the main text).

To validate our simulations in Chapter 16.4, for a wide range of both absorption coefficients, we perform simulations at three different values of the IR absorption coefficient ($\gamma L = 0.10$, 5 and 50). For each case, the optical absorption coefficient to the exciting wavelength was varied over a wide range: $0.01 \leq \alpha L \leq 100$. The product between the amplitude derivative and phase derivative is depicted in 16.6a-c. Note that for $|r| \geq \mu$, the product of the derivatives is constant, independent of the optical absorption at the exciting wavelength and for each of the three different values of γL. This is analogous to the product of the slopes $m_{\ln(|T\sqrt{(r)}|)} \cdot m_\psi = {}^{\pi f}/_{D}$ for the optically opaque case.[5] Actually, the optically opaque case is covered in our simulations, as represented in 16.6c for large values of αL. 16.6d-f show the product of derivatives for three different values of the optical absorption to the exciting wavelength ($\alpha L = 0.10$, 5 and 50), covering a wide range of IR absorption coefficients $0.01 \leq \gamma L \leq 100$. Once again, for $|r| \geq \mu$, the product of the derivatives is constant, independent of the IR optical absorption and for each of the three different values of αL. The optically opaque case is covered in 16.6f for large values of γL.

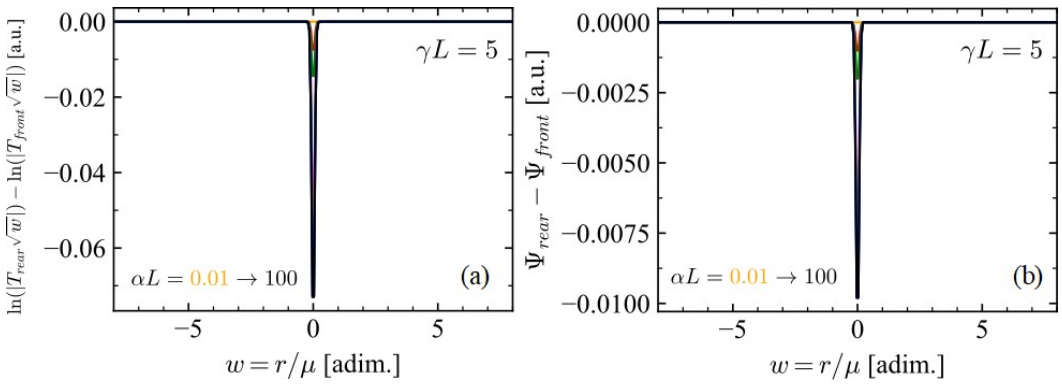


Fig. 16.6.: Difference in (a) amplitude and (b) phase between the surface temperature at $z = L$ (rear-face) and the front surface temperature at $z = 0$. Note that differences appear only near to the excitation spot, but for $|r| \gtrapprox \mu$ both temperatures are equal.

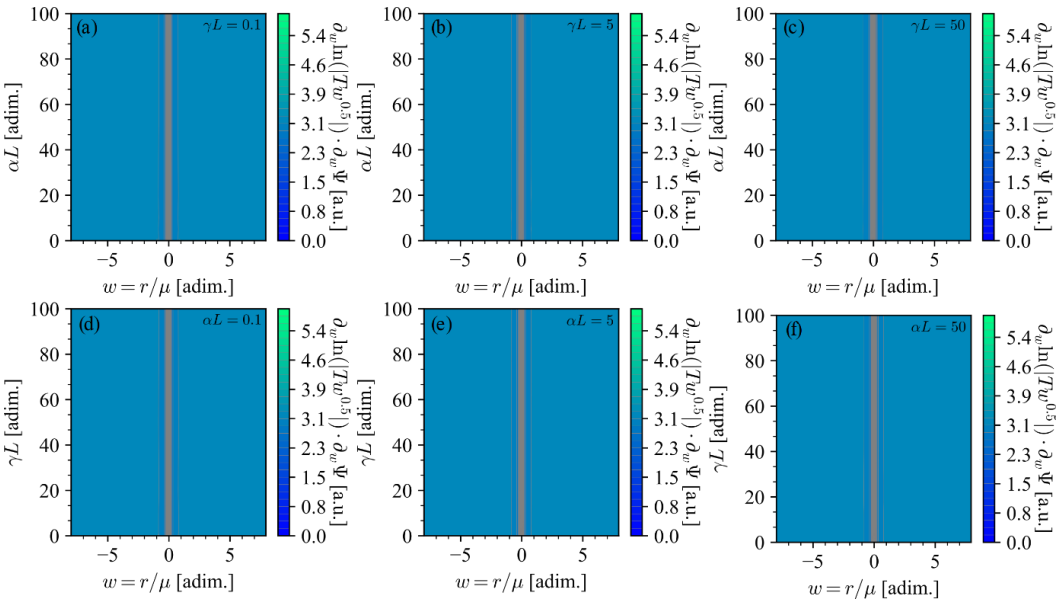


Fig. 16.7.: Product between amplitude and phase derivatives. Upper row (a)-(c): three different values of the IR absorption coefficient ($\gamma L = 0.10$, 5 and 50), covering a wide range of optical absorption coefficients to the exciting wavelength $0.01 \leq \alpha L \leq 100$. Lower row (d)-(f): three different values of the optical absorption to the exciting wavelength ($\alpha L = 0.10$, 5 and 50), covering a wide range of IR absorption coefficients $0.01 \leq \gamma L \leq 100$.

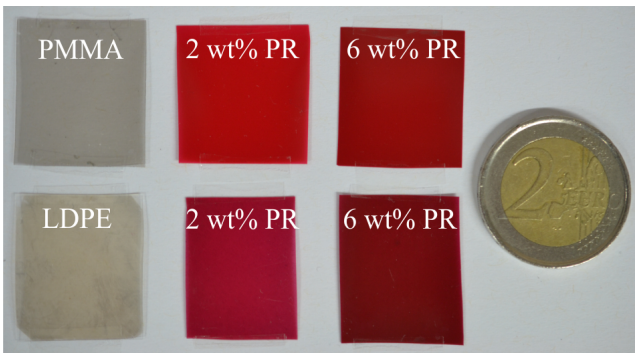


Fig. 16.8.: Photographs of LIT samples. Upper row: PMMA film with 20 nm carbon, PMMA film with 2 and 6 wt% PR, respectively. Lower row: LDPE film with 20 nm carbon, LDPE film 2 and 6 wt% PR, respectively.

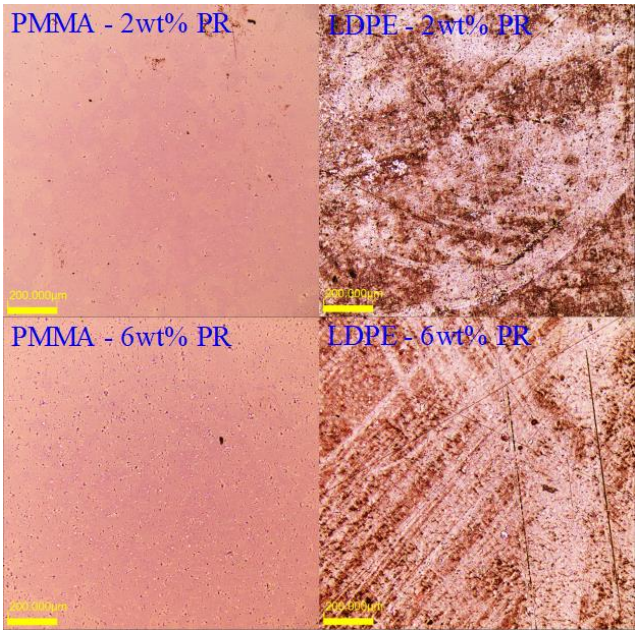


Fig. 16.9.: Micrographs of LIT samples with PR. Upper row: PMMA and LDPE films with 2 wt% PR. Lower row: PMMA and LDPE films with 6 wt% PR.

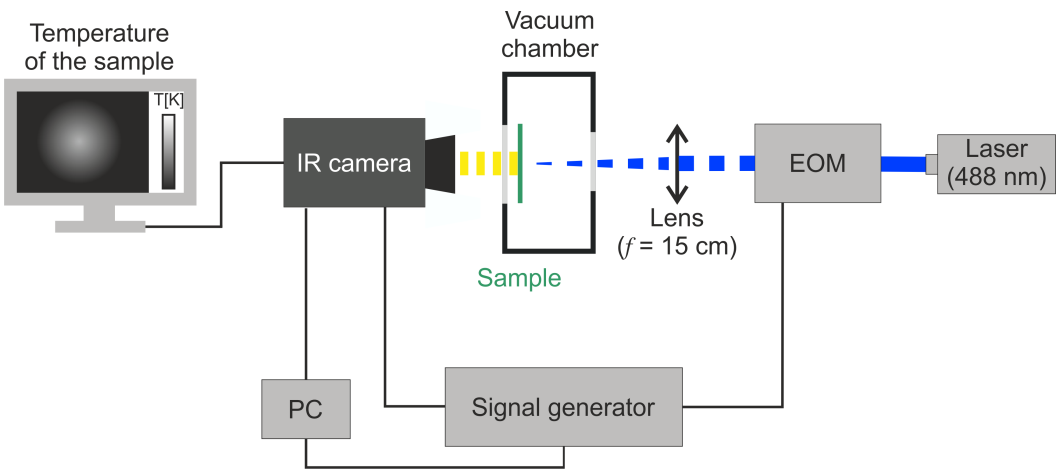


Fig. 16.10.: Schematic set-up used for lock-in thermography measurements. PC and EOM stand for personal computer and electro optic modulator, respectively

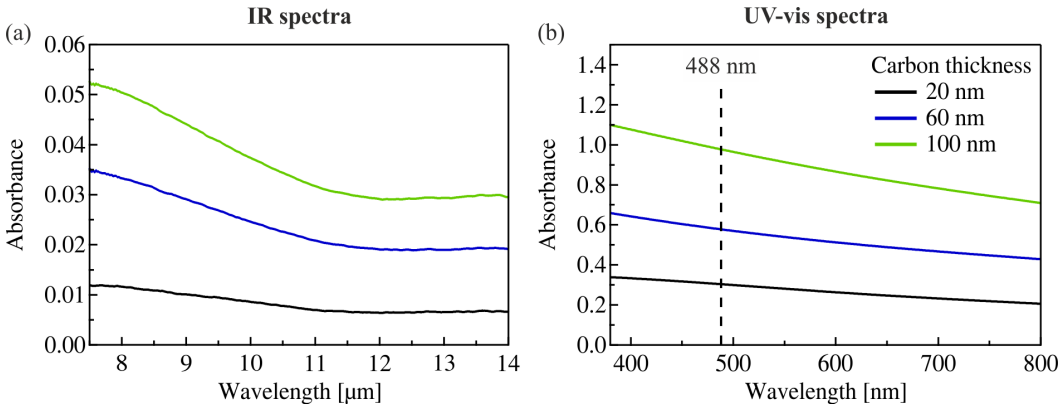


Fig. 16.11.: Influence of carbon coating on the optical properties: (a) IR spectra and (b) UV-vis spectra of 20 nm, 60 nm, and 100 nm carbon (The spectra are represented relative to an uncoated KBr disc as control). The dashed line at 488 nm marks the wavelength of the incident laser used in lock-in measurements. The absorbance of IR and UV-vis light increases with an increasing carbon layer. The coating leads to a higher absorbance of light in the UV-vis range compared to IR light.

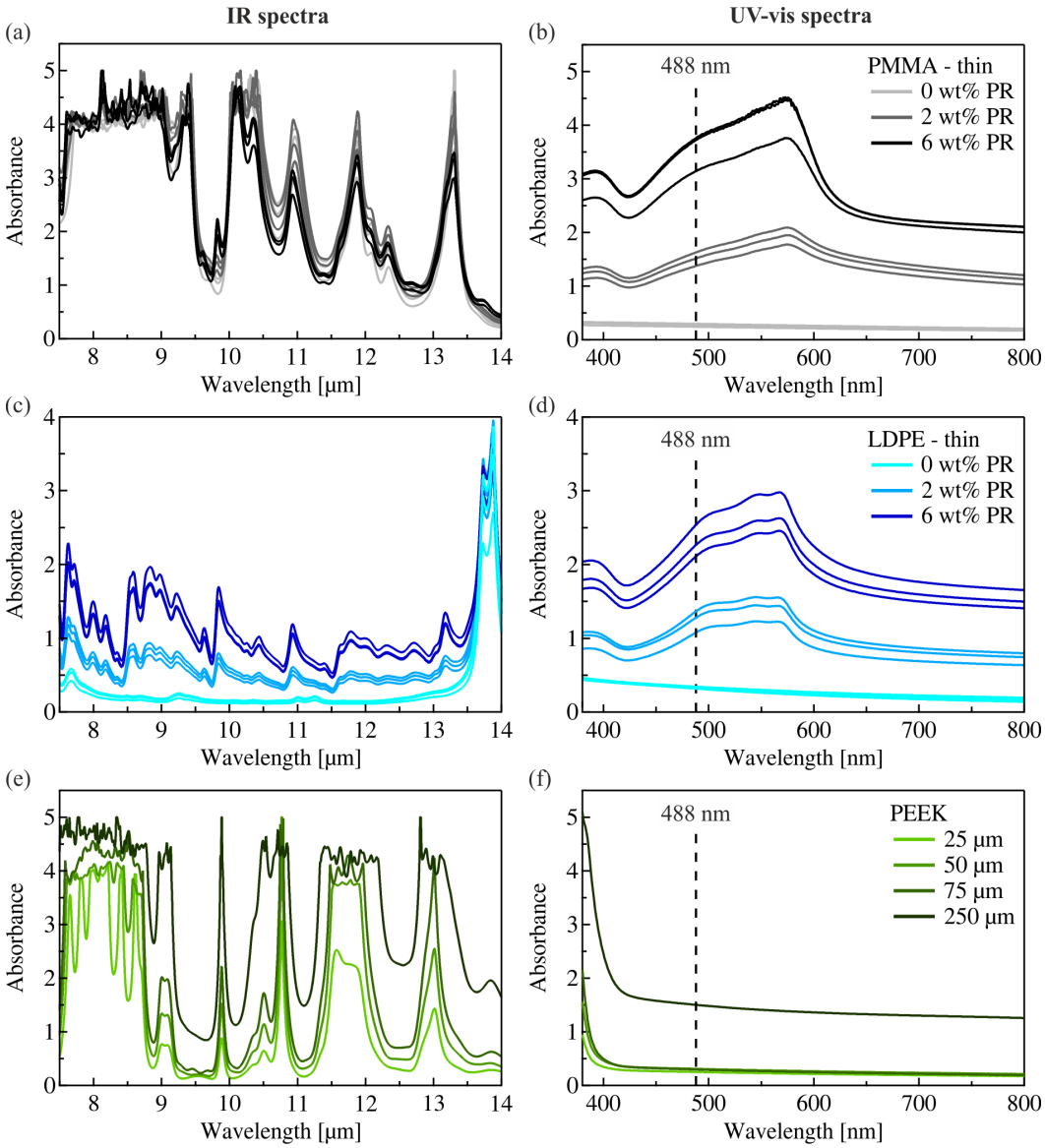


Fig. 16.12.: Optical properties of the thin, free-standing polymeric films measured with lock-in thermography. IR spectra and (b) UV-vis spectra of PMMA films with 0 wt%, 2 wt%, and 6 wt% PR, respectively. (c) IR spectra and (c) UV-vis spectra of LDPE films with 0 wt%, 2 wt%, and 6 wt% PR, respectively. (d) IR spectra and (e) UVvis spectra of a 25 µm, 50 µm, 75 µm, and 250 µm PEEK film, respectively. The dashed line at 488 nm marks the wavelength of the incident laser used in lock-in measurements. Absorbance values higher than 4 units are difficult to resolve accurately with our detector. Therefore, we set the opaqueness threshold to $A_{\mathrm{Vis}} > 4$ (for the excitation wavelength) and $A_{\mathrm{IR}} > 4$ (in the IR range).

Experimental section

Reference thermal diffusivity measurements of thick PMMA and LDPE samples

1) Preparation of thick reference samples for Xenon flash analysis

1 mm thick PMMA samples were fabricated by compounding and injection molding. PMMA pellets and PR powder were mixed under N_2 flow in a twin-screw compounder with a stirring speed of 40 rpm and at a temperature of 240 °C. Then, the compounded material was directly filled into the injection unit. Disks with a diameter of 27 mm and a thickness of 1 mm were fabricated using an injection force of 6 kN and a tool temperature of 20 °C. In this way, PMMA disks with 0 wt%, 2 wt%, and 6 wt% PR, respectively, were prepared. LDPE samples were taken from the remaining LDPE disks which were already prepared using injection molding (see Experimental section in the main text).

2) Cross-plane thermal diffusivity measurements using Xenon flash analysis (XFA)

Xenon flash analysis (XFA) was used to obtain reference thermal diffusivity values for the fabricated PMMA and LDPE samples. We performed cross-plane thermal diffusivity measurements on an XFA500 Xenon flash apparatus (Linseis), equipped with an InSb infrared (IR) detector. The upper and lower sample surfaces were coated with a thin graphitic layer (each 10 µm). Since the thickness of the coating is significantly thinner than the sample thickness (20 µm $\ll$ 1000 µm), it does not affect the measurement result. The coating ensures a good absorbance of the Xenon flash and a high IR emissivity. The sample thickness which is needed to obtain the thermal diffusivity was measured with a Litematic VL-50 (Mitutoyo). All XFA measurements were performed in air at room temperature (295 K). The LDPE samples were measured before and after thermal annealing (90 °C, one week).

3) X-ray diffraction (XRD) and small angle X-ray scattering (SAXS)

To demonstrate the amorphous nature of the PMMA samples and the semi-crystallinity of the LDPE samples we performed X-ray diffraction (XRD). The measurements

were conducted in Bragg-Brentano geometry on an Empyrean system with a PIXEL solid-state detector (PANalytical, Almelo, Netherlands) using Cu-radiation.

Small-angle X-ray scattering (SAXS) was performed on the PMMA and LDPE samples using the lab-based system Ganesha Air (SAXSLAB, Denmark), equipped with a rotating anode (copper, MicroMax 007HF, Rigaku Corporation, Japan) and a position sensitive detector (Pilatus 300K, Dectris). The samples (and air as background) were measured as obtained in parallel and perpendicular beam geometry. Different detector positions were used to cover a larger scattering range q ($q = \frac{4\pi}{\lambda} \sin\left(\frac{2\theta}{2}\right)$).

4) Discussion of polymer microstructure

The pure PMMA samples show broad oscillations without prominent Bragg reflections in the XRD regime proving their amorphous nature (Figure 16.13, black curve). The broad oscillations around 13.5°, 30° and 42° result from the packing and defined backbone sequence of the polymeric chains. These characteristic patterns are also obtained for the dye-doped PMMA samples (Figure 16.11, red curve, and Figure 16.13a, b), which additionally show small reflections due to tiny crystallites of phenol red. The combination of XRD and SAXS of a pure PMMA sample (Figure 16.13, inset) indicates a rough surface of the films ($I(q) \sim q^{-2.8}$). In that context, one may imagine the texture of the sample area as a dense fractal-like object.

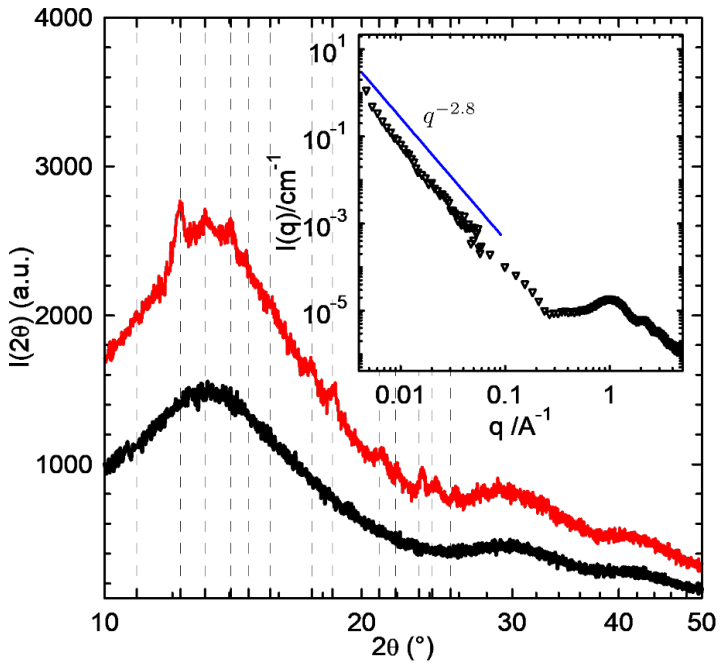


Fig. 16.13.: XRD pattern of thick PMMA samples without PR (black line) and with 6 wt% PR (red line) as well as the expected pattern of phenol red (gray vertical dashed lines). The inset shows the X-ray intensity of the pure PMMA sample over a wider q range (triangles, SAXS and XRD) and a theoretical intensity calculated for a mass fractal (blue line)

As can be seen in Figure 16.14a and b, the XRD patterns reveal no significant differences in the polymeric structure between the thin and thick PMMA samples prepared for lock-in thermography and Xenon flash analysis, respectively. In both cases, the small Bragg reflections base on the incorporated PR dye and increase slightly with increasing concentration of the dye. This shows that all samples exhibit a similar polymeric microstructure independent of the PR content, sample thickness, and preparation method (injection molding vs. doctor-blading).

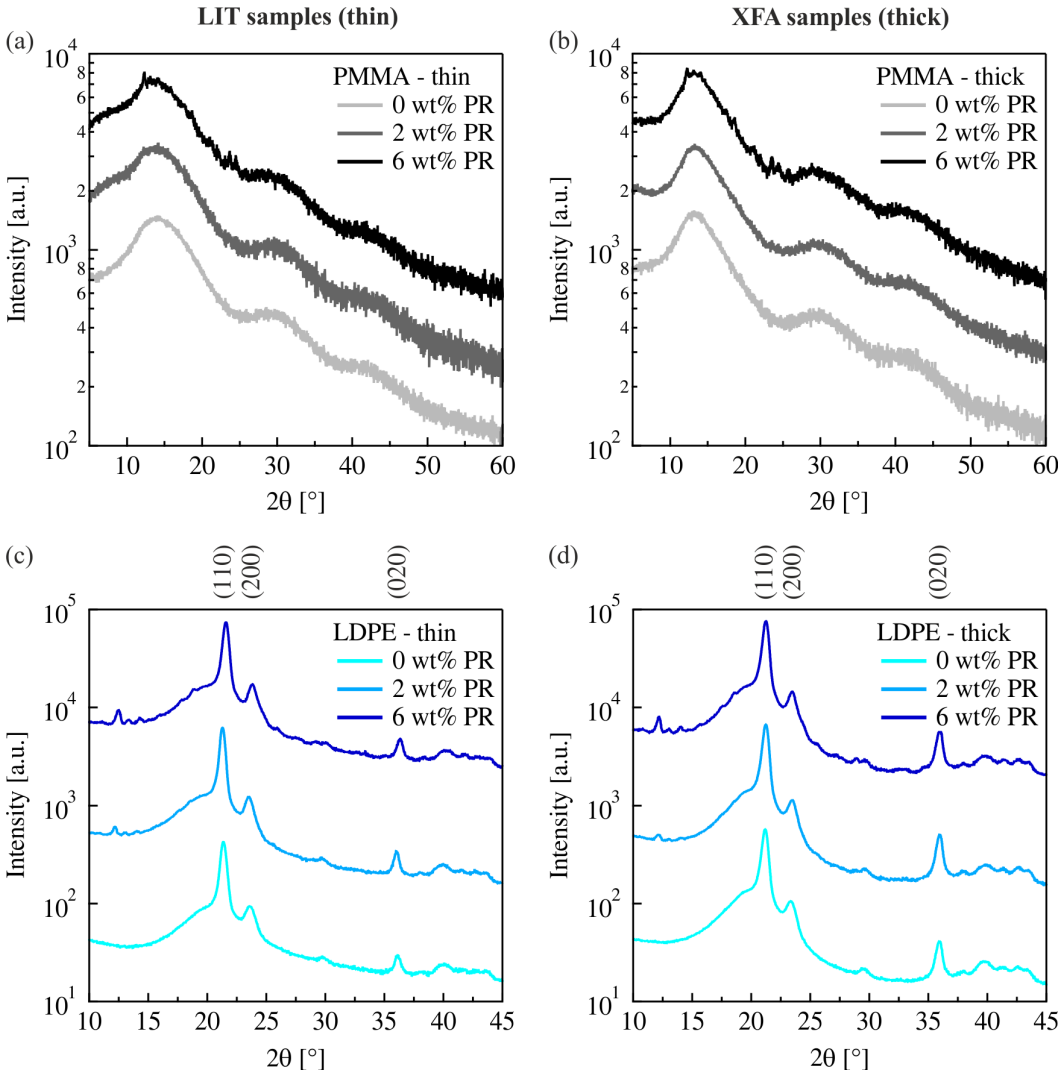


Fig. 16.14.: XRD curves of amorphous PMMA and semi-crystalline LDPE: (a) thin PMMA used for lock-in thermography (LIT) measurements, and (b) thick PMMA for Xenon flash analysis (XFA), (c) thin LDPE for LIT measurements, and (d) thick LDPE for XFA. Curves are presented with a vertical shift.

LDPE is a low-density, semi-crystalline polymer. The semi-crystallinity of the LDPE films is visible by the pronounced Bragg reflexes at 21.2° (100 % peak), 23.3° and 36.0° (Figure 16.14c and d). The reflections correspond to [110], [200], and [020] of space group Pnam. As expected, the PR doped films show additionally small Bragg spots due to the dye.

We found comparable degrees of crystallinity (∼ 45%) for all LDPE samples. Consequently, the addition of phenol red does not affect the degree of crystallinity of the LDPE samples. Furthermore, also the subsequent hot-pressing of the injection molded LDPE disks does not change the degree of crystallinity.

Usually, one expects for a semi-crystalline film that

(i) the amorphous and crystalline regions are randomly distributed over the solid material (isotropy) and that

(ii) the small crystallites are polydisperse.

Nevertheless, due to external forces like shear stress or elevated temperatures phase segregation between the amorphous phase and the crystallites cannot be ruled out.

Consequently, we performed SAXS experiments in parallel and perpendicular beam geometries to study the in-plane and cross-plane polymeric microstructures, respectively. Figure 16.15 shows an example of such a study for the LDPE samples with 6 wt% PR. Astonishing, the SAXS patterns taken perpendicular exhibit a broad reflection at $q \sim 0.04$ Å^{-1}, which corresponds to a correlation length of about 15 nm and hints to a two-phase LDPE system (segregation of crystallites and amorphous LDPE). This feature is anisotropic spot-like for the thick sample (Figure 16.15a) and isotropic for the thin film (Figure 16.15b). As a consequence, the crystallites of the thick samples have a stronger tendency to align lamellar-like parallel to the sample surface (i.e., in in-plane direction) compared to the thin samples. In the mesoscopic scattering signals observed for the thick and thin samples in parallel beam geometry (Figure 16.15c and 16.15d), the oscillation at $q \sim 0.04$ Å^{-1} is attenuated, and the patterns appear more isotropic. Thus, the thick samples (prepared for XFA) and thin films (prepared for LIT) exhibit a different microstructure despite their comparable degree of crystallinity. Both samples show a preferred orientation of the crystallites in the in-plane direction and more random orientation in the crossplane direction. However, the anisotropy is much more pronounced for the thick samples. This difference in the microstructure may be associated with the different fabrication processes and the applied temperature gradients (injection molding vs. injection molding and subsequent hot-pressing).

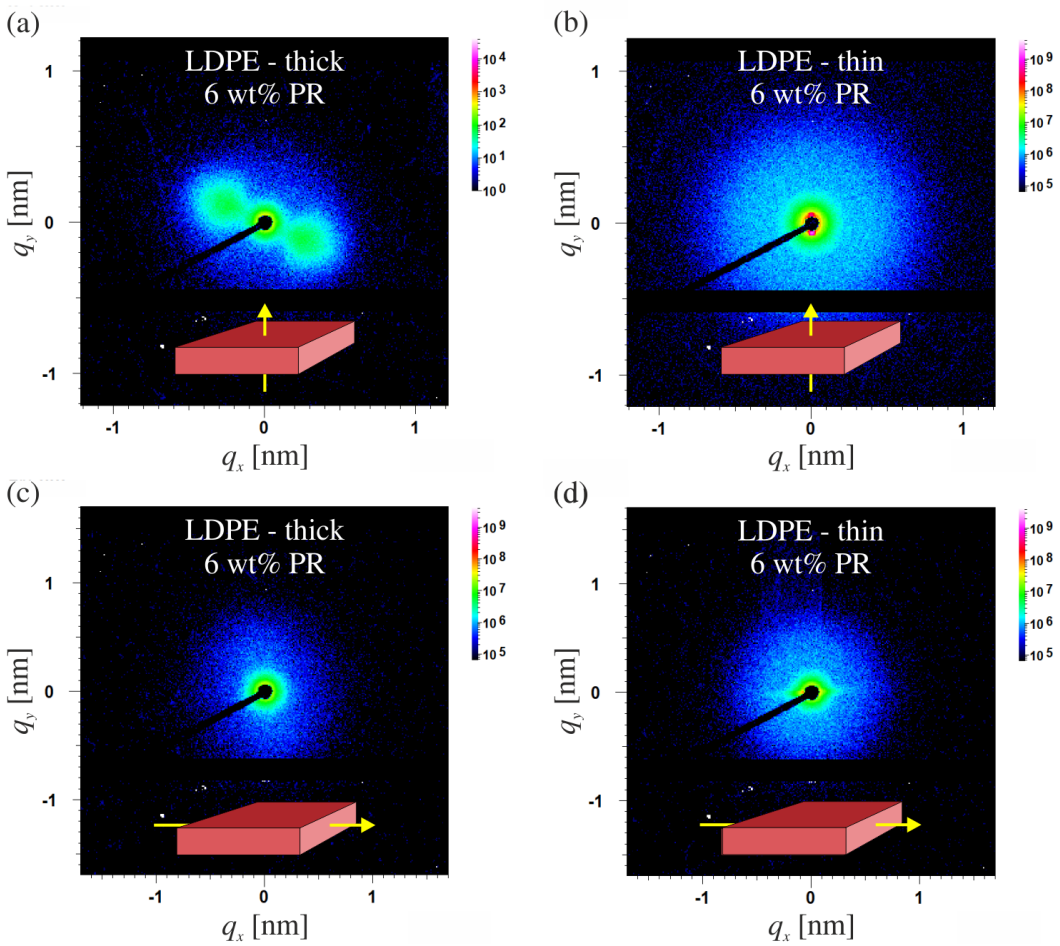


Fig. 16.15.: 2D SAXS patterns of LDPE films with 6 wt% PR in perpendicular (a, b) and parallel (c, d) beam geometry (see sketch: sample with beam direction).

In Figure 16.16 the two-dimensional SAXS data of a thick LDPE sample as used for XFA and after additional annealing at 90 °C for one week are compared to illuminate the effect of the temperature/fabrication method on the orientation of the crystallites. All annealed samples show a significantly more preferred orientation of the crystallites compared to the non-annealed state. Note, that the spots appear even triangular for the data of the annealed film in perpendicular geometry and at the same time, the amorphous halo is less pronounced. We conclude that a direction dependent crystal growth is introduced by thermal annealing. The same trend is observed in case of the thin films, but much less pronounced.

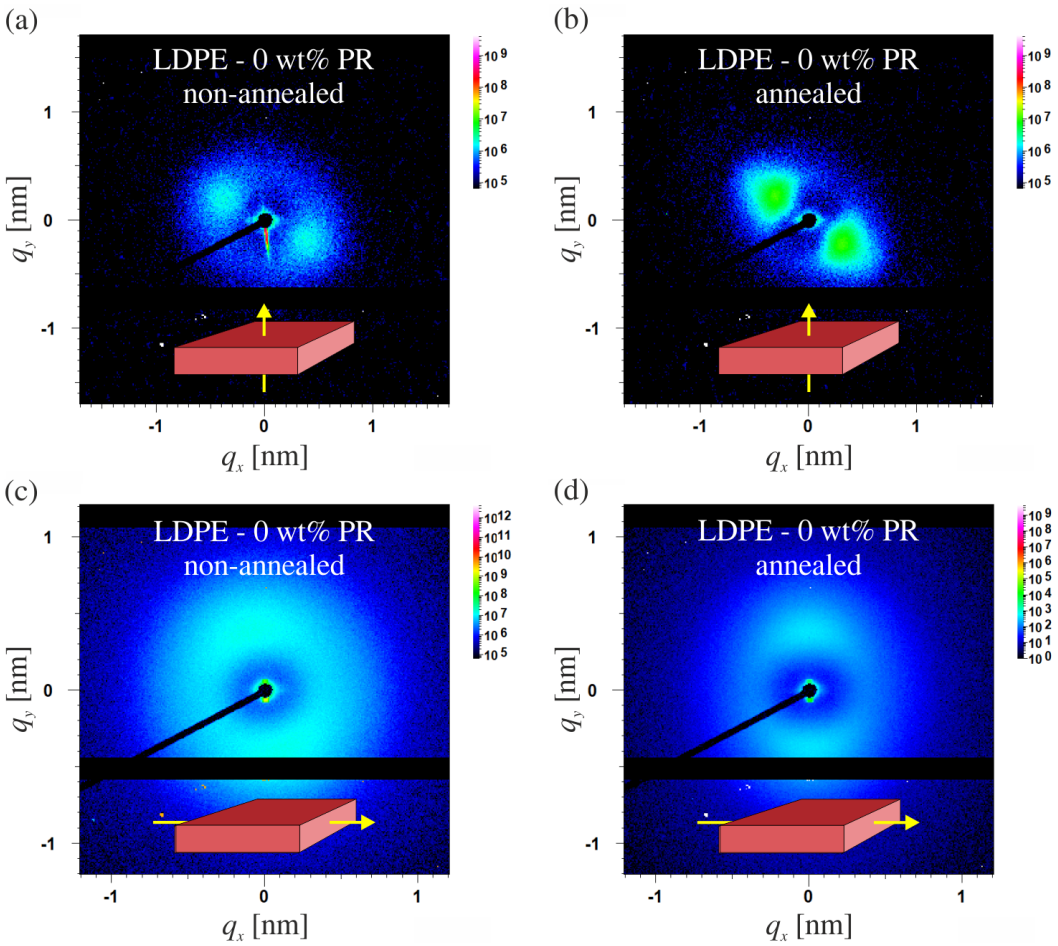


Fig. 16.16.: 2D SAXS patterns of thick LDPE samples (for XFA) with 0 wt% PR in perpendicular (a, b) and parallel (c, d) beam geometry (see sketch: sample with beam direction)

Comparison of thermal diffusivity data (LIT vs. XFA)

Figure 16.17 summarizes the thermal diffusivity values obtained for the fabricated PMMA and LDPE samples with 0, 2, and 6 wt% PR, respectively. The *in-plane* thermal diffusivity of the thin, free-standing samples was determined using lock-in thermography (LIT). The 1 mm thick reference samples were characterized using Xenon flash analysis (XFA). It is to be noted that this measurement technique gives the *cross-plane* thermal diffusivity.

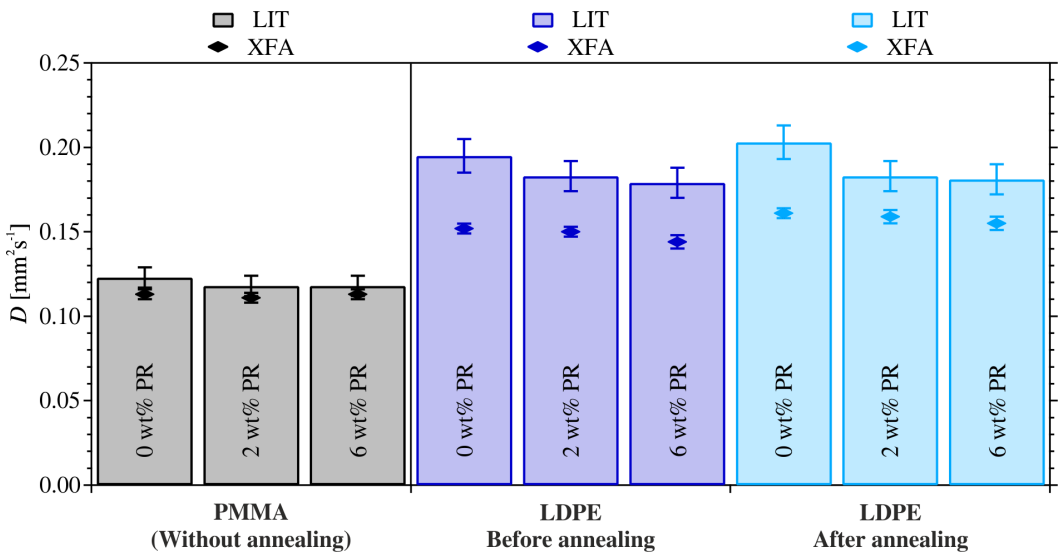


Fig. 16.17.: Thermal diffusivity D of PMMA and LDPE with 0, 2, and 6 wt% phenol red (PR), respectively. The thin samples ($\approx$ 200 ţm) were measured with lock-in thermography ((LIT), bars) that determines the in-plane thermal diffusivity; the thick samples ($\approx$ 1 mm) were analyzed with Xenon flash analysis ((XFA), markers) that gives the cross-plane thermal diffusivity. LDPE samples were measured before and after thermal annealing (90 °C, one week). The error bars arise from 5 % uncertainty (LIT) and from the standard deviation of the averaged value (XFA). Two to three samples were measured in each case.

As discussed in the main text, we obtain an in-plane thermal diffusivity of $\sim$ 0.12 mm^{2}s^{-1} for all PMMA samples independent of the PR content. The cross-plane values obtained from XFA are $\sim$ 0.11 mm^{2}s^{-1}, and thus close to the data obtained from LIT. This good agreement confirms that the microstructure of the amorphous polymer is similar (see XRD curves in Figure 16.13, Figure 16.14a and 16.14b) leading to isotropic thermal diffusivity values.

However, in the case of LDPE (before annealing), we extracted significantly lower cross-plane thermal diffusivities from XFA ($\sim$ 20 %). We relate this discrepancy to the difference in the underlying microstructure as discussed above. Since the degree of crystallinity of all samples is the same, the preferred in-plane orientation leads apparently to a reduced cross-plane thermal diffusivity as measured by XFA. Thermal annealing at 90 °C for one week led to a more pronounced anisotropy within the thick films. In comparison to the values before thermal annealing, we obtained similar in-plane thermal diffusivity values from LIT and slightly higher cross-plane thermal diffusivity values from XFA after annealing. In summary, we found that comparable in-plane and cross-plane thermal diffusivity values are obtained for an amorphous polymer independent of the fabrication method. In case of a semicrystalline polymer, the fabrication method has a strong influence on the polymer microstructure, and thus on the in-plane and cross-plane thermal diffusivity, which renders a quantitative comparison difficult.

References

[1] N. W. Pech-May, A. Mendioroz, and A. Salazar, „Simultaneous measurement of the in-plane and in-depth thermal diffusivity of solids using pulsed infrared thermography with focused illumination,“ *NDT & E International*, vol. 77, pp. 28–34, 2016.

[2] S. O. Kanstad and P.-E. Nordal, „Experimental aspects of photothermal radiometry,“ *Canadian Journal of Physics*, vol. 64, no. 9, pp. 1155–1164, 1986.

[3] A. Salazar, A. Mendioroz, and R. Fuente, „The strong influence of heat losses on the accurate measurement of thermal diffusivity using lock-in thermography,“ *Applied Physics Letters*, vol. 95, no. 12, p. 121 905, 2009.

[4] H. G. Walther, U. Seidel, W. Karpen, and G. Busse, „Application of modulated photothermal radiometry to infrared transparent samples,“ *Review of Scientific Instruments*, vol. 63, no. 11, pp. 5479–5480, 1992.

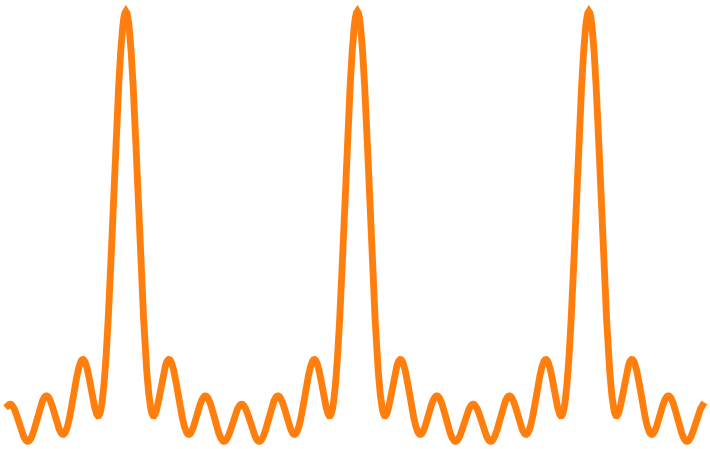
[5] A. Mendioroz, R. Fuente-Dacal, E. Apiñaniz, and A. Salazar, „Thermal diffusivity measurements of thin plates and filaments using lock-in thermography,“ *Review of Scientific Instruments*, vol. 80, no. 7, p. 074 904, 2009.

17

Reducing DC Heating in Lock-in Thermography: the Advantage of Thermal Multiplexing

Bernd A. F. Kopera and Markus Retsch

Department of Chemistry, Physical Chemistry I, University of Bayreuth,
Universitätsstraße 30, 95447 Bayreuth, Germany



in preparation

17.1 Abstract

Many modern measurement methods for heat transfer work in the frequency domain. The sample is excited periodically with a sine wave and the amplitude and phase of the temperature response is recorded. A certain amount of DC heating is unavoidable if the sample can only be heated, e.g. by an intensity modulated light source. This DC temperature offset influences the measured thermal properties because they are in general temperature dependent. We explore square and multiplexed excitation signals with the goal of reducing DC heating. We apply these signals to lock-in thermography and show the feasibility of a simultaneous measurement at multiple frequencies. In addition, we propose the use of the Goertzel algorithm to efficiently extract individual spectral components from the measured signal.

17.2 Introduction

Many modern measurement methods for heat transfer work in the frequency domain. The sample is heated periodically and the amplitude and phase of the temperature response is recorded. Periodic signals can be measured with high accuracy by lock-in detection. Frequency domain techniques are therefore advantageous over impulse[1],[2] or constant perturbation[3]–[5] techniques.

Examples for frequency domain techniques are the photoacoustic technique,[6],[7] the 3ω technique,[8]–[12] frequency domain thermo reflectance,[13],[14] and lock-in thermography.[15]–[19] All these techniques have in common that they use a frequency sweep. The change of amplitude or phase vs. frequency is analyzed to obtain the thermal properties of the sample. The most common excitation signal is a pure sine wave. Sine waves need to be offset by their amplitude to generate a purely positive signal. Half of the heating power is therefore wasted in the DC component.

DC heating is unavoidable if the sample can only be heated and not cooled, e.g. by an intensity modulated light source. However, we should strive to minimize the DC heating as much as possible. The thermal properties of most materials are temperature dependent.[20] Reducing the DC temperature offset is therefore important to measure thermal properties at the intended temperature.

Our contribution addresses this issue. We present two optimized excitation signals for frequency domain techniques that contain multiple frequencies. A square wave, with shorter duty cycle, and a sum of sine waves (multiplexed). The square wave is simple to generate, whereas the sum of sine waves offers more degrees of freedom.

These degrees of freedom come at the cost of arbitrary signal generation. Arbitrary intensity modulated light or heat sources are readily available nowadays. We modulate the intensity of our diode laser directly with an analog voltage signal. Electro- and acousto-optic modulators can impose arbitrary intensity modulations on a laser beam. Electric heating methods, like the 3ω technique, can generate an arbitrary heating signal with an analog current source. A multi frequency 3ω method was recently developed by Dames *et al.*[12]. They measure at multiple frequency to increase the detection speed and accuracy of the 3ω method.

We focus on lock-in thermography to proof the reduction in DC heating. However, we think that our signal design can be adapted for different frequency domain techniques. A necessary condition is that signals at different frequencies do not influence each other.

Lock-in thermography[21] (LIT) is a well established thermal characterization method for thin, freestanding films and fibers. The sample is periodically heated with an intensity modulated light source. Common sources are flash lamps and point or line shaped laser beams. An IR camera detects the position dependent amplitude and phase of the resulting temperature oscillations on the sample surface. The thermal diffusivity is then computed from the slope of the linearized amplitude and phase vs. distance from the heat source.[15],[16],[22]

LIT is also used for non-destructive testing. An example is pulsed phase thermography[23],[24] (PPT) where a short pulse, instead of a sine wave is used. The short pulse contains a continuous excitation spectrum. The penetration depth of thermal oscillations depends on their frequency. Higher frequencies decay faster and penetrate less into the sample. This effect can be used to sense material defects at different sample depths by analyzing different spectral components in the pulse spectrum. A different excitation signal has been proposed by Chatterjee *et al.*[25] and Tuli *et al.*[26] They describe the use of a frequency modulated signal. They use a chirp pulse to excite the sample. Chirps change their frequency continuously during the measurement. This creates a continuous spectrum with a limited frequency range. Chirps are therefore similar to impulse excitations used in PPT.

However, all these signal have two drawbacks:

- they create an unnecessary large DC temperature offset and
- the signal power is not optimally distributed to the spectral components that are analyzed.

We want to concentrate the signal power to the specific frequencies that are later analyzed. This should increase the signal-to-noise ratio and allow us to reduce the measurement time.

We also measure multiple excitation frequencies simultaneously. This is possible for lock-in thermography, because heat and temperature spread diffusively through a solid material over large length scales. The heat equation for isotropic materials,

$$\frac{\partial}{\partial t} T(x, y, z, t) = \alpha \cdot \vec{\nabla}^2 T(x, y, z, t), \tag{17.1}$$

is a linear differential equation. A linear combination of two solutions is again a solution. In other words, heat fluxes from two different sources do not interfere with each other, even if they are on the same spot. The overall heat flux is the vector sum (superposition) of the two heat fluxes. The temperature differences from two heat sources are added (superimposed) to yield the overall temperature difference. The superposition principle is preserved in the frequency domain. Two periodic thermal excitations do not influence each other, especially when they have different frequencies.

Finally, we propose the use of the Goertzel algorithm[27] to efficiently extract individual spectral components from the measured signal. Current approaches use either the definition of the Fourier transform[26] or the fast Fourier transform algorithm (FFT). Computing the entire spectrum is inefficient, if we are only interested in a few, well defined spectral components. In addition, the Goertzel algorithm can be executed in real time, processing each measurement point in time sequentially. In contrast, the FFT algorithm[28] requires all points in time simultaneously. This can be a bottleneck for measurements with large amounts of data like high speed and high resolution thermography.

17.3 Methods

17.3.1 Our Lock-in Thermography Setup

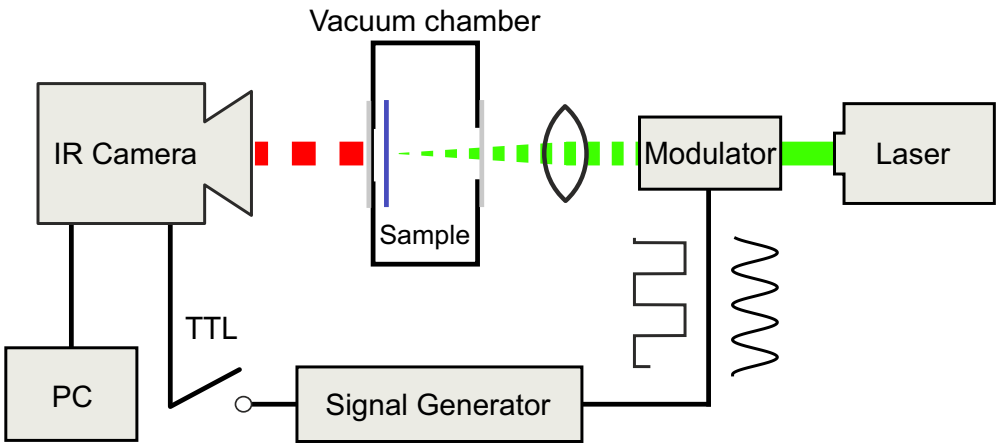


Fig. 17.1.: Components of our lock-in thermography setup. An intensity modulated, line shaped laser beam is focused on the, freestanding thin film inside the vacuum chamber. The laser periodically heats the sample. The intensity of our diode laser is modulated with an analog signal (e.g. sine or square) from the signal generator. The IR camera can be phase locked to the signal generator via a TTL signal. Lock-in detection is handled by the software.

The components of our lock-in thermography setup are shown in Figure 17.1. We use a VarioCAM HD 875 from Infratec for thermal imaging with a resolution of 1024 × 768 pixels. Our IR camera detects wavelengths ranging from 7.5 to 14 µm. The pixel size is 29 µm at a working distance of 33 mm. We placed the IR camera onto a micrometer controlled translation stage. This allows us to manually focus the camera by adjusting the camera-sample distance instead of changing the focus. We designed a custom vacuum chamber and sample holder. The vacuum chamber is mounted on a x-y-z micrometer translation stage for alignment. A rotary vane pump from Vacuubrand, type RZ 9 ensures a gas pressure of less than 10^{-3} hPa around the sample. The low gas pressure significantly reduces heat losses from the sample. The sample is periodically heated by the laser from the back. We use a diode laser type 13LR12-m125+55CM-520-56-O08-T15-PS-7 from Schäfter+Kirchhoff. This laser has a wavelength of 520 nm and creates a 40 µm ($^1/e^2$) laser line at a working distance of 122 mm. We create the analog modulation signal with a RIGOL DG1000 signal generator. In addition, we can generate a square wave TTL signal with the IR camera. This allows us to phase lock the laser modulation to the IR camera. We then extract the amplitude and phase images at specific frequencies with the Goertzel algorithm. The source code for our software is in the SI section 17.8.1.

17.3.2 Square Signals

Square signals are easy to generate. A simple shutter imposes an on/off intensity modulation onto almost any light source. The duty cycle, $D = t_{on}/T$, determines the on-time per period, T. The average power, and therefore the DC heating $(I_0 \cdot D)$ are directly proportional to the duty cycle. We reduce the DC component of a square wave by reducing the duty cycle. Square waves contain higher harmonics.

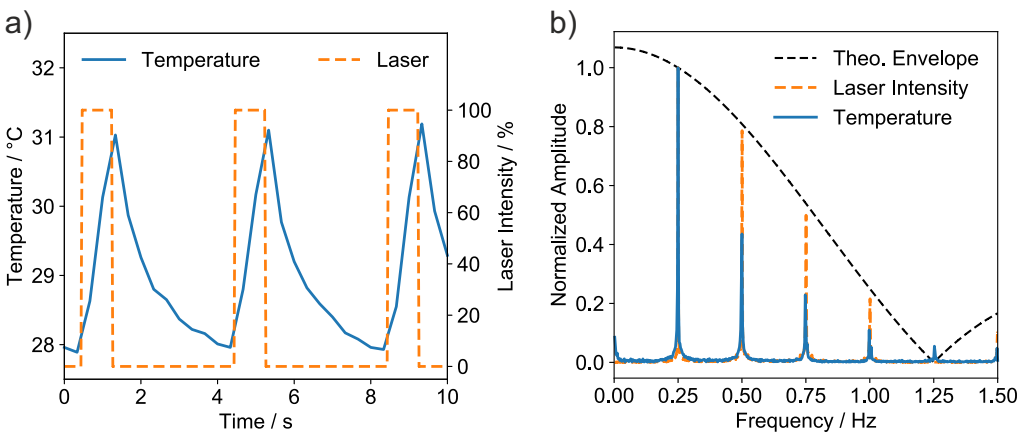


Fig. 17.2.: a) Laser intensity (dashed orange) for a square wave with 20 % duty cycle and a frequency of 0.25 Hz. The corresponding surface temperature near the laser line is shown in blue. Note that the temperature does not follow the square wave of the laser intensity. b) Amplitude spectrum of the surface temperature and laser power normalized to the 1st harmonic. Higher harmonics in the temperature are attenuated relative to the laser excitation. The dashed line represents the theoretical envelope (Eq. (17.3)) for square waves with constant duty cycle.

We can represent a square wave by its Fourier series (see SI section 17.7.2):

$$I(t) = I_0 \cdot \left[D + \sum_{n=1}^{\infty} 2 \cdot \frac{\sin(\pi n D)}{\pi n} \cdot \cos\left(\frac{2\pi n t}{T}\right) \right] \tag{17.2}$$

The amplitudes of the spectral components are therefore given by:

$$A_n = I_0 \cdot 2 \cdot \frac{\sin(\pi n D)}{\pi n} \tag{17.3}$$

We can analyze the higher harmonics during a lock-in thermography experiment. Figure 17.2a shows the resulting temperature signal during a lock-in thermography measurement created by a square wave excitation. Note, that the temperature does not follow the excitation signal. This distortion is evident in the spectrum (Fig. 17.2b). Higher harmonics in the temperature signal are attenuated relative to the excitation signal. The attenuation originates from the heat transfer in the sample.[25] However, the higher harmonics can still be used for analysis.

17.3.3 Multiplexed Signal Design

In this section, we design an excitation signal with multiple spectral components. We combine multiple intensity modulated signals into one. Our goal is to reduce the DC component. DC heating both wastes energy and creates a temperature offset in the sample. A signal,

$$I(t) = A_1 + A_1 \cdot \cos\left(2\pi f t + \phi\right), \tag{17.4}$$

with only one sine wave requires a DC offset equal to its amplitude, A_1, to be fully positive. We propose a superposition of N sine waves like

$$I(t) = \sum_{n=1}^{N} A_n \cdot \cos\left(2\pi f_n \cdot t + \phi_n\right) + s_n. \tag{17.5}$$

These sine waves have different frequencies, f_n, amplitudes, A_n, phases, ϕ_n. The offsets, s_n, are necessary to obtain a purely positive signal. Note that our signal is defined by a $4N$ dimensional parameter space. Although we want many degrees of freedom, these are usually too many. We therefore carefully reduce the number of free parameters for the frequencies, amplitudes, and offsets.

We begin with the offsets. The individual offsets, s_n, can be summed into a global offset, S. To do so, we split the signal into two sums

$$I(t) = \sum_{n=1}^{N} A_n \cdot \cos\left(2\pi f_n \cdot t + \phi_n\right) + \sum_{n=1}^{N} s_n. \tag{17.6}$$

The second sum becomes the global offset and we get

$$I(t) = S + \sum_{n=1}^{N} A_n \cdot \cos\left(2\pi f_n \cdot t + \phi_n\right). \tag{17.7}$$

The signal has to be always positive (only heating) or 0. Therefore, we have to set the global offset to the negative minimum of the signal without offsets:

$$S = -\min\left\{\sum_{n=1}^{N} A_n \cdot \cos\left(2\pi f_n \cdot t + \phi_n\right)\right\}. \tag{17.8}$$

This means that our signal is shifted in the positive y-direction by the amount S. The DC component is then $I_0 \cdot S$. The offset, S, is now fully determined by the frequencies, amplitudes, and phases we choose. In addition, the offset given by equation (17.8) is the optimal offset.

Optimal means that the DC component is minimal for a given set of frequencies, amplitudes, and phases. The minimal necessary offset in the *worst case* is given by the sum of the individual amplitudes:

$$S_{\text{worst case}} = \sum_{n=1}^{N} A_n \tag{17.9}$$

We can interpret this case as summing up individual sine waves that are already fully positive. There would be no benefit from the interference of partially negative sine waves.

Next, we describe the limitations around the frequencies. Choose as few frequencies as possible. Every additional frequency consumes part of the overall signal power. This in turn, dilutes the power per frequency. However, we can not choose arbitrary frequencies for two reasons:

1. The ratio of every frequency pair has to be an integer ratio. This ensures that the resulting signal is again periodic.

2. The discrete Fourier transform only yields amplitude and phase at certain, discrete frequencies. These frequency bins are defined by the number of samples (N_{frames}) and the sampling frequency (f_{sample}) of the IR camera. The available frequency bins, f_k, are

$$f_k = k \cdot \frac{f_{\text{sample}}}{N_{\text{frames}}} \text{ for all } k \text{ in } \left[1;\ \frac{N_{\text{frames}}}{2}\right]. \tag{17.10}$$

   The maximal frequency is limited by the Nyquist-Shannon sampling theorem[29] to half of the sampling frequency. Signals at frequencies between the bins leak into adjacent bins. This should be avoided to simplify the amplitude and phase extraction.

Choosing frequencies according to equation (17.10) automatically ensures rational frequency pairs. The period, T_0, of the resulting signal is then given by

$$T_0 = \frac{1}{\min\{f_n\}} \tag{17.11}$$

We now look for rules regarding the amplitudes at these frequencies. All frequencies are measured for the same duration. Our goal is to keep the signal-to-noise ratio, S/N, constant for all frequencies. The signal-to-noise ratio is proportional to the amplitude of the signal and the square root of the number of measured periods.

The number of periods is given by the ratio of the measurement time and the period of the spectral component. Overall we get

$$S/N \propto A_n \cdot \sqrt{\frac{t_m}{T_n}} = A_n \cdot \sqrt{t_m} \cdot \sqrt{f_n}. \tag{17.12}$$

The ratio of S/N for the $n-$th frequency relative to the zeroth frequency,

$$\frac{A_n \cdot \sqrt{t_m} \cdot \sqrt{f_n}}{A_0 \cdot \sqrt{t_m} \cdot \sqrt{f_0}} = \mathcal{C}, \tag{17.13}$$

is some constant $\mathcal{C}$. We cancel the identical terms and solve for A_n, yielding

$$A_n = \mathcal{C} \cdot A_0 \cdot \sqrt{\frac{f_0}{f_n}}. \tag{17.14}$$

The amplitudes in our signal should therefor be proportional to

$$A_n \propto \sqrt{\frac{1}{f_n}}. \tag{17.15}$$

We assumed that the excitation signal (laser light) is directly proportional to the temperature signal. The relative ratio of amplitudes in the temperature signal should then be identical to the excitation signal. However, this is usually not the case. Higher frequencies are attenuated more than low frequencies in a lock-in thermography experiment.[25] The amplitude ratios should therefore be adapted to each specific measurement technique.

We can now freely choose the N phases of the individual sine waves. Any known and global phase shift can later be undone. This gives us more flexibility when we optimize certain properties of the signal. We use this freedom to reduce the offset S, and therefore the DC temperature rise as much as possible. We were unable to find an analytic expression that directly yields an optimal set of phase shifts. Therefore, we numerically explored the phase parameter space.

We found, that the optimal set of phase shifts is close to 0° for relative amplitudes described by equation (17.15). However, we did find sets of phases that are marginally *better* ($\ll 1\,\%$) than signals with all phases set to 0°. (see SI Figure 17.9 and SI Table 17.3) We also found, that the theoretical reduction in DC component is highest (50 %) for two sine waves and reduces to around 20 % for seven sine waves. Different amplitude ratios will most likely yield different results. Overall, numerical optimization of the phase sets is feasible and yields close to optimal signals.

17.3.4 Materials

We use Kapton foil (MT200+ from Goodfellow Ltd.) with a thickness of 25 µm as a reference material for the lock-in thermography measurements. We coated both sides with 20 nm of carbon by thermal evaporation. This increases the emissivity in the IR range and the absorption of the laser light.

17.4 Results and Discussion

17.4.1 Measuring the DC Temperature Offset

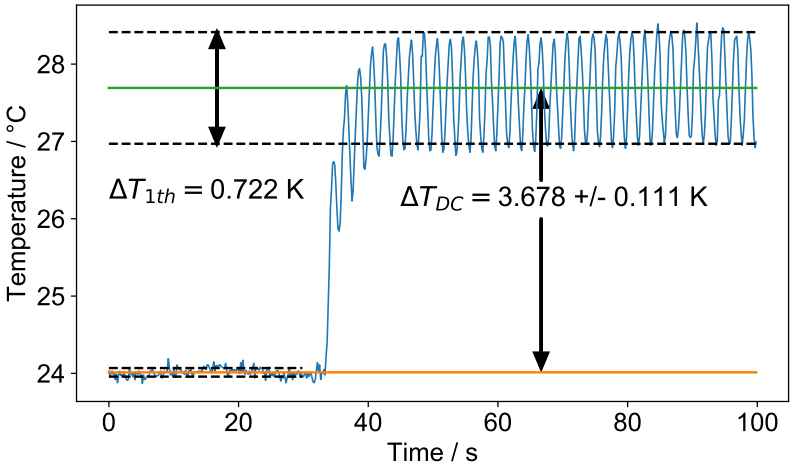


Fig. 17.3.: Surface temperature vs. time recorded with our IR camera near the heating center. I included the baseline (orange) to compute the true DC temperature offset. The line laser starts to periodically heat the sample at t = 34.8 s. Thermal equilibrium is then reached after less than 30 s. The DC temperature offset is the difference between the orange baseline and the average value during the oscillations (green line).

We measured the DC temperature rise during a lock-in thermography measurement with the procedure shown in Fig. 17.3. First, we recorded the baseline temperature. We then switched on the modulated laser and equilibrated the sample. The DC temperature rise is the difference between the baseline temperature and the average temperature after equilibration. In addition, we obtained the amplitude of the thermal oscillations by integrating the corresponding peak in the amplitude spectrum. Note that we can not use the DC component of the amplitude spectrum to get the DC temperature rise. The DC component in the amplitude spectrum represents the average temperature in the equilibrated state.

17.4.2 Less DC Heating with Square Waves

We now show how square wave excitations can be optimized beyond the performance of sine waves. The frequency of the excitation is usually determined by either the samples thermal properties or the measurement technique.

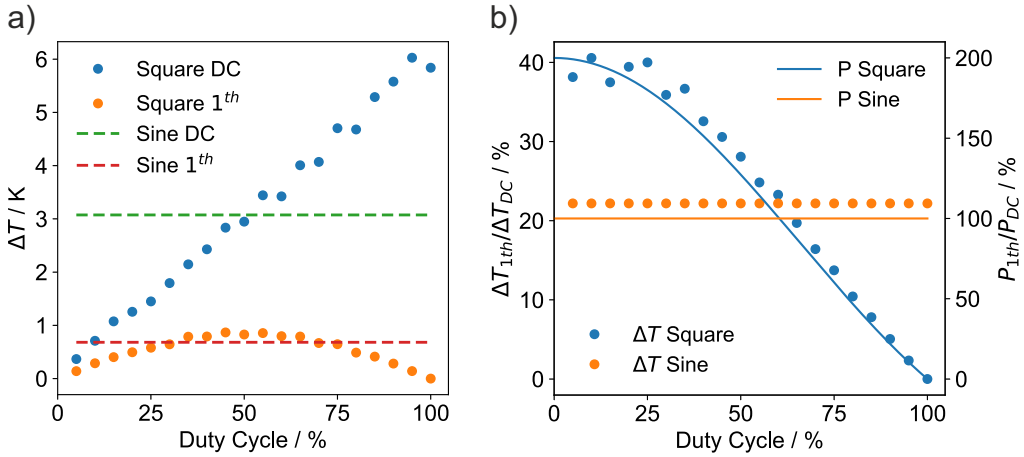


Fig. 17.4.: a) Temperature amplitudes of the DC component and 1st harmonic of a square wave excitation as a function of duty cycle. The dashed lines represent the corresponding values for a sine wave with equal amplitude and frequency. Note that a square wave with 50 % duty cycle creates temperature amplitudes similar to a sine wave. b) Ratio of the temperature amplitudes (dots) and signal power in the laser beam (lines) as a function of duty cycle. A square wave has a 50 % better ratio of the temperature amplitudes at lower duty cycles than a sine wave.

The duty cycle, D, of the square wave is the easiest parameter to adjust. Increasing the duty cycle beyond 50 % is counter productive. This only increases the DC heating and reduces the power in the first harmonic.

Reducing the duty cycle has two positive (P) and one negative (N) effect:

1. The thermal energy, deposited in the sample is reduced. The signal-to-noise ratio will therefore decrease. (N)
2. The DC temperature rise is reduced in accordance. (P)
3. The relative amplitudes of higher harmonics increase. (P)

Figure 17.4a shows the DC temperature rise, ΔT_{DC} and amplitude, ΔT_{1st} for a square signal as a function of duty cycle. A square wave with 50 % duty cycle creates similar ΔT_{DC} and ΔT_{1st} to a sine wave. A square wave with a duty cycle of less than 50 % creates less DC heating than a pure sine wave. Square waves with a duty

cycles of around 20-25 % are therefore superior to a sine wave. The DC temperature rise in our sample was reduced by ≈ 66 % from 3 to 1 K.

Figure 17.4b compares the ratios of the signal powers in the laser intensity (lines) and the temperature (dots). Square waves with 50 % duty cycle have power ratios similar to sine waves. Reducing the duty cycle to around 20-25 % almost doubles the power ratio for the first harmonic. This again shows that square waves with shorter duty cycle use the available signal power more efficiently.

17.4.3 Less DC Heating with Two Sine Waves

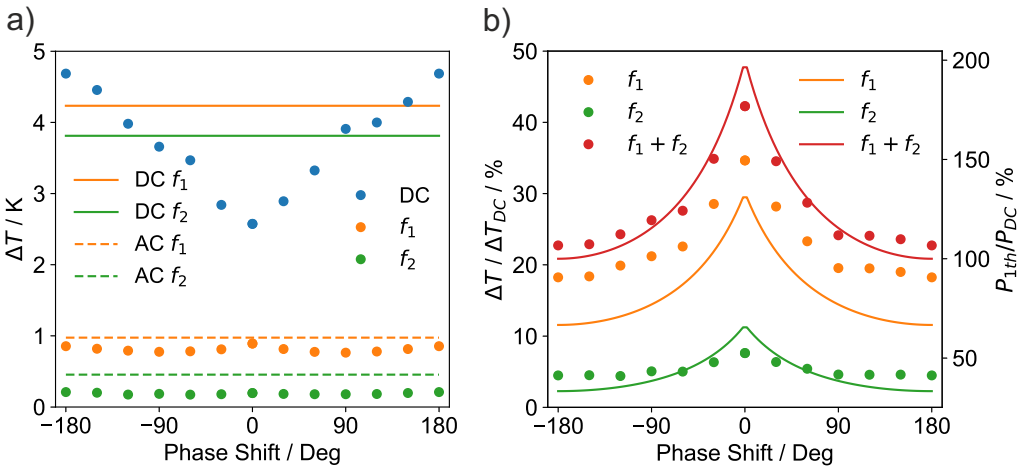


Fig. 17.5.: a) DC temperature rise and amplitudes (dots) of the 1st, and 2nd harmonic of the sum of two sine waves as a function of relative phase shift. The lines represent the corresponding values for individual sine waves with equal frequency. Note the sharp minimum in the DC heating at a phase shift of 0°. b) Ratio of the temperature amplitudes (dots) and signal power in the laser beam (lines) as a function of phase shift. The sum of two sine waves has a much better ratio of the temperature amplitudes at zero phase shift than the individual sine waves.

We now demonstrate the reduction in DC heating with two sine waves. The first sine wave has a frequency of $f_1 = 0.4\,\mathrm{Hz}$ and a relative amplitude of 1. The second sine wave has a frequency of $f_2 = 0.8\,\mathrm{Hz}$ and a relative amplitude of 0.5. We can freely adjust the relative phase shift, $\Delta\phi$, and obtain the following signal:

$$I(t) = \cos\left(2\pi\frac{0.4}{\mathrm{s}}\cdot t\right) + \frac{1}{2}\cdot\cos\left(2\pi\frac{0.8}{\mathrm{s}}\cdot t + \Delta\phi\right) \tag{17.16}$$

We then shift and rescale this signal to use the laser intensity optimally:

$$I_{\mathrm{rescaled}}(t) = I_0 \cdot \frac{I(t) - \min\{I(t)\}}{\max\{I(t)\} - \min\{I(t)\}}. \tag{17.17}$$

This creates an all positive intensity that fully uses the laser power, I_0. Rescaling only changes the absolute amplitude but not the relative ratios. The resulting signal for $\Delta\phi = 0$, the corresponding temperature signal from the sample, and the corresponding spectra are shown in the SI, Fig. 17.10.

Figure 17.5a shows the resulting DC temperature rise and the temperature amplitudes for the two sine waves. The amplitudes in the composite signal (dots) are very similar to the amplitudes from the individual sine waves (dashed lines). This means, that the magnitude of the individual amplitudes are preserved. In addition, they show no phase shift dependence. On the other hand, the DC temperature rise shows a pronounced minimum around a phase shift of 0°. Choosing the correct phase shift reduces the DC temperature rise by $\approx$ 46 %, from 4.69 to 2.57 K. Note, that the DC temperature rise is even smaller than for each individual sine wave (solid lines in Figure 17.5a).

Figure 17.5b compares the ratios of the 1st and 2nd harmonic to the DC component. The ratios for the laser intensity are lines. The corresponding ratios for the temperature signal are dots. An increasing proportion of the signal power is in the amplitudes of the two sine waves instead of the DC component. The overall signal power is therefore distributed more efficiently to the two spectral components we want.

17.4.4 Multi Frequency Lock-in Thermography

We now compare results from multi frequency lock-in thermography measurements. Our goal is to show that simultaneous measurements at multiple frequencies yield the same results as sequential measurements with pure sine waves. We use a 25 µm thick Kapton foil as the reference material. We compare three different excitation signals:

- single, sequentially measured, sine waves,
- a square wave with 20 % duty cycle, and
- the sum of four sine waves with minimal DC component.

Tab. 17.1.: Optimized parameters for multiplexed excitation signals. The amplitudes of the sine sum have inverse square root ratios. The amplitudes of the higher harmonics in the square signal are given by equation (17.3). The four individual sine waves have the same frequencies as the spectral components in these signals. All thermal decay lengths are small compared to the sample dimensions ($\approx$ 15 mm).

		Rel. Amplitude		
n	Frequency [Hz]	Sine Sum	Square ($D = 0.2$)	Thermal decay length [mm]
1	0.2	1	1	1.7
2	0.4	$1/\sqrt{2} \approx 0.707$	0.809	1.2
3	0.6	$1/\sqrt{3} \approx 0.577$	0.539	1
4	0.8	$1/\sqrt{4} = 0.5$	0.250	0.85

The frequencies and relative amplitudes in the square and sine sum signals are given in Table 17.1. We choose the frequency range such that the thermal oscillations decay long before they hit the sample boundaries ($\approx$ 15 mm). The thermal decay length, μ, for a given thermal diffusivity, α, at frequency, f, is given by[18]

$$\mu = \sqrt{\frac{\alpha}{\pi \cdot f}}. \tag{17.18}$$

Figure 17.6a shows the optimized signal, containing four sine waves that modulates the laser intensity with the samples surface temperature. Higher harmonics in the amplitudes spectrum (Figure 17.6b) are attenuated relative to the first harmonic. However, they are still well above the noise background.

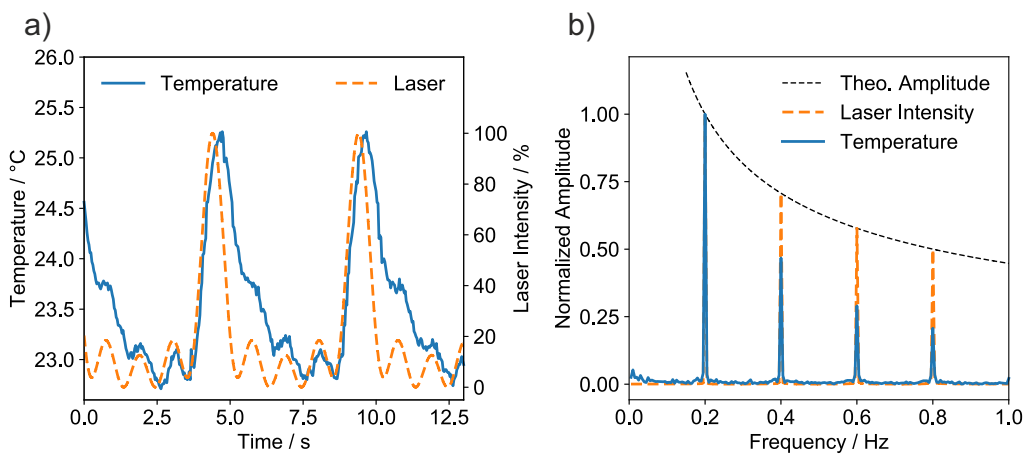


Fig. 17.6.: a) Surface temperature near the laser line and laser intensity as a function of time. The signal consists of four sine waves with minimal DC component. Note that the temperature does not follow the laser intensity. b) Amplitude spectrum of the surface temperature and laser power normalized to the 1st harmonic. Higher harmonics in the temperature are attenuated relative to the laser excitation.

We recorded 9000 frames at 6 frames per second after thermal equilibration for every signal. We then used the Goertzel algorithm[27] to efficiently extract the amplitude and phase images of individual spectral components. The frequencies are given in Table 17.1. Our implementation of the Goertzel algorithm in Python 3 is in the SI, section 17.8.1.

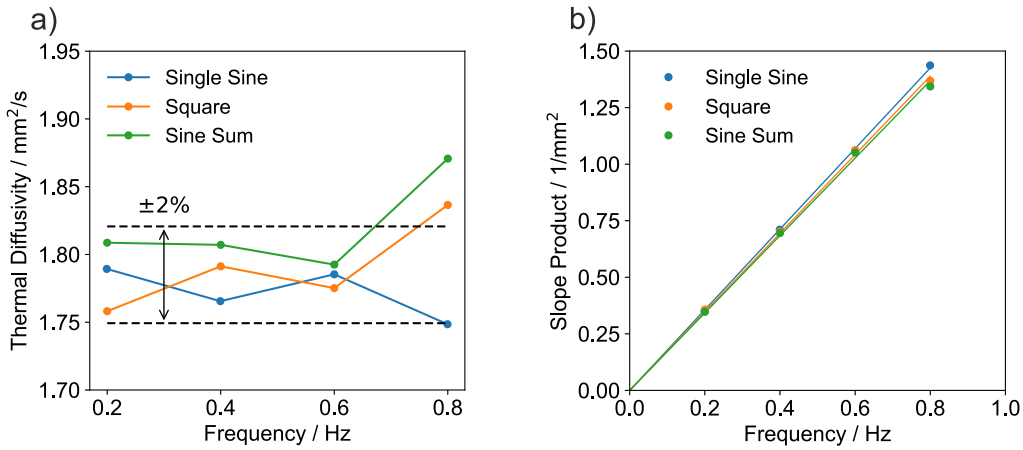


Fig. 17.7.: a) Thermal diffusivity vs. frequency computed with equation (17.19). All values lie within a small window of $\pm 2\,\%$. b) Product of the amplitude and phase slopes vs. frequency. The linear fit is forced through the origin. All three excitation signals create similar slope products.

Tab. 17.2.: Thermal diffusivity of Kapton (MT200+) measured with different excitation signals. The data from multiplexed measurements is almost identical to the result from a sequential measurement with individual sine waves.

Signal	Thermal diffusivity [$\mathrm{mm^2/s}$]
Single sine waves	1.76 ± 0.02
Square wave	1.81 ± 0.02
Sum of sine waves	1.84 ± 0.02

We analyzed the amplitude and phase images with the procedure from Wolf *et al.*[16] to obtain the thermal diffusivity. In essence, we compute the slopes of the linearized amplitude (m_A) and phase (m_ϕ) vs. distance from the laser line. The thermal diffusivity, α, at frequency, f is then given by

$$\alpha = \frac{\pi \cdot f}{m_\mathrm{A} \cdot m_\phi}. \tag{17.19}$$

Multi frequency measurements offer us two different analysis procedures: we can analyze each frequency individually (Eq. (17.19)), or we analyze all frequencies together. To do so, we perform a linear fit of the slope product $(m_\mathrm{A} \cdot m_\phi)$ vs. frequency. Solving equation 17.19 for the slope product yield

$$m_\mathrm{A} \cdot m_\phi = \frac{\pi \cdot f}{\alpha}. \tag{17.20}$$

We then take the derivative for the frequency:

$$\frac{\partial \left(m_\mathrm{A} \cdot m_\phi\right)}{\partial f} = \frac{\pi}{\alpha}. \tag{17.21}$$

This allows us to analyze measurements at multiple frequencies together.

Figure 17.7a shows the resulting thermal diffusivity for each individual frequency. All values are well within a $\pm 2\,\%$ error margin. The increased spread at 0.8 Hz is most likely caused by the short thermal decay length. Figure 17.7b shows the slope product vs. frequency for the three excitation signals. The resulting thermal diffusivities, computed with equation (17.21) are given in Table 17.2. Again, all values fall well within a $\pm 2\,\%$ error margin. We therefore conclude that all three signal yield essentially the same thermal diffusivity.

17.5 Conclusion

We showed, that a square wave with a duty cycle between 20 to 25 % creates 66 % less DC heating compared to a sine wave. In addition, the signal power in the first harmonic is doubled relative to a square wave with 50 % duty cycle. Square waves are therefore superior to sine waves. We also optimized signals with few spectral components by adjusting their relative phase shift. The optimal signals showed up to 46 % less DC heating. We finally compared lock-in thermography measurements conducted with individual sine waves, a square wave, and an optimized signal with four spectral components. All three signals produce almost identical values for the thermal diffusivity of Kapton. Applying multiplexed signals to techniques other than lock-in thermography requires careful consideration of the linearity of the involved physical effects. However, we do think that other techniques can benefit from both the reduction in DC heating and the simultaneous measurement at multiple frequencies. We therefore encourage others to develop similarly improved signals for any periodic heating method.

Acknowledgments

B.A.F.K. acknowledges support from the Elite Network of Bavaria. We thank Dr. Nelson Wilbur Pech-May for fruitful discussions.

17.6 References

[1] W. J. Parker, R. J. Jenkins, C. P. Butler, and G. L. Abbott, „Flash method of determining thermal diffusivity, heat capacity, and thermal conductivity," *Journal of Applied Physics*, p. 1679, 32 1961.

[2] A. Philipp, N. W. Pech-May, B. A. F. Kopera, *et al.*, „Direct measurement of the in-plane thermal diffusivity of semitransparent thin films by lock-in thermography: An extension of the slopes method," *Analytical Chemistry*, vol. 91, no. 13, pp. 8476–8483, 2019.

[3] I. Yang, D. Kim, and S. Lee, „Construction and preliminary testing of a guarded hot plate apparatus for thermal conductivity measurements at high temperatures," *International Journal of Heat and Mass Transfer*, vol. 122, pp. 1343–1352, 2018.

[4] D. Salmon, „Thermal conductivity of insulations using guarded hot plates, including recent developments and sources of reference materials," *Measurement Science and Technology*, vol. 12, no. 12, R89–R98, 2001.

[5] J. Kuhn, H.-P. Ebert, M. Arduini-Schuster, D. Büttner, and J. Fricke, „Thermal transport in polystyrene and polyurethane foam insulations," *International Journal of Heat and Mass Transfer*, vol. 35, no. 7, pp. 1795–1801, 1992.

[6] H. Hu, X. Wang, and X. Xu, „Generalized theory of the photoacoustic effect in a multilayer material," *Journal of Applied Physics*, vol. 86, no. 7, pp. 3953–3958, 1999.

[7] V. Singh, T. L. Bougher, A. Weathers, *et al.*, „High thermal conductivity of chain-oriented amorphous polythiophene," *Nature Nanotechnology*, vol. 9, no. 5, pp. 384–390, 2014.

[8] D. G. Cahill, „Thermal conductivity measurement from 30 to 750 k: The 3w method," *Review of Scientific Instruments*, pp. 802–808, 61 1990.

[9] C. Dames and G. Chen, „1w, 2w, and 3w methods for measurements of thermal properties," *Review of Scientific Instruments*, p. 124 902, 76 2005.

[10] S. Kommandur, A. Mahdavifar, P. J. Hesketh, and S. K. Yee, „3-omega technique using a microbridge heater for low power gas sensing," *Sensors and Actuators A: Physical*, pp. 213–238, 233 2015.

[11] S. D. Lubner, J. Choi, G. Wehmeyer, *et al.*, „Reusable bi-directional 3w sensor to measure thermal conductivity of 100-µm thick biological tissues," *Review of Scientific Instruments*, p. 014 905, 68 2015.

[12] W. Hodges and C. Dames, „A multi-frequency 3w method for tracking moving phase boundaries," *Review of Scientific Instruments*, vol. 90, no. 9, p. 094 903, 2019.

[13] A. J. Schmidt, R. Cheaito, and M. Chiesa, „A frequency-domain thermoreflectance method for the characterization of thermal properties," *Review of Scientific Instruments*, vol. 80, no. 9, p. 094 901, 2009.

[14] D. Rodin and S. K. Yee, „Simultaneous measurement of in-plane and through-plane thermal conductivity using beam-offset frequency domain thermoreflectance," *Review of Scientific Instruments*, vol. 88, no. 1, p. 014 902, 2017.

[15] L. Fabbri and P. Fenici, „Three-dimensional photothermal radiometry for the determination of the thermal diffusivity of solids," *Review of Scientific Instruments*, vol. 66, no. 6, pp. 3593–3600, 1995.

[16] A. Wolf, P. Pohl, and R. Brendel, „Thermophysical analysis of thin films by lock-in thermography," *Journal of Applied Physics*, vol. 96, no. 11, pp. 6306–6312, 2004.

[17] A. Muscio, P. G. Bison, S. Marinetti, and E. Grinzato, „Thermal diffusivity measurement in slabs using harmonic and one-dimensional propagation of thermal waves," *International Journal of Thermal Sciences*, vol. 43, no. 5, pp. 453–463, 2004.

[18] A. Mendioroz, R. Fuente-Dacal, E. Apiñaniz, and A. Salazar, „Thermal diffusivity measurements of thin plates and filaments using lock-in thermography," *Review of Scientific Instruments*, vol. 80, no. 7, p. 074 904, 2009.

[19] A. Salazar, A. Mendioroz, R. Fuente, and R. Celorrio, „Accurate measurements of the thermal diffusivity of thin filaments by lock-in thermography," *Journal of Applied Physics*, vol. 107, no. 4, p. 043 508, 2010.

[20] Y. S. Touloukian, *Thermophysical Properties of Matter*. New York: IFI/Plenum, 1973.

[21] N. W. Pech-May, A. Mendioroz, and A. Salazar, „Simultaneous measurement of the in-plane and in-depth thermal diffusivity of solids using pulsed infrared thermography with focused illumination," *NDT & E International*, vol. 77, pp. 28–34, 2016.

[22] B. Zhang and R. E. Imhof, „Theoretical analysis of the surface thermal wave technique for measuring the thermal diffusivity of thin slabs," *Applied Physics A Materials Science & Processing*, vol. 62, no. 4, pp. 323–334, 1996.

[23] C. Boué, F. Cassagne, C. Massoud, and D. Fournier, „Thermal imaging of a vein of the forearm: Analysis and thermal modelling,“ *Infrared Physics & Technology*, vol. 51, no. 1, pp. 13–20, 2007.

[24] E. A. Galapon and K. M. L. Martinez, „Exactification of the poincaré asymptotic expansion of the hankel integral: Spectacularly accurate asymptotic expansions and non-asymptotic scales,“ *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 470, no. 2162, p. 20 130 529, 2014.

[25] K. Chatterjee, S. Tuli, S. G. Pickering, and D. P. Almond, „A comparison of the pulsed, lock-in and frequency modulated thermography nondestructive evaluation techniques,“ *NDT & E International*, vol. 44, no. 7, pp. 655–667, 2011.

[26] R. Mulaveesala and S. Tuli, „Theory of frequency modulated thermal wave imaging for nondestructive subsurface defect detection,“ *Applied Physics Letters*, vol. 89, no. 19, p. 191 913, 2006.

[27] G. Goertzel, „An algorithm for the evaluation of finite trigonometric series,“ *The American Mathematical Monthly*, vol. 65, no. 1, pp. 34–35, 1958.

[28] J. W. Cooley and J. W. Tukey, „An algorithm for the machine calculation of complex fourier series,“ *Mathematics of Computation*, vol. 19, no. 90, pp. 297–297, 1965.

[29] C. Shannon, „Communication in the presence of noise,“ *Proceedings of the IEEE*, vol. 86, no. 2, pp. 447–457, 1998.

17.7 Supporting Information

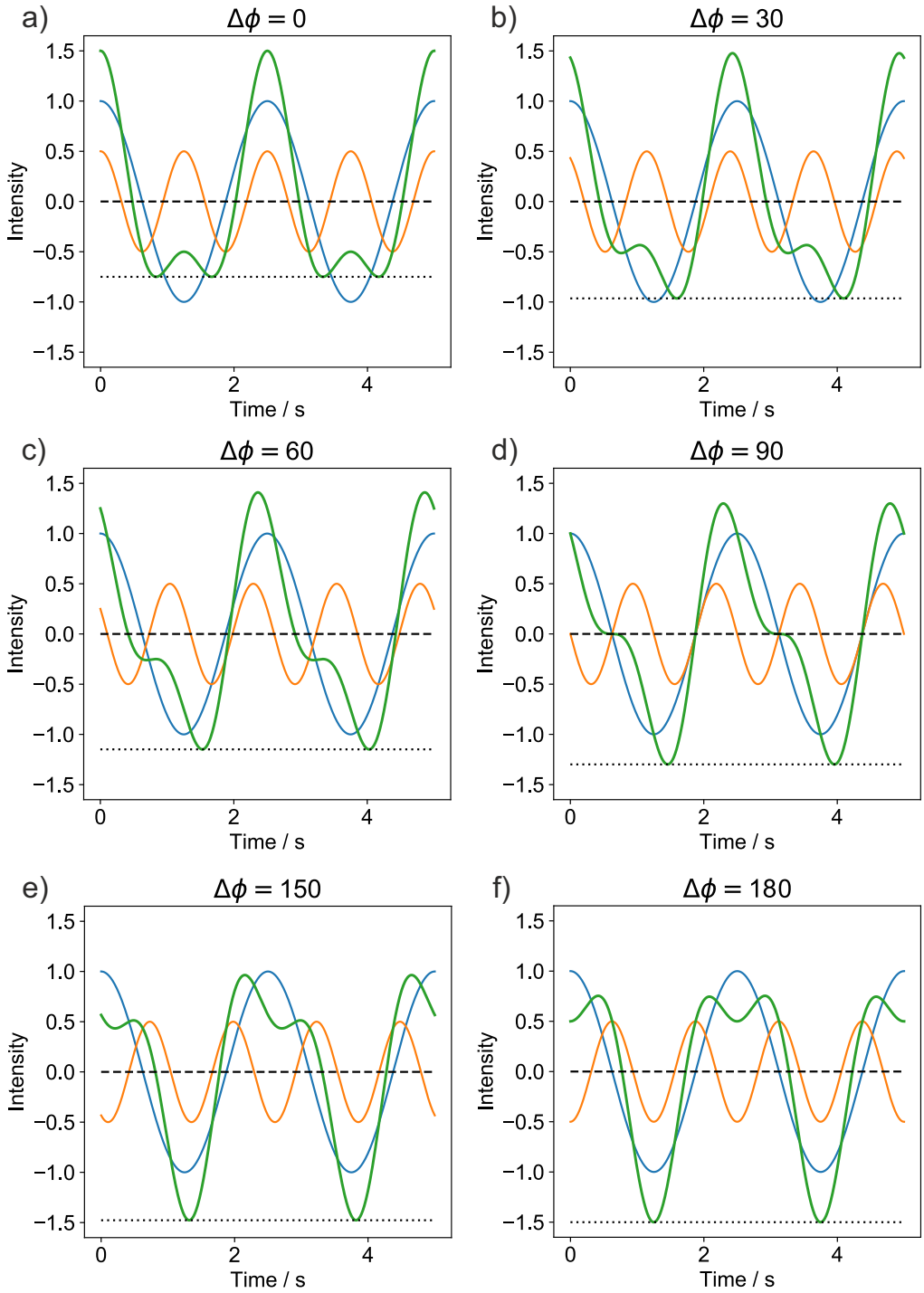


Fig. 17.8.: Interference signal (green) created by the summation of two, phase shifted sine waves (blue and orange). The DC component of all signals shown here is zero. However, they are also partially negative. Light intensities can only be modulated with fully positive signals. Therefore, we have to add a DC component to the resulting signal by shifting it in the positive y-direction. We want this shift to be as small as possible. The shift is smallest for a phase difference of $\Delta\phi = 0$ and largest for a phase difference of $\Delta\phi = 180°$. Note that this difference is created by the interference of the individual spectral components.

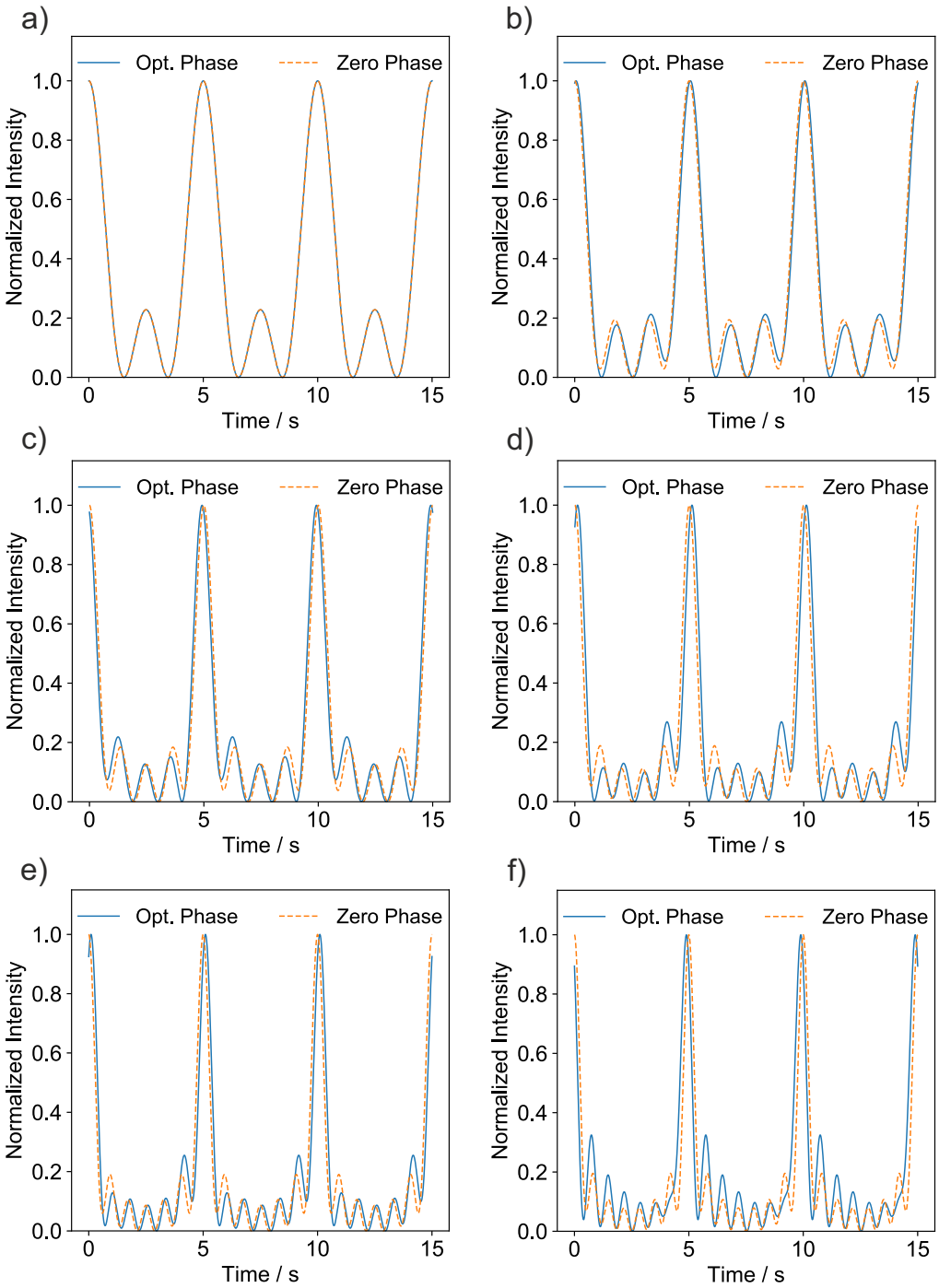


Fig. 17.9.: Comparison between multiplexed signals with numerically optimized phases (blue lines) and a phase shift of zero (orange dashed). The number of sine waves is a) 2, b) 3, c) 4, d) 5, e) 6, and f) 7. Both signals are very similar. Their DC components differ by less than 2 %.

Tab. 17.3.: DC reduction with multiple sine waves. We compare the relative reduction in the DC component ($\Delta S\,\%$), with increasing number of sine waves, N to the worst case DC component. The signal with an numerically optimized set of phases performs very similar to a signal with zero phase shift. The $\approx 2\,\%$ difference for $N = 7$ is most likely caused by the large, high dimensional parameter space. Overall, multiplexing with an arbitrary signal works best with few (≈ 5) spectral components.

N	Rel. Difference [%]	$\Delta S\,\%$ Opt. Phases	$\Delta S\,\%$ Zero Phase
2	0.0002	51.77	51.77
3	0.1257	37.96	38.09
4	0.0302	29.59	29.62
5	0.0761	25.22	25.29
6	0.0212	21.79	21.77
7	2.3663	21.96	19.59

17.7.1 Temperature signal from two sine waves

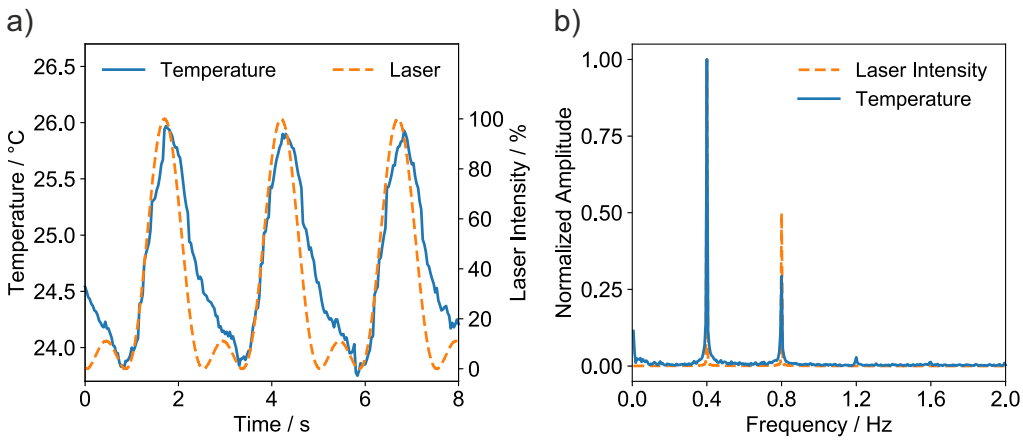


Fig. 17.10.: a) Surface temperature near the laser line and laser intensity as a function of time. Note that the temperature does not follow the laser intensity. b) Amplitude spectrum of the surface temperature and laser power normalized to the 1st harmonic. Higher harmonics in the temperature are attenuated relative to the laser excitation.

17.7.2 Fourier Series of a Square Wave

We now derive the Fourier series representation of a square wave. The square wave we us is shown in Figure 17.11.

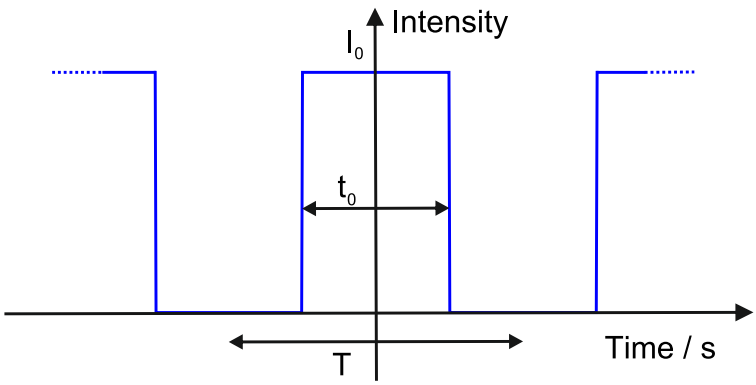


Fig. 17.11.: Square wave centered around the origin with period T and on-time t_0.

This square wave has period T and on-time t_0. The duty cycle, D, is therefore given by t_0/T. We call the intensity in the on-state I_0. Overall, our signal is defined by

$$I(t) = \begin{cases} I_0, & t \in [-t_0/2 + k \cdot T; t_0/2 + k \cdot T] \quad k \text{ in } \mathbb{Z} \\ 0, & \text{else} \end{cases} \tag{17.22}$$

Fourier discovered,[1] that every periodic signal, $I(t)$, can be represented by an infinite sum of sine and cosine waves like

$$I(t) = A_0 + \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{2\pi nt}{T}\right) + B_n \cdot \sin\left(\frac{2\pi nt}{T}\right) \tag{17.23}$$

Our task is now to find values for the coefficients A_n and B_n. Note that the square wave in Figure 17.11 has two special properties:

1. the signal is symmetric around the y-axis, (even function)
2. the signal is always greater or equal to 0. (always positive)

Even functions only have cosine components in their Fourier series because sine waves are uneven. We conclude that all B_n coefficients are 0. This leaves us with

$$I(t) = A_0 + \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{2\pi nt}{T}\right) \tag{17.24}$$

as our Fourier series.

Functions which are always positive have a non-zero DC component (A_0). The DC component represents the average value of the function. We compute the average by

$$A_0 = \frac{1}{T} \int_{-T/2}^{T/2} I(t)\,\mathrm{d}t. \tag{17.25}$$

The square wave is zero outside of $[-t_0/2; t_0/2]$. This allows us to limit the integration to this interval like

$$A_0 = \frac{1}{T} \int_{-T/2}^{T/2} I(t)\,\mathrm{d}t = \frac{1}{T} \int_{-t_0/2}^{t_0/2} I(t)\,\mathrm{d}t. \tag{17.26}$$

The intensity is constant inside this interval, which leaves us with

$$A_0 = \frac{1}{T} \int_{-t_0/2}^{t_0/2} I_0\,\mathrm{d}t. \tag{17.27}$$

Next, we factor out the constant intensity I_0 and solve the integral:

$$A_0 = \frac{I_0}{T} \int_{-t_0/2}^{t_0/2} 1\,\mathrm{d}t = \frac{I_0}{T} \left[t\right]_{-t_0/2}^{t_0/2} = \frac{I_0}{T} \left[\frac{t_0}{2} - \left(-\frac{t_0}{2}\right)\right] = I_0 \cdot \frac{t_0}{T} \tag{17.28}$$

Note that the DC component, A_0, is directly related to the duty cycle, D, by

$$\boxed{A_0 = I_0 \cdot \frac{t_0}{T} = I_0 \cdot D}. \tag{17.29}$$

Finally, we have to determine the amplitudes A_n. We showed in section 17.7.3 that the amplitudes are given by

$$A_n = \frac{2}{T} \int_0^T I(t) \cdot \cos\left(\frac{2\pi n t}{T}\right)\,\mathrm{d}t \tag{17.30}$$

Again, we limit the integration to the non-zero part of our square wave

$$A_n = \frac{2}{T} \int_{-t_0/2}^{t_0/2} I_0 \cdot \cos\left(\frac{2\pi n t}{T}\right)\,\mathrm{d}t \tag{17.31}$$

and use the anti-derivative:

$$A_n = \frac{2}{T} \cdot I_0 \cdot \left[\frac{\sin\left(\frac{2\pi n t}{T}\right)}{\frac{2\pi n}{T}}\right]_{-t_0/2}^{t_0/2}. \tag{17.32}$$

Inserting the upper and lower bounds gives us

$$A_n = \frac{2}{T} \cdot I_0 \cdot \left[\frac{\sin\left(\frac{2\pi n t_0/2}{T}\right)}{\frac{2\pi n}{T}} - \frac{\sin\left(\frac{2\pi n(-t_0/2)}{T}\right)}{\frac{2\pi n}{T}} \right]. \tag{17.33}$$

We simplify the fractions by canceling T and the factor 2 in the sine to get

$$A_n = \frac{I_0}{\pi n} \cdot \left[\sin\left(\frac{\pi n t_0}{T}\right) - \sin\left(\frac{-\pi n t_0}{T}\right) \right]. \tag{17.34}$$

We note that the sine function is even and apply the identity,

$$-\sin(-x) = \sin(x), \tag{17.35}$$

to get

$$A_n = \frac{I_0}{\pi n} \cdot \left[\sin\left(\frac{\pi n t_0}{T}\right) + \sin\left(\frac{\pi n t_0}{T}\right) \right] = \frac{I_0}{\pi n} \cdot 2 \cdot \sin\left(\frac{\pi n t_0}{T}\right) \tag{17.36}$$

Finally, we introduce the duty cycle ($D = {}^{t_0}/_{T}$):

$$\boxed{A_n = 2 \cdot I_0 \cdot \frac{\sin(\pi n D)}{\pi n}}. \tag{17.37}$$

The Fourier series for our square wave is therefore

$$I(t) = I_0 \cdot D + \sum_{n=1}^{\infty} 2 \cdot I_0 \cdot \frac{\sin(\pi n D)}{\pi n} \cdot \cos\left(\frac{2\pi n t}{T}\right). \tag{17.38}$$

We simplify by factoring out the intensity I_0 and get

$$\boxed{I(t) = I_0 \cdot \left[D + \sum_{n=1}^{\infty} 2 \cdot \frac{\sin(\pi n D)}{\pi n} \cdot \cos\left(\frac{2\pi n t}{T}\right) \right]}. \tag{17.39}$$

17.7.3 Fourier Series of Periodic Signals

In this section, we show how the amplitudes in a Fourier series are computed. We look for the amplitude of the k-th harmonic, A_k. The general procedure works like this: We start by multiplying both sides of the the Fourier Series (Eq. (17.24))

$$I(t) = A_0 + \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{2\pi nt}{T}\right) \tag{17.40}$$

with the cosine of the k-th harmonic to get

$$I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) = \left[A_0 + \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{2\pi nt}{T}\right)\right] \cdot \cos\left(\frac{2\pi kt}{T}\right). \tag{17.41}$$

We expand the square brackets into

$$I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) = A_0 \cdot \cos\left(\frac{2\pi 2t}{T}\right) + \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{2\pi nt}{T}\right) \cdot \cos\left(\frac{2\pi kt}{T}\right). \tag{17.42}$$

We then integrate both sides over one period, T, like

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = \int_0^T A_0 \cdot \cos\left(\frac{2\pi 2t}{T}\right) + \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{2\pi nt}{T}\right) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t. \tag{17.43}$$

We split the integral on the right at the plus sign into two:

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = \int_0^T A_0 \cdot \cos\left(\frac{2\pi 2t}{T}\right) \mathrm{d}t + \int_0^T \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{2\pi nt}{T}\right) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t. \tag{17.44}$$

The term with the DC component, A_0, is zero because we integrate a cosine over a full period. The positive and negative half waves cancel each other out and we get

$$\int_0^T A_0 \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = 0. \tag{17.45}$$

This simplifies equation (17.44) to

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = \int_0^T \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{2\pi nt}{T}\right) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t. \tag{17.46}$$

We assume that the infinite sum is convergent. This allows us to exchange the sum with the definite integral into

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = \sum_{n=1}^{\infty} A_n \cdot \int_0^T \cos\left(\frac{2\pi nt}{T}\right) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t. \tag{17.47}$$

The positive and negative parts of the product of two cosines is equal if they have different frequencies. All integrals on the right side are therefore zero, except in the case where $n = k$. We can replace n with k and obtain

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = A_k \cdot \int_0^T \cos\left(\frac{2\pi kt}{T}\right) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t. \tag{17.48}$$

We then simplify the integrand,

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = A_k \cdot \int_0^T \left[\cos\left(\frac{2\pi kt}{T}\right)\right]^2 \mathrm{d}t, \tag{17.49}$$

and solve the integral with the anti-derivative of

$$\int \left[\cos\left(\frac{2\pi kt}{T}\right)\right]^2 \mathrm{d}t = \frac{t}{2} + \frac{T \sin\left(\frac{4\pi kt}{T}\right)}{8\pi k} + \mathcal{C} \tag{17.50}$$

to get

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = A_k \cdot \left[\frac{t}{2} + \frac{T \sin\left(\frac{4\pi kt}{T}\right)}{8\pi k}\right]_0^T. \tag{17.51}$$

Inserting the upper and lower bounds gives us

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = A_k \cdot \left[\frac{T}{2} + \frac{T \sin\left(\frac{4\pi kT}{T}\right)}{8\pi k} - \frac{0}{2} - \frac{T \sin\left(\frac{4\pi k0}{T}\right)}{8\pi k}\right]. \tag{17.52}$$

We simplify the terms in the square bracket,

$$\int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t = A_k \cdot \frac{T}{2}, \tag{17.53}$$

and solve for A_k, leaving us with

$$\boxed{A_k = \frac{2}{T} \int_0^T I(t) \cdot \cos\left(\frac{2\pi kt}{T}\right) \mathrm{d}t}. \tag{17.54}$$

17.8 References

[1] E. M. Stein, *Fourier Analysis: An Introduction (Princeton Lectures in Analysis, Volume 1)*. Princeton University Press, 2003.

17.8.1 Goertzel Algorithm Implementation

```python
from logging import getLogger
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import time
from pathlib import Path

class ImageRTGoertzel():
    """
    Defines a 2D Goertzel filter for an image sequence.
    Each pixel is processed individualy.
    """
    def __init__(self, FreqBin, fps, ImageDim, NSamples, FreqID, ProjectName):
        """
        Constructor for the 2D Goertzel filter.

        Parameters
        ----------
        FreqBin : int
            The frequency bin for this filter
        fps : int
            Frames per second of the IR movie
        ImageDim : array of int
            Dimension of the image frames [x, y]
        NSamples : int
            Number of frames, determines the frequency
            resolution
        FreqID : int
            ID to separate the data file names
        """
        self.FreqBin = FreqBin
        self.ProjectName = ProjectName
        self.FPS = fps
        self.FreqID = FreqID
        self.NSamples = NSamples
        self.Freq = FreqBin/NSamples * fps
        self.ImageDim = ImageDim
        self.FrameCount = 0
        self.Omega = 2 * np.pi * FreqBin/NSamples
        self.AmplImage = np.zeros(ImageDim)
        self.PhaseImage = np.zeros(ImageDim)
        self.Sentinel = getLogger("Watchtower")
        # goertzel algorithm variables
        self.z0 = np.zeros(ImageDim)
        self.z1 = np.zeros(ImageDim)
        self.z2 = np.zeros(ImageDim)
        self.c = 2*np.cos(self.Omega)
        self.ResultPath = Path() / "DFTData"
        self.ResultPath.mkdir(parents = True, exist_ok = True)
        self.PlotPath = Path() / "Plots" / "Raw Demultiplexed"
        self.PlotPath.mkdir(parents = True, exist_ok = True)
        self.Sentinel.info("Goertzel Structure at frequency:"+\
                           " {:f} created!".format(self.Freq))

    def AddFrame(self, NewFrame):
        """
        Adds a new frame to the 2D Goertzel filter.

        Parameters
        ----------
        NewFrame : 2D numpy array
            New thermogram for the next time step
        """
        self.FrameCount += 1
        self.z0 = NewFrame + self.c * self.z1 - self.z2
```

```
66          self.z2 = self.z1
67          self.z1 = self.z0
68
69          return True
70
71      def getFrameCount(self):
72          return self.FrameCount
73
74      def getRealPart(self):
75          """
76          Returns the current real part
77          """
78          return (np.cos(self.Omega) * self.z1 - self.z2)/(self.FrameCount/2)
79
80      def getImagPart(self):
81          """
82          Returns the current imaginary part
83          """
84          return (np.sin(self.Omega) * self.z1)/(self.FrameCount/2)
85
86      def getAmplImage(self):
87          """
88          Creates the current amplitude image
89          """
90          XKreal = self.getRealPart()
91          XKimag = self.getImagPart()
92          self.AmplImage = np.abs(XKreal + 1j*XKimag)
93          self.Sentinel.info("Amplitude image at frame {}".format(self.FrameCount)+\
94                             " for frequency {} Hz created.".format(self.Freq))
95          return self.AmplImage
96
97      def getPhaseImage(self):
98          """
99          Creates the current phase image
100         """
101         XKreal = self.getRealPart()
102         XKimag = self.getImagPart()
103         self.PhaseImage = np.angle(XKreal + 1j*XKimag, deg = True)
104         self.Sentinel.info("Phase image at frame {}".format(self.FrameCount)+\
105                            " for frequency {} Hz created.".format(self.Freq))
106         return self.PhaseImage
107
108     def saveAmplData(self):
109         """
110         Saves the extracted amplitude image to an ASCII file
111         for further processing. The ASCII file is similar to the
112         file created by the IRBIS software.
113         """
114         # get the current Amplitude Image
115         AmplData = self.getAmplImage()
116         AmplPath = self.ResultPath /\
117                    "{}_DFTConv_{!s}_Amp.txt".format(self.ProjectName, self.FreqID)
118         ResultFile = open(str(AmplPath.absolute()), "w")
119         ResultFile.write("Amplitude for each Pixel of Project: {}\n".format(self.ProjectName))
120         ResultFile.write("Data processed on :{}\n".format(time.ctime(time.time())))
121         ResultFile.write("LIT.LockInFrequency[Hz]={:f}\n".format(self.Freq))
122         ResultFile.write("[Data] Amplitude in [K]\n")
123         for line in range(len(AmplData)-1, -1, -1):  # invert y-axis
124             for row in range(0, len(AmplData[0]), 1):
125                 ResultFile.write(str(round(AmplData[line][row], 5)) + ";")
126             ResultFile.write("\n")
127         ResultFile.close()
128         self.Sentinel.info("Amplitude Data for Frequency:"+\
129                            " {:f} Hz saved!".format(self.Freq))
130         return True
131
132     def savePhaseData(self):
133         """
```

```
        Saves the extracted phase image to an ASCII file
        for further processing. The ASCII file is similar to the
        file created by the IRBIS software.
        """
        # get the current phase image
        PhaseData = self.getPhaseImage()
        PhasePath = self.ResultPath /\
                    "{}_DFTConv_{!s}_Phase.txt".format(self.ProjectName, self.FreqID)
        ResultFile = open(str(PhasePath.absolute()), "w")
        ResultFile.write("Phase for each Pixel of Project: {}\n".format(self.ProjectName))
        ResultFile.write("Data processed on :{}\n".format(time.ctime(time.time())))
        ResultFile.write("LIT.LockInFrequency[Hz]={:f}\n".format(self.Freq))
        ResultFile.write("[Data] Phase in [Deg]\n")
        for line in range(len(PhaseData)-1, -1, -1):   # invert y-axis
            for row in range(0, len(PhaseData[0]), 1):
                ResultFile.write(str(round(PhaseData[line][row], 5)) + ";")
            ResultFile.write("\n")

        ResultFile.close()
        self.Sentinel.info("Phase Data for Frequency:"+\
                           " {:f} Hz saved!".format(self.Freq))

        return True

    def plotAmplImage(self):
        # get the current Amplitude Image
        AmplData = self.getAmplImage()
        # plot the amplitude image
        fig = plt.figure()
        ax = fig.add_subplot(111, aspect = "equal")
        ExtLimits = [0, len(AmplData[1]) ,\
                     0, len(AmplData)]
        img = plt.imshow(AmplData, cmap = 'viridis',\
                         interpolation = 'none',\
                         origin = 'upper',\
                         extent = ExtLimits,\
                         norm = LogNorm())
        plt.colorbar(label = "Amplitude / K")
        ax.set_xlabel("Distance x / Pixel")
        ax.set_ylabel("Distance y / Pixel")
        ax.set_title("Amplitude, f = {:f} Hz".format(self.Freq))
        plt.tight_layout()
        FigPath = self.PlotPath / (self.ProjectName + "_Ampl_" +\
                            str(self.FreqID) + ".png")
        plt.savefig(str(FigPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Amplitude Image for Frequency:"+\
                           " {:f} Hz plotted!".format(self.Freq))

        return True

    def plotPhaseImage(self):
        # get the current Amplitude Image
        PhaseData = self.getPhaseImage()
        # plot the amplitude image
        fig = plt.figure()
        ax = fig.add_subplot(111, aspect = "equal")
        ExtLimits = [0, len(PhaseData[1]),\
                     0, len(PhaseData)]
        img = plt.imshow(PhaseData, cmap = 'inferno', interpolation = 'none',\
                         origin = 'upper', extent = ExtLimits)
        plt.colorbar(label = "Phase / Degree")
        ax.set_xlabel("Distance x / Pixel")
        ax.set_ylabel("Distance y / Pixel")
        ax.set_title("Phase, f = {:f} Hz".format(self.Freq))
        plt.tight_layout()
        FigPath = self.PlotPath / (self.ProjectName + "_Phase_" +\
                                   str(self.FreqID) + ".png")
```

```
        plt.savefig(str(FigPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Phase Image for Frequency:"+\
                           " {:f} Hz plotted!".format(self.Freq))

        return True

    def __str__(self):
        RString = "Goertzel Image Filter Object:\n"
        RString += "DFT Frequency: {:f} Hz\n".format(self.Freq)
        RString += "Sampling rate (fps): {:f} Hz\n".format(self.fps)
        RString += "Image Dimensions: {:s} Pixel\n".format(self.ImageDim)
        RString += "Current State:\n"
        RString += "Frame Count: {:s}\n".format(self.FrameCount)

        return RString
```

```
from logging import getLogger
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import time
from pathlib import Path
from LIT_ImageRTGoertzel import ImageRTGoertzel

class ImageGoertzelPeak():
    """
    Uses a 2D Goertzel filter to extract the amplitude and phase
    of peaks with finite width.
    """
    def __init__(self, FreqBin, fps, ImageDim, NSamples,\
                 FreqID, FreqWidth, ProjectName):
        """
        Constructor for the 2D Goertzel filter.

        Parameters
        ----------
        FreqBin : int
            The frequency bin for this filter
        fps : int
            Frames per second of the IR movie
        ImageDim : array of int
            Dimension of the image frames [x, y]
        NSamples : int
            Number of frames, determines the frequency
            resolution
        FreqWidth : int (uneven)
            Number of neighboring frequency bins
        FreqID : int
            ID to separate the data file names
        """
        self.FreqBin = FreqBin
        self.ProjectName = ProjectName
        self.FPS = fps
        self.FreqID = FreqID
        self.NSamples = NSamples
        self.Freq = FreqBin/NSamples * fps
        self.ImageDim = ImageDim
        self.FrameCount = 0
        self.Omega = 2 * np.pi * FreqBin/NSamples
        self.AmplImage = np.zeros(ImageDim)
        self.PhaseImage = np.zeros(ImageDim)
        self.Sentinel = getLogger("Watchtower")

        # create a list with one G-Filter for each frequency bin
        self.GList = []
        for k in range(FreqBin - int(FreqWidth/2),\
```

```
                        FreqBin + int(FreqWidth/2)+1, 1):
            self.GList.append(ImageRTGoertzel(k, fps, ImageDim,\
                                              NSamples, k, ProjectName))

        self.ResultPath = Path() / "DFTData"
        self.ResultPath.mkdir(parents = True, exist_ok = True)
        self.PlotPath = Path() / "Plots" / "Raw Demultiplexed"
        self.PlotPath.mkdir(parents = True, exist_ok = True)

        self.Sentinel.info("Finite peak width Goertzel"+\
                           " Structure at frequency:"+\
                           " {:f} created!".format(self.Freq))

    def AddFrame(self, NewFrame):
        """
        Adds a new frame to the 2D Goertzel filter.

        Parameters
        ----------
        NewFrame : 2D numpy array
            New thermogram for the next time step
        """
        for k in range(0, len(self.GList), 1):
            self.GList[k].AddFrame(NewFrame)

        return True

    def getFrameCount(self):
        return self.FrameCount

    def getRealPart(self):
        """
        Returns the current real part
        """
        RealImage = self.GList[0].getRealPart()
        for k in range(1, len(self.GList), 1):
            RealImage += self.GList[k].getRealPart()

        return RealImage

    def getImagPart(self):
        """
        Returns the current imaginary part
        """
        ImagImage = self.GList[0].getImagPart()
        for k in range(1, len(self.GList), 1):
            ImagImage += self.GList[k].getImagPart()

        return ImagImage

    def getAmplImage(self):
        """
        Creates the current amplitude image
        by summing all the amplitudes for
        every frequency bin.
        """
        self.AmplImage = self.GList[0].getAmplImage()
        for k in range(1, len(self.GList), 1):
            self.AmplImage += self.GList[k].getAmplImage()
        self.Sentinel.info("Amplitude image at frame {}".format(self.FrameCount)+\
                           " for frequency {} Hz created.".format(self.Freq))
        return self.AmplImage

    def getPhaseImage(self):
        """
        Creates the current phase image
        by averaging the phase for
        every frequency bin.
```

```
        """
        self.PhaseImage = self.GList[0].getPhaseImage()
        for k in range(1, len(self.GList), 1):
            self.PhaseImage += self.GList[k].getPhaseImage()
        self.PhaseImage = self.PhaseImage / len(self.GList)
        self.Sentinel.info("Phase image at frame {}".format(self.FrameCount)+\
                           " for frequency {} Hz created.".format(self.Freq))
        return self.PhaseImage

    def saveAmplData(self):
        """
        Saves the extracted amplitude image to an ASCII file
        for further processing. The ASCII file is similar to the
        file created by the IRBIS software.
        """
        # get the current Amplitude Image
        AmplData = self.getAmplImage()
        AmplPath = self.ResultPath /\
                   "{}_DFTConv_{!s}_Amp.txt".format(self.ProjectName, self.FreqID)
        ResultFile = open(str(AmplPath.absolute()), "w")
        ResultFile.write("Amplitude for each Pixel of Project: {}\n".format(self.ProjectName))
        ResultFile.write("Data processed on :{}\n".format(time.ctime(time.time())))
        ResultFile.write("LIT.LockInFrequency[Hz]={:f}\n".format(self.Freq))
        ResultFile.write("[Data] Amplitude in [K]\n")
        for line in range(len(AmplData)-1, -1, -1):  # invert y-axis
            for row in range(0, len(AmplData[0]), 1):
                ResultFile.write(str(round(AmplData[line][row], 5)) + ";")
            ResultFile.write("\n")

        ResultFile.close()
        self.Sentinel.info("Amplitude Data for Frequency:"+\
                           " {:f} Hz saved!".format(self.Freq))
        return True

    def savePhaseData(self):
        """
        Saves the extracted phase image to an ASCII file
        for further processing. The ASCII file is similar to the
        file created by the IRBIS software.
        """
        # get the current phase image
        PhaseData = self.getPhaseImage()
        PhasePath = self.ResultPath /\
                    "{}_DFTConv_{!s}_Phase.txt".format(self.ProjectName, self.FreqID)
        ResultFile = open(str(PhasePath.absolute()), "w")
        ResultFile.write("Phase for each Pixel of Project: {}\n".format(self.ProjectName))
        ResultFile.write("Data processed on :{}\n".format(time.ctime(time.time())))
        ResultFile.write("LIT.LockInFrequency[Hz]={:f}\n".format(self.Freq))
        ResultFile.write("[Data] Phase in [Deg]\n")
        for line in range(len(PhaseData)-1, -1, -1):   # invert y-axis
            for row in range(0, len(PhaseData[0]), 1):
                ResultFile.write(str(round(PhaseData[line][row], 5)) + ";")
            ResultFile.write("\n")
        ResultFile.close()
        self.Sentinel.info("Phase Data for Frequency:"+\
                           " {:f} Hz saved!".format(self.Freq))

        return True

    def plotAmplImage(self):
        # plot all the individual amplitude images
        for GFilter in self.GList:
            GFilter.plotAmplImage()
        # get the current combined Amplitude Image
        AmplData = self.getAmplImage()
        # plot the amplitude image
        fig = plt.figure()
        ax = fig.add_subplot(111, aspect = "equal")
```

```
        ExtLimits = [0, len(AmplData[1]) ,\
                     0, len(AmplData)]
        img = plt.imshow(AmplData, cmap = 'viridis',\
                         interpolation = 'none',\
                         origin = 'upper',\
                         extent = ExtLimits,\
                         norm = LogNorm())
        plt.colorbar(label = "Amplitude / K")
        ax.set_xlabel("Distance x / Pixel")
        ax.set_ylabel("Distance y / Pixel")
        ax.set_title("Amplitude, f = {:f} Hz".format(self.Freq))
        plt.tight_layout()
        FigPath = self.PlotPath / (self.ProjectName + "_Ampl_" +\
                            str(self.FreqID) + "_Comb.png")
        plt.savefig(str(FigPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Amplitude Image for Frequency:"+\
                           " {:f} Hz plotted!".format(self.Freq))

        return True

    def plotPhaseImage(self):
        # plot all the individual phase images
        for GFilter in self.GList:
            GFilter.plotPhaseImage()
        # get the current Amplitude Image
        PhaseData = self.getPhaseImage()
        # plot the amplitude image
        fig = plt.figure()
        ax = fig.add_subplot(111, aspect = "equal")
        ExtLimits = [0, len(PhaseData[1]),\
                     0, len(PhaseData)]
        img = plt.imshow(PhaseData, cmap = 'inferno', interpolation = 'none',\
                         origin = 'upper', extent = ExtLimits)
        plt.colorbar(label = "Phase / Degree")
        ax.set_xlabel("Distance x / Pixel")
        ax.set_ylabel("Distance y / Pixel")
        ax.set_title("Phase, f = {:f} Hz".format(self.Freq))
        plt.tight_layout()
        FigPath = self.PlotPath / (self.ProjectName + "_Phase_" +\
                                   str(self.FreqID) + "Comb_.png")
        plt.savefig(str(FigPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Phase Image for Frequency:"+\
                           " {:f} Hz plotted!".format(self.Freq))

        return True

    def __str__(self):

        RString = "Goertzel Image Filter Object:\n"
        RString += "DFT Frequency: {:f} Hz\n".format(self.Freq)
        RString += "Sampling rate (fps): {:f} Hz\n".format(self.fps)
        RString += "Image Dimensions: {:s} Pixel\n".format(self.ImageDim)
        RString += "Current State:\n"
        RString += "Frame Count: {:s}\n".format(self.FrameCount)

        return RString
```

```
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from logging import getLogger
import numpy as np
import time
from os import getcwd, listdir
from os.path import join
from LIT_ImageGoertzelPeak import ImageGoertzelPeak
```

```python


class Demultiplexer:
    """
    Demultiplexer to extract multiple spectral components from
    a sequence of frames.
    """
    def __init__(self, FrameFolder, FPS, InitFreqList,\
                 StartFrame, MaxFrameN, FreqWidth, ExpName):
        """
        Constructor for the demultiplexer.

        Parameters
        ----------
        FrameFolder : Path object
            Path to the folder with the ASCII frames
        FPS : int
            Frames per second of the IR movie
        InitFreqList : list of float
            List with frequency bin suggestions
        StartFrame : int
            Index for the first frame to be used
        MaxFrameN : int
            Maximal number of frames to be used
        FreqWidth : int
            Uneven number of bins around a frequency
        ExpName : str
            Name of the experiment

        Raises
        ------
        ValueError if StartFrame > Total Frame Number
        """
        self.fps = FPS
        self.InitFreqList = InitFreqList
        self.FrameFolder = FrameFolder
        self.MaxFrameN = MaxFrameN
        self.ProjectName = ExpName
        self.FrameCount = 0
        self.Sentinel = getLogger("Watchtower")
        self.FrameFiles = self.getAllFrames(FrameFolder)
        if FreqWidth % 2 == 0:
            self.FreqWidth = FreqWidth+1
        else: self.FreqWidth = FreqWidth

        # load in the first frame to get the image dimensions
        self.FirstFrame = self.readFrame(self.FrameFiles[0])
        self.FrameDim = (len(self.FirstFrame), len(self.FirstFrame[0]))
        del self.FirstFrame
        self.FrameN = min([self.MaxFrameN, len(self.FrameFiles)])

        if StartFrame >= self.FrameN:
            ErrorMsg = "The Start Frame: {}".format(self.FrameN)+\
                       " is larger than the total number"+\
                       " of frames: {}".format(self.FrameN)
            self.Sentinel.error(ErrorMsg)
            raise ValueError(ErrorMsg)

        self.StartFrame = StartFrame
        # DFT Bin size / step size
        self.FreqStep = self.fps/(self.FrameN-StartFrame)
        # create a list of specific frequencies to extract
        self.FreqBinList = []
        self.FreqList = []
        for FInit in self.InitFreqList:
            FreqIdx = int(round(FInit/self.FreqStep, 1))
            self.FreqBinList.append(FreqIdx)
            self.FreqList.append(FreqIdx * self.FreqStep)
```

```
        # for each frequency create a new GoertzelImage Object
        self.GoertzelImg = []
        for FIdx, F in enumerate(self.FreqBinList):
            self.GoertzelImg.append(ImageGoertzelPeak(F, self.fps,\
                                                      self.FrameDim,\
                                                      (self.FrameN-StartFrame),\
                                                      FIdx, self.FreqWidth,\
                                                      self.ProjectName))
    # getter
    def getFreqList(self):
        return self.FreqList

    def getFreqBinList(self):
        return self.FreqBinList

    def getFPS(self):
        return self.fps

    def getPixelSize(self):
        return self.PixelSize

    def getFrameN(self):
        return self.FrameN

    def getFreqStepSize(self):
        return self.FreqStep

    def getFrameFileList(self):
        return self.FrameFiles

    def getStartFrame(self):
        return self.StartFrame

    def getFreqWidth(self):
        return self.FreqWidth

    def getAllFrames(self, FrameFolder):
        # extract the idx from the file name
        def getIndex(FileName):
            F = FileName.stem
            idx = 4  # at least the last 4 digits
            # or more for > 9999 files
            while F[-idx] in [str(k) for k in range(0, 10, 1)]:
                idx += 1
            return int(F[-(idx-1):])

        # grab all .asc files from the folder
        Files = FrameFolder.glob("*.asc")
        # sort the frame files by the last numeric index
        FrameFiles = sorted(Files, key = getIndex)

        return FrameFiles

    def readFrame(self, FilePath):
        """
        Reads in a thermogram from the ASCII raw data

        Parameters
        ----------
        FilePath : Path object
            Path to the ASCII (.asc) file

        Returns
        -------
        Image : 2D np.array
            Numpy arra with the 2D thermogram data
        """
```

```
        DataFile = open(str(FilePath.absolute()), "r")
        datalines = DataFile.readlines()
        DataFile.close()
        Image = []
        # look for the [Data] tag
        n = 0
        while "[Data]" not in datalines[n]:
            n += 1

        # read in all lines sequentially
        for line in range(n+1, len(datalines), 1):
            S = datalines[line].replace("," , ".")
            SArray = S.split("\t")
            NewLine = [float(k) for k in SArray[:-1]]
            Image.append(NewLine)

        return np.array(Image)

    def processFrames(self):
        """
        This method extractes the amplitudes and phases
        for each pixel from the image sequence
        """
        for FrameIdx in range(self.StartFrame, self.FrameN, 1):
            FramePath = self.FrameFiles[FrameIdx]
            NewFrame = self.readFrame(FramePath)
            self.Sentinel.info("Now processing frame: " + str(FramePath.name))
            self.FrameCount += 1
            for F in range(0, len(self.FreqList), 1):
                self.GoertzelImg[F].AddFrame(NewFrame)

        return True

    def saveData(self):
        for F in range(0, len(self.FreqList), 1):
            self.GoertzelImg[F].saveAmplData()
            self.GoertzelImg[F].savePhaseData()

        return True

    def plotData(self):
        for F in range(0, len(self.FreqList), 1):
            self.GoertzelImg[F].plotAmplImage()
            self.GoertzelImg[F].plotPhaseImage()

        return True

    def __str__(self):
        RString = "LIT Demultiplexer Object:\n"
        RString += "Project Name: {}".format(self.ProjectName)
        RString += "Sampling rate (fps): {:f} Hz\n".format(self.fps)
        RString += "Total number of Frames: {:s}".format(self.FrameN)
        RString += "Image Pixel Size: {:f} mm\n".format(self.PixelSize)
        RString += "Image Dimensions: {:s} Pixel\n".format(self.FrameDim)
        RString += "Frequency step size: {:f} Hz\n".format(self.FreqStep)
        RString += "Initial Frequency Ranges:\n"

        for FRange in self.InitFreqList:
            RString += str(FRange) + " Hz\n"
        RString += "Selected DFT Bins:\n"
        for F in FreqList:
            RString += str(F) + " Hz\n"
        RString += "Current State:\n"
        RString += "Frame Count: {:s}\n".format(self.FrameCount)

        return RString
```

A Source Code

A.1 Bravais Lattice Analysis

A.1.1 Phase Diagram Computation

```
import matplotlib.pyplot as plt
import numpy as np

"""
This skript calculates the (S, alpha) phase diagrams
for the unidirectional stretching of an hexagonal lattice.
"""

# User Input
SMin = 1       # minimal stretching factor
SMax = 10      # maximal stretching factor
AlphaMin = 0   # minimal angle with y-axis / DEG
AlphaMax = 30  # maximal angle with y-axis / DEG

# Setting Plot Parameters
params = {
   'axes.labelsize': 13,
   'font.size': 13,
   'legend.fontsize': 13,
   'xtick.labelsize': 13,
   'ytick.labelsize': 13,
   'text.usetex': False,
   'figure.figsize': [4 * 1.3, 4]
   }
plt.rcParams.update(params)

def calcLength(a):
    return np.sqrt(a[0]**2 + a[1]**2)

def calcAngle(a, b):
    DotP = a[0] * b[0] + a[1] * b[1]
    Angle = np.arccos(DotP/(calcLength(a)*calcLength(b)))
    return Angle

def stretchVector(a, AngleA, S):
    DotP = -a[0] * np.sin(AngleA) + a[1]*np.cos(AngleA)
    aPrime = [a[0] - (S-1)*DotP*np.sin(AngleA),\
              a[1] + (S-1)*DotP*np.cos(AngleA)]
    return aPrime

def reduceBasisVectors(c, d):
    # Lagrange Gauss algorithmus
    B1 = calcLength(c)**2
    mu = (c[0]*d[0] + c[1]*d[1])/B1
    d = [d[0] - int(round(mu))*c[0],\
         d[1] - int(round(mu))*c[1]]
    B2 = calcLength(d)**2

```

```
    while B2 < B1:
        [c, d] = [d, c]
        B1 = B2
        mu = (c[0]*d[0] + c[1]*d[1])/B1
        d = [d[0] - int(round(mu))*c[0],\
             d[1] - int(round(mu))*c[1]]
        B2 = calcLength(d)**2
    # map the angle to 60 - 90 degree
    if calcAngle(c, d) >= np.pi/2:
        c[0] = -c[0]
        c[1] = -c[1]

    return [c, d]

def calcBasisVectors(AngleA, Stretch):

    # initial grid vectors
    a = [1, 0]
    b = [1/2, -1/2*np.sqrt(3)]
    # stretch the vectors
    c = stretchVector(a, AngleA*np.pi/180, Stretch)
    d = stretchVector(b, AngleA*np.pi/180, Stretch)
    # reduce c and d to the smallest, orthogonal, vectors
    [c, d] = reduceBasisVectors(c, d)
    return [c, d]

def calcPhaseDiagram(SMin, SMax, AlphaMin, AlphaMax):

    # create the mesh
    SPoints = 100
    AlphaPoints = 100
    S = np.linspace(SMin, SMax, SPoints)
    Alpha = np.linspace(AlphaMin, AlphaMax, AlphaPoints)
    S, Alpha = np.meshgrid(S, Alpha)
    Phi = np.zeros([SPoints, AlphaPoints])
    Ratio = np.zeros([SPoints, AlphaPoints])

    # calculate the angles and ratios
    for SPos in range(0, SPoints, 1):
        for APos in range(0, AlphaPoints, 1):
            # calculate the grid vectors:
            [c, d] = calcBasisVectors(Alpha[SPos][APos], S[SPos][APos])
            Phi[SPos][APos] = calcAngle(c, d)
            Length_c = calcLength(c)
            Length_d = calcLength(d)
            Ratio[SPos][APos] = min([Length_c/Length_d,\
                                     Length_d/Length_c])
        print("Row calculation Nr. " + str(SPos) + " of " +\
              str(SPoints) + " Finished!")
    return [S, Alpha, Phi, Ratio]

def plotPhaseDiagrams(S, Alpha, Phi, Ratio):
    # plot the phase
    plt.figure()
    plt.pcolormesh(Alpha, S, Phi*180/np.pi,
                   cmap = "viridis", vmin = 60, vmax = 90,
                   shading = 'gouraud')
    plt.title("Angle between grid vectors")
    plt.ylabel("Stretch Factor")
    plt.xlabel("Stretch Direction [°]")
    plt.colorbar()
    plt.tight_layout()
    plt.savefig("PhaseDiagram_Angle.png", dpi = 600)
    plt.savefig("PhaseDiagram_Angle.pdf")
    plt.show()
    plt.close()

    # plot the ratio
```

```
    plt.figure()
    plt.pcolormesh(Alpha, S, Ratio, cmap = "jet",
                   vmin = 0, vmax = 1, shading = 'gouraud')
    plt.title("Ratio of grid vector lengths")
    plt.ylabel("Stretch Factor")
    plt.xlabel("Stretch Direction [°]")
    plt.colorbar()
    plt.tight_layout()
    plt.savefig("PhaseDiagram_Ratio.png", dpi = 600)
    plt.savefig("PhaseDiagram_Ratio.pdf")
    plt.show()
    return True

### MAIN

# calcluate the phase diagram
[S, Alpha, Phi, Ratio] = calcPhaseDiagram(SMin, SMax,\
                                          AlphaMin, AlphaMax)
print("PhaseDiagram finished.")
# plot the phase diagram
plotPhaseDiagrams(S, Alpha, Phi, Ratio)
print("Phase Diagrams plotted.")
```

A.1.2 Plot Stretched Lattices

```
import matplotlib.pyplot as plt
from numpy import sqrt, cos, sin, pi, arccos
from copy import deepcopy

### DASHBOARD
ImageSize = 4 # Size of the Image (d_Particle = 1)
AngleG = 0    # rotation angle in DEG
AngleA = 30   # rotation angle in DEG
Stretch = 3   # stretching factor in y-direction

# Setting Plot Parameters
params = {
   'axes.labelsize': 12,
   'font.size': 12,
   'legend.fontsize': 12,
   'xtick.labelsize': 12,
   'ytick.labelsize': 12,
   'text.usetex': False,
   'figure.figsize': [4 * 1.8, 4]
   }

plt.rcParams.update(params)

def rotateVector(P, angle):
    P_rot = [0,0]
    P_rot[0] = P[0]*cos(angle) - P[1]*sin(angle)
    P_rot[1] = P[0]*sin(angle) + P[1]*cos(angle)

    return P_rot

def calcAngle(a, b):
    DotP = a[0] * b[0] + a[1] * b[1]
    Angle = arccos(DotP / (calcLength(a) * calcLength(b)))

    return Angle

def calcHexParticles(RadialDist, AngleG, AngleA, Stretch):
    HexGrid = []
    # grid vectors
    a = [1, 0]
```

```
    b = [1/2, -1/2*sqrt(3)]
    # rotation angles in radians
    AngleA = AngleA * pi / 180
    AngleG = AngleG * pi / 180
    Bound = int(RadialDist*Stretch)*5
    for i in range(-Bound, Bound, 1):
        for j in range(-Bound, Bound, 1):
            # calc the particles
            P = [a[0] * i + b[0] * j,\
                 a[1] * i + b[1] * j]

            # cut out a circular region
            if P[0]**2 + P[1]**2 < RadialDist**2*1.5:

                # rotate it gamma
                P = rotateVector(P, AngleG)
                # rotate it alpha
                P = rotateVector(P, -AngleA)
                # Stretch it
                P[1]= P[1] * Stretch
                # rotate it alpha back
                P = rotateVector(P, AngleA)
                HexGrid.append(P)

    return HexGrid

def calcLength(a):
    return sqrt(a[0]**2 + a[1]**2)

def findShortestPair(VecList):

    # find the shortest vector
    ShortestVec = VecList[0]

    for V in VecList:
        if calcLength(ShortestVec) > calcLength(V):
            ShortestVec = V
    # find the second shortest one
    SecShortVec = [100, 100]
    for V in VecList:
        if calcLength(SecShortVec) > calcLength(V) and\
           V != ShortestVec:
            SecShortVec = V

    return [ShortestVec, SecShortVec]

def reduceBasisVectors(c, d):

    Invariant = abs(c[0]*d[1] - c[1]*d[0])
    SumLength = sum([calcLength(c), calcLength(d)])
    NewSumLength = SumLength - 1

    while (SumLength - NewSumLength) > 10**(-5):
        SumLength = sum([calcLength(c), calcLength(d)])
        # calculate the two other basis vectors
        f = [c[0] + d[0], c[1] + d[1]]
        g = [c[0] - d[0], c[1] - d[1]]
        [c, d] = findShortestPair([c, d, f, g])
        NewSumLength = sum([calcLength(c), calcLength(d)])

    if calcAngle(c, d) >= pi/2:
        c[0] = -c[0]
        c[1] = -c[1]

    # make sure the invariant stayed constant
    NewInv = abs(c[0]*d[1] - c[1]*d[0])
    if abs(Invariant - NewInv) > 10**(-8):
        print("Invariant has changed!!")
```

```
        assert False

    return [c, d]

def calcBasisVectors(AngleG, AngleA, Stretch):

    # grid vectors
    a = [1, 0]
    b = [1/2, -1/2*sqrt(3)]
    # rotation angles in radians
    AngleA = AngleA * pi / 180
    AngleG = AngleG * pi / 180
    # calculate a set of stretched basis vectors
    # rotate it beta
    c = rotateVector(a, AngleG)
    d = rotateVector(b, AngleG)
    # rotate it alpha
    c = rotateVector(c, -AngleA)
    d = rotateVector(d, -AngleA)
    # Stretch it
    c[1]= c[1] * Stretch
    d[1]= d[1] * Stretch
    # rotate it alpha back
    c = rotateVector(c, AngleA)
    d = rotateVector(d, AngleA)
    # reduce c and d to the smallest vectors
    [c, d] = reduceBasisVectors(c, d)

    return [a, b, c, d]

def plotParticles(HexGrid, HexGridStretched, HexCore, HexCoreStretched,\
                  GridVec, ImageSize, AngleG, AngleA, Stretch):

    fig = plt.figure()
    ax1 = fig.add_subplot(1,2,1, aspect = "equal")
    ax2 = fig.add_subplot(1,2,2, aspect = "equal")
    CoreColors = ["r", "y", "b", "m", "r", "b", "c",\
                  "g", "g", "k", "g", "g", "c", "b",\
                  "r", "m", "b", "y", "r"]
    LSCore =     ["dotted", "dotted", "dotted", "dotted", "dashed",\
                  "dashed", "dotted", "dotted", "dashed", "-",\
                  "dashed", "dotted", "dotted", "dashed", "dashed",\
                  "dotted", "dotted", "dotted", "dotted"]
    for P in HexGrid:
        if P not in HexCore:
            circ = plt.Circle((P[0],P[1]), radius = 0.45,\
                              color = "k", fill = False, lw = 1)
            ax1.add_patch(circ)

    for P in HexGridStretched:
        if P not in HexCoreStretched:
            circ = plt.Circle((P[0],P[1]), radius = 0.45,\
                              color = "k", fill = False, lw = 1)
            ax2.add_patch(circ)

    for P in range(0, len(HexCore), 1):
        circ = plt.Circle((HexCore[P][0], HexCore[P][1]), radius = 0.45,\
                          color = CoreColors[P], fill = False,\
                          linestyle = LSCore[P], lw = 1)
        ax1.add_patch(circ)

    for P in range(0, len(HexCoreStretched), 1):
        circ = plt.Circle((HexCoreStretched[P][0], HexCoreStretched[P][1]),\
                          radius = 0.45, color = CoreColors[P],\
                          fill = False, linestyle = LSCore[P], lw = 1)
        ax2.add_patch(circ)

    ax1.plot([0], [0], "ro", markersize = 2) # mark the central particle
```

```
    ax2.plot([0], [0], "ro", markersize = 2) # mark the central particle
    ax1.set_xlim(-ImageSize/2*Stretch, ImageSize/2*Stretch)
    ax1.set_ylim(-ImageSize/2*Stretch, ImageSize/2*Stretch)
    ax2.set_xlim(-ImageSize/2*Stretch, ImageSize/2*Stretch)
    ax2.set_ylim(-ImageSize/2*Stretch, ImageSize/2*Stretch)

    #plot the stretching vectors
    Sx = -Stretch * sin(AngleA * pi / 180)
    Sy = Stretch * cos(AngleA * pi / 180)
    ax1.arrow(0, 0, Sx, Sy, head_width = 0.1,\
              head_length = 0.1, fc = 'r', ec = 'k')
    ax2.arrow(0, 0, Sx, Sy, head_width = 0.1,\
              head_length = 0.1, fc = 'r', ec = 'k')

    # plot the grid vectors
    #a
    ax1.arrow(0, 0, GridVec[0][0], GridVec[0][1], head_width = 0.1,\
              head_length = 0.1, fc = 'g', ec = 'b')
    #b
    ax1.arrow(0, 0, GridVec[1][0], GridVec[1][1], head_width = 0.1,\
              head_length = 0.1, fc = 'g', ec = 'b')
    #c
    ax2.arrow(0, 0, GridVec[2][0], GridVec[2][1], head_width = 0.1,\
              head_length = 0.1, fc = 'g', ec = 'b')
    #d
    ax2.arrow(0, 0, GridVec[3][0], GridVec[3][1], head_width = 0.1,\
              head_length = 0.1, fc = 'g', ec = 'b')
    ax1.set_xlabel("x")
    ax1.set_ylabel("y")
    ax2.set_xlabel("x")
    ax2.set_ylabel("y")
    ax1.set_xticks([-5, 0, 5])
    ax1.set_yticks([-5, 0, 5])
    ax2.set_xticks([-5, 0, 5])
    ax2.set_yticks([-5, 0, 5])
    ax1.set_title(r"$\beta$ = " + str(AngleG))
    ax2.set_title(r"$\alpha$ = " + str(AngleA) + "°, S = " + str(round(Stretch, 5)))
    G = str(AngleG).replace(".", "_")
    A = str(AngleA).replace(".", "_")
    S = str(Stretch).replace(".", "_")
    plt.tight_layout()
    plt.savefig("HexGrid_Gamma" + G + "Alpha" + A + "S" + S + ".png", dpi = 600)
    plt.savefig("HexGrid_Gamma" + G + "Alpha" + A + "S" + S + ".pdf")
    plt.savefig("HexGrid_Gamma" + G + "Alpha" + A + "S" + S + ".eps")
    plt.show()

    return True

# MAIN
# calcluate a circular region with a dense hex grid rotate by AngleG
HexGrid = calcHexParticles(ImageSize*5, AngleG, 0, 1)
print("Rotated HexGrid finished.")

# calculate the 7 core particles of the hex grid rotated by AngleG
HexCore = calcHexParticles(2, AngleG, 0, 1)
print("Rotated HexCore finished.")

# calculate the 7 core particles of the hex grid rotated and stretched
HexCoreStretched = calcHexParticles(2, AngleG, AngleA, Stretch)
print("Rotated HexCoreStretched finished.")

HexGridStretched = calcHexParticles(ImageSize*5, AngleG, AngleA, Stretch)
print("Rotated HexGridStretched finished.")

# calculate the four grid vectors
GridVec = calcBasisVectors(AngleG, AngleA, Stretch)

# plot the HexGrid and the stretched HexGrid
```

```
plotParticles(HexGrid, HexGridStretched, HexCore, HexCoreStretched,\
                  GridVec, ImageSize, AngleG, AngleA, Stretch)
print("Plotting finished.")
```

A.2 Computing Colloidal Glasses

```
from matplotlib.pyplot import *
from datetime import datetime
from math import sqrt, pi

### DASHBOARD
# Boundaries for the Selection Box
ExpName = "Test_Input"      # Name of the File and Experiment
CoorFile = "TestCoor.csv"   # Name of the Coor File (mit Endung z.B: .txt)
BoxSize = 5.5  #µm
# Particle Properties
Radius = 0.5                # Normalisierter Radius

# Setting Plot Parameters
params = {
   'axes.labelsize': 12,
   'font.size': 12,
   'legend.fontsize': 12,
   'xtick.labelsize': 12,
   'ytick.labelsize': 12,
   'text.usetex': False,
   'figure.figsize': [4.5 * 1.618, 4.5]
   }

rcParams.update(params)

def getRawData(FileName):
    """
    This function read in the raw particles positions and radii
    from the .csv file

    Parameters
    ----------
        FileName : str
            Name for the ascii file with the particle coordinates

    Returns
    -------
        ParticleList : list of list of float
            List of [x,y,z,r] particle coordinates
    """

    DataFile = open(FileName, "r")
    datalines = DataFile.readlines()
    DataFile.close()
    ParticleList = []
    for line in datalines:
        temp = line.split(",")
        Particle = []
        for k in temp:
            Particle.append(float(k))
        ParticleList.append(Particle)

    return ParticleList

def SelectParticles(ParticleList, BoxSize):
    """
    This function selects all particles within a box with of size
    "BoxSize" and returns them in a new list. The box is placed in
    the center of the particle cloud

    Parameters
    ----------
        ParticleList : list of list of float
            List of [x,y,z,r] particle coordinates
```

```
65          BoxSize : float
66              edge length of the selection cube
67      Returns
68      -------
69          NewPartList : list of list of float
70              List of [x,y,z,r] particle coordinates with their
71              centers inside the box volume
72      """
73      # empty lists for the selected particles
74      NewPartList = []
75      ZMax = max([ParticleList[Z][2] for Z in range(0, len(ParticleList), 1)])
76      print("zMax/2: " + str(ZMax/2))
77      # go through all particles and find the ones inside the box + radius
78      for Particle in ParticleList:
79          if abs(Particle[0]) <= (BoxSize/2 + Particle[3]):
80              if abs(Particle[1]) <= (BoxSize/2 + Particle[3]):
81                  if abs(Particle[2] - ZMax/2) <= (BoxSize/2 + Particle[3]):
82                      CenteredPart = Particle
83                      CenteredPart[2] -= ZMax/2
84                      NewPartList.append(CenteredPart)
85
86      return NewPartList
87
88  def rescaleParticleSet(ParticleList, Radius):
89      """
90      This function rescales all particle positions and diameters
91      such that the new diameter of the particles is the
92      one specified by the used in the variable "Radius".
93
94      Parameters
95      ----------
96          ParticleList : list of list of float
97              List of [x,y,z,r] particle coordinates
98          Radius : float
99              New radius for all particle in the particle list
100
101     """
102     NewPartList = []
103     for Particle in ParticleList:
104         RescalePart = []
105         OldRad = Particle[-1]
106         for k in Particle:
107             RescalePart.append(k*Radius/OldRad)
108         NewPartList.append(RescalePart)
109
110     return NewPartList
111
112 def createAutoCADFile(P_List, BoxSize, ExpName):
113     """
114     This function creates the AutoCAD file with
115     all the particles in it
116     """
117     CADFile = open(ExpName + "_CAD.scr", "a")
118     writeParticles(CADFile, P_List, 0)
119     writeBox(CADFile, 0, 0, 0, BoxSize)
120     CADFile.close()
121
122     return True
123
124 def writeParticles(File, ParticleList, XOffset):
125     """
126     This function writes the list of particles with a specific
127     Offset in the X-direction into the provided File
128
129     Parameters
130     ----------
131         File : io file object
132             ASCII file to write in
```

```
        ParticleList : list of list of float
            List of [x,y,z,r] particle coordinates
        XOffset : float
            Offset in the X-direction for all particles
    """
    # write the AutoCAD code for every particle
    for Particle in ParticleList:
        File.write("SPHERE\n")
        File.write(str(Particle[0] - XOffset) + ",") # X
        File.write(str(Particle[1]) + ",") # Y
        File.write(str(Particle[2]) + "\n")# Z
        # save the diameter for AutoCAD
        File.write(str(Particle[3]) + "\n")

    return True

def writeBox(File, XPos, YPos, ZPos, BoxSize):
    """
    This function writes a box with the center at
    [XPos, YPos, ZPos] of size "BoxSize". The box
    faces are aligned with the axes of the coordinate
    system.

    Parameters
    ----------
        File : io file object
            ASCII file to write in
        XPos,YPos,ZPos : floats
            Coordinates of the box center
        BoxSize : float
            Edge length of the box
    """
    File.write("BOX\n")
    File.write("C\n")
    File.write(str(XPos) + "," + str(YPos) + "," + str(ZPos) + "\n")
    File.write("C\n")
    File.write("{}\n".format(BoxSize))

    return True

### MAIN

#read in all particle coordinates
ParticleList = getRawData(CoorFile)
print("Data read-in Finished.")

# extract the particles with their centers inside the BOX
ParticleList = SelectParticles(ParticleList, BoxSize)
print(str(len(ParticleList)) + " Particles Selected.")

# Rescale them to the overall Diameter
ParticleList = rescaleParticleSet(ParticleList, Radius)
print("Particles Resized")

#Write AutoCAD Files
createAutoCADFile(ParticleList, BoxSize, ExpName)
print("AutoCAD File Finished")
print("Program Finished!")
```

A.3 Lock-In Thermography Analysis

A.3.1 Main LIT-Analysis Class

```python
import logging
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from LIT_RScale_Analyzer import RScale_Analyzer

class LIT_Analyzer():
    """Class to analyze LIT measurements of a single sample"""
    def __init__(self, ExpName, AutoRun):
        """
        Initialize LIT_Analyzer class

        Parameters
        ----------
        ExpName : str
           Name part used for all saved data files
        """
        self.Sentinel = logging.getLogger("Watchtower")
        self.ExpName = ExpName
        # initialize all variables
        self.RawDataFolderPath = None
        self.RScale = None
        self.SweepData = None
        self.RScale_Instrument = None
        self.AutoRun = AutoRun
        # change plotting parameters
        plt.rcParams.update({'font.size': 14})

    def processRScaleImage(self, ImgPath, LinePeriod):
        """
        Computes the pixel RScale from the reference Image

        Parameters
        ----------
        ImgPath : Path
            Path to the image of the reference wafer
        LinePeriod : float
            Periodicity of the lines on the reference wafer
        """
        self.RScale_Instrument = RScale_Analyzer(ImgPath, LinePeriod,
                                                 self.ExpName)
        self.RScale = self.RScale_Instrument.AnalyzeRScale()
        self.Sentinel.info("New RScale computed: {}".format(self.RScale)+\
                           " mm/Pixel")
        return True

    def setRScale(self, NewRScale):
        """
        Alternative to processRScaleImage if the same
        scaleing is used for all samples.

        Parameters
        ----------
        NewRScale : float, positive
            Pixel size of the IR camera in mm /pixel

        Raises
        ------
        ValueError : If NewRScale is <= 0
        """
```

```
        if NewRScale <= 0:
            ErrorMsg = "The new RScale: {}".format(NewRScale)+\
                       "is less then 0! Use a positive number."
            self.Sentinel.error(ErrorMsg)
            raise ValueError(ErrorMsg)

        self.RScale = NewRScale
        self.RScale_Instrument.RScale = NewRScale
        return True

    def setRawDataFolderPath(self, NewRawDataPath):
        self.RawDataFolderPath = NewRawDataPath
        return True

    def computeSAlpha(self, LockInFreq, AmplFitList, PhaseFitList):
        """
        Computes the standard deviation of the thermal diffusivity
        by Gaussian error propagation

        Parameters
        ----------
        LockInFreq : float, positive
            Frequency of the lock in measurements

        AmplFitList, PhaseFitList : list
            contains: [[slope, y-intersect], std_slope, EndPosList]
        """
        Std = (np.pi * LockInFreq /(AmplFitList[0][0] *\
                                    PhaseFitList[0][0]**2) *\
                                    PhaseFitList[1])**2
        Std += (np.pi * LockInFreq /(AmplFitList[0][0]**2 *\
                                     PhaseFitList[0][0]) *\
                                     AmplFitList[1])**2

        return Std

    def calcAlphaLinReg(self, Sweep):
        """
        Computes the overall thermal diffusivity from a range of different frequencies.
        The product of the slopes is fitted with a line that passes through the origin.

        Parameters
        ----------
            FreqList : list of float
                List with the used Lock-In frequencies
            m_Phase : list of float
                List with the slopes of the linear phase fits
            SPFitList : list of float
                List with the standard deviation of the phase slopes
            m_Ampl : list of float
                List with the slopes of the linear amplitude fits
            SAFitList : list of float
                List with the standard deviation of the slopes
        Returns
        -------
            [[AlphaLinReg, SAlphaLinReg,\
            SlopeProd, SProdError, LinearSlope] : floats
                Thermal diffusivity and standard deviation in mm^2/s
                Product of the amplitude and phase slopes with error
                Slope of the linear fit
        """
        FreqList = Sweep.getFreqList()
        [m_Phase, SPFitList] = Sweep.getPhaseSlopeList()
        [m_Ampl, SAFitList] = Sweep.getAmplSlopeList()
        # multiply the slopes of amplitude and phase
        SlopeProd = np.array(m_Phase) * np.array(m_Ampl)
        SProdError = np.sqrt((np.array(m_Phase) * np.array(SAFitList))**2 +\
                             (np.array(m_Ampl) * np.array(SPFitList))**2)
```

```
129         # initial guess for the slope of the slope product
130         LinearSlopeInit = sum(abs(SlopeProd) * np.array(FreqList)) /\
131                           sum(np.array(FreqList)**2)
132         def fitFunc(f, m):
133             return f * m
134         Fit = curve_fit(fitFunc, FreqList, SlopeProd,\
135                         p0 = LinearSlopeInit)
136         LinearSlope = abs(Fit[0][0])
137         SLinSlope = np.sqrt(Fit[1][0][0])
138         # linear fit of the product versus frequency
139         AlphaLinReg = abs(np.pi / LinearSlope)
140         SAlphaLinReg = abs(np.pi/LinearSlope**2*SLinSlope)
141 
142         return [AlphaLinReg, SAlphaLinReg, SlopeProd, SProdError, LinearSlope]
143 
144     def LinearFit(self, X, Y):
145         """
146         Computes a linear fit on the input (X/Y) data
147         """
148         Lin_Fit = np.polyfit(X, Y, deg = 1, cov = True)
149         S_Lin_Fit = np.sqrt(Lin_Fit[1][0][0]) # sqrt(Var(m))
150         return [Lin_Fit[0], S_Lin_Fit]
151 
152     def LinFitRegion(self, LinDist, LinY, Type, MinDist,\
153                      MaxDist, Frequency):
154         """
155         Performs a linear fit on the linearized (Dist / Y) data.
156         The the predefined distance boundaries are used by default.
157         The user can also select the boundaries by clicking on the plot.
158 
159         Parameters
160         ----------
161         LinDist : numpy array
162             Array with the linearized distance from the laser heating
163         LinY : numpy array
164             Array with the linearized amplitude data
165         Type : data type, either "Amplitude" or "Phase"
166         MinDist : float, positive
167             Minimal distance from the heating origin to use
168         MaxDist : float, positive
169             Maximal distance from the heating origint to use
170         Frequency : float, positive
171             Lock In Frequency of the measurement
172 
173         Raises
174         ------
175         ValueError if Type is not recognized
176 
177         Returns
178         -------
179         Y_Fit : list
180             slope and y intersection of the linear fit
181         S_Y_Fit : list
182             standard deviations for the slope and y intersection
183         EndPosList : list of lists
184             Index and value of the minimal and maximal distance used
185         """
186         LinDist = np.array(LinDist)
187         LinY = np.array(LinY)
188 
189         def onMouseClick(event):
190             if event.xdata == None:
191                 return True
192             # save the position of the mouse when the button was pressed
193             # find the locations of the end Positions
194             BoundIdx = np.argmin(abs(LinDist - event.xdata))
195             Position = [event.xdata, BoundIdx]
196             # display the recieved data to the user
```

```
            self.Sentinel.info("You pressed Key: " + ascii(event.key) +\
              " at position in Data Koordinates: " + str([event.xdata, event.ydata]))
            # find the closer boundary
            if Position[0] <= EndPosList[0][0] or \
               abs(Position[0] - EndPosList[0][0]) <\
               abs(Position[0] - EndPosList[1][0]):
                # move the left boundary
                EndPosList[0] = Position
                # replot the left boundary
                LeftBound.set_xdata([Position[0]]*2)

            elif abs(Position[0] - EndPosList[0][0]) >\
                 abs(Position[0] - EndPosList[1][0]) or\
                 Position[0] >= EndPosList[1][0]:
                # move the right boundary
                EndPosList[1] = Position
                # replot the right boundary
                RightBound.set_xdata([Position[0]]*2)

            # redo the linear fit
            Left = EndPosList[0][1]
            Right = EndPosList[1][1]
            [Y_Fit, S_Y_Fit] = self.LinearFit(LinDist[Left:Right],\
                                              LinY[Left:Right])
            # plot the linear fit
            X_Line = [LinDist[Left], LinDist[Right]]
            Y_Line = Y_Fit[0] * np.array(X_Line) + Y_Fit[1]
            FitGraph.set_data(X_Line, Y_Line)
            plt.draw()  # redraw everything
            return True

        StartIdx = np.argmin(abs(MinDist - LinDist))
        EndIdx = np.argmin(abs(MaxDist - LinDist))
    # list for the boundaries [left, right]
        EndPosList = [[MinDist, StartIdx], [MaxDist,  EndIdx]]
        fig = plt.figure()
        # connect the MouseClickEvent with the Canvas
        fig.canvas.mpl_connect('button_press_event', onMouseClick)
        ax = plt.subplot()
        # plot the measured amplitude data
        ax.plot(LinDist, LinY, ".", markersize = 1)
        # do the linear fit
        Left = EndPosList[0][1]
        Right = EndPosList[1][1]
        [Y_Fit, S_Y_Fit] = self.LinearFit(LinDist[Left:Right],\
                                          LinY[Left:Right])

        if self.AutoRun == False:
            # plot the linear fit
            X_Line = [LinDist[Left], LinDist[Right]]
            Y_Line = Y_Fit[0] * np.array(X_Line) + Y_Fit[1]
            # create two boundary dummy plots where the data is updated later on
            YExtent = ax.get_ylim()
            LeftBound, = ax.plot([MinDist, MinDist], YExtent, "k--")
            RightBound, = ax.plot([MaxDist, MaxDist], YExtent, "k--")
            # create a plot for the linear fit
            FitGraph, = ax.plot(X_Line, Y_Line, "r-", lw = 2, label = "Linear Fit")
            ax.set_xlabel("Distance / mm")

            if Type == "Amplitude":
                ax.set_ylabel("Linearized Amplitude")
                YLimits = ax.get_ylim()
                ax.set_ylim(max([-2, YLimits[0]]))

            elif Type == "Phase":
                ax.set_ylabel("Phase / rad")
                NewYMin = min([P for P in LinY if P < 2*np.pi and P > -2*np.pi])
                NewYMax = max([P for P in LinY if P < 2*np.pi and P > -2*np.pi])
```

```
                ax.set_ylim(NewYMin, NewYMax)

            else:
                ErrorMsg = "The Type {} for the linearized plot".format(Type)+\
                           " is not recognized. Use either Amplitude or Phase"
                self.Sentine.error(ErrorMsg)
                raise ValueError(ErrorMsg)

            ax.set_title("Frequency: " + str(Frequency) + " Hz")
            fig.tight_layout()
            plt.show()
        self.Sentinel.info("Linear amplitude fit completed.")
        return [Y_Fit, S_Y_Fit, EndPosList]
```

A.3.2 Inherited Point Excitation LIT-Analysis Class

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.colors import LogNorm
from matplotlib.ticker import MaxNLocator
from LIT_SweepData import LIT_SweepData
from LIT_Analyzer import LIT_Analyzer

class LIT_PointAnalyzer(LIT_Analyzer):
    """Class to analyze LIT measurements with a point excitation"""

    def __init__(self, ExpName, AutoRun):
        """
        Initialize LIT_PointAnalyzer class

        Parameters
        ----------
        ExpName : str
           Name part used for all saved data files
        """
        LIT_Analyzer.__init__(self, ExpName, AutoRun)

    def FindCenterFromAmp(self, AmplData):
        """
        Determines the center point for a point laser excitation.
        The center is the pixel with the largest amplitude.

        Parameters
        ----------
        AmplData : numpy array
            Contains the 2D amplitude image

        Returns
        -------
        MaxRow : int
            Row index for the pixel with maximal ampltiude
        MaxLine : int
            Line index for the pixel with maximal amplitude
        MaxAmpl : float
           Value of the maximal amplitude
        """
        # initialize the search parameters
        MaxLine = 0
        MaxRow = 0
        MaxAmpl = AmplData[2][0][0]
        # go throug every Pixel
        for Line in range(0,len(AmplData[2]),1):
            for Row in range(0,len(AmplData[2][Line]),1):

```

```
50                  # check if the current pixel value is
51                  # higher than the previous maximum
52                  if AmplData[2][Line][Row] > MaxAmpl:
53                      MaxLine = Line
54                      MaxRow = Row
55                      MaxAmpl = AmplData[2][Line][Row]
56          self.Sentinel.info("Center detected at: " +\
57                             str([MaxRow, MaxLine, MaxAmpl]))
58          return [MaxRow, MaxLine, MaxAmpl]
59
60      def CalcRadialData(self, Data, Center, MaxR):
61          """
62          Computes the radial distance from each pixel
63          to the center pixel within a predefined range.
64
65          Parameters
66          ----------
67          Data : numpy array
68              Contains the 2D image data
69          Center : list of float
70              Contains the X,Y,A center pixel data
71          MaxR : float
72              Maximal radius for the conversion in pixel
73
74          Returns
75          -------
76          Radius : list of float
77              List of all the radial distances
78          Y : list of float
79              List with the corresponding y-values
80          """
81          # initialize empty arrays for the calculated data
82          Y = []
83          Radius = []
84          Lmin = int(max([0, Center[1] - MaxR]))
85          Lmax = int(min([len(Data[2]), Center[1] + MaxR]))
86          Rmin = int(max([0, Center[0] - MaxR]))
87          Rmax = int(min([len(Data[2][0]), Center[0] + MaxR]))
88          # convert to Radial Data
89          for Line in range(Lmin, Lmax,1):
90              for Row in range(Rmin, Rmax, 1):
91                  RadDist = np.sqrt((Data[0][Line][Row]-Center[0])**2+\
92                                    (Data[1][Line][Row]-Center[1])**2)
93                  if RadDist < MaxR:
94                      Y.append(Data[2][Line][Row])  # Intensity stays the same
95                      Radius.append(RadDist)
96          # Sort by increasing Radius and bring order to the Universe
97          Radius, Y = (list(t) for t in zip(*sorted(zip(Radius, Y))))
98          self.Sentinel.info("2D to radial conversion finished!")
99          return [Radius, Y]
100
101     def SelectMaxR(self, Measurement):
102         """
103         Shows the amplitude image with the center.
104         The user can then select an appropriate maximal radius
105         for the radial average.
106
107         Parameters
108         ----------
109             Measurement : LIT_SinglePointMeas object
110                 contains all data from a single measurement
111         Returns
112         -------
113             MaxR : float
114                 Maximal radius for the radial average in pixel
115         """
116         # default maximal radius in pixel:
117         MaxR = min(Measurement.RawAmplData[2].shape)/3
```

```
118
119         def onMouseClick(event):
120             if event.xdata != None:
121                 # compute the radial distance to the center
122                 MaxR = np.sqrt((Measurement.Center[0]-event.xdata/self.RScale)**2+\
123                                (Measurement.Center[1]-event.ydata/self.RScale)**2)
124                 self.Sentinel.info("New MaxR selected: {}".format(MaxR*self.RScale))
125             else:
126                 return True
127             # display the recieved data to the user
128             self.Sentinel.info("You pressed Key: " + ascii(event.key) +\
129               " at position in Data Koordinates: " + str([event.xdata,\
130                                                           event.ydata]))
131             # plot the new cricle around the center
132             circle1.set_radius(MaxR*self.RScale)
133             plt.draw()  # redraw everything
134             return True
135
136         fig = Measurement.getRawAmplCanvas()
137         # connect the MouseClickEvent with the Canvas
138         fig.canvas.mpl_connect('button_press_event', onMouseClick)
139         ax1 = fig.gca()
140         ax1.plot([Measurement.Center[0]*self.RScale],\
141                  [Measurement.Center[1]*self.RScale], "ro")
142         # plot the circle around the center
143         circle1 = plt.Circle(np.array(Measurement.Center[0:2])*self.RScale,\
144                              MaxR*self.RScale,\
145                              color = 'r', fill = False)
146         ax1.add_artist(circle1)
147         self.Sentinel.info("Click on the image to change the radial distance")
148         plt.show()
149         return MaxR
150
151     def convertToRadial(self, Measurement):
152         """
153         Converts the raw data from a single point measurement
154         into radial data. The use selects the maximal radial distance.
155
156         Parameters
157         ----------
158         Measurement : SinglePointMeas object
159             Contains all raw and result data from a single point
160             excitation measurement
161         """
162         # compute the measurement ID
163         Measurement.ID = str(Measurement.LockInFreq).replace(".","_") + "_Hz"
164         # find the center pixel
165         Measurement.Center = self.FindCenterFromAmp(Measurement.RawAmplData)
166         # let the user select the maximal radius
167         if AutoRun == True:
168             Measurement.MaxR = Measurement.MaxDist * 2
169         else:
170             Measurement.MaxR = self.SelectMaxR(Measurement)
171         # convert the amplitude
172         [AmplRadii, RadialAmpl] = self.CalcRadialData(Measurement.RawAmplData,\
173                                                       Measurement.Center,\
174                                                       Measurement.MaxR)
175         Measurement.AmplRadii = AmplRadii
176         Measurement.RadialAmpl = RadialAmpl
177         # convert the phase
178         [PhaseRadii, RadialPhase] = self.CalcRadialData(Measurement.RawPhaseData,\
179                                                         Measurement.Center,\
180                                                         Measurement.MaxR)
181         [PhaseRadii, RadialPhase] = np.unwrap([PhaseRadii, RadialPhase])
182         Measurement.PhaseRadii = PhaseRadii
183         Measurement.RadialPhase = RadialPhase
184         # plot the 2D zoomed Amplitude
185         Measurement.PlotRadialAmplZoom(self.SweepData.PlotPathZoom2D)
```

```
186         # plot the 2D zoomed Phase
187         Measurement.PlotRadialPhaseZoom(self.SweepData.PlotPathZoom2D)
188         return True
189
190     def AnalyzeIsotropicMeas(self, RawDataFolder, MinDist, MaxDist):
191         """
192         Analyzes the LIT measurement of an isotropic
193         sample excited with a point source (laser)
194         at one or multiple frequencies.
195
196         Parameters
197         ----------
198             RawDataFolder : Path object
199                 path to the folder containing all the ASCII raw data files
200             MinDist : float, >0 in [mm]
201                 default minimal distance used for the linear fit
202             MaxDist : float, >0 in [mm]
203                 default maximal distance used for the linear fit
204         """
205         # read in all the files from a sweep
206         self.RawDataFolderPath = RawDataFolder
207         self.SweepData = LIT_SweepData(self.RawDataFolderPath,\
208                                        self.RScale_Instrument,\
209                                        self.ExpName, "Point")
210         # compute the radial average for all files
211         for Measurement in self.SweepData.SMeasList:
212
213             # save some data in the singe frequency objects
214             Measurement.MinDist = MinDist
215             Measurement.MaxDist = MaxDist
216             # convert the 2D Data to 1D radial data
217             self.convertToRadial(Measurement)
218             # fit the linearized amplitude
219             LinAmplDist = np.array(Measurement.AmplRadii)*self.RScale
220             LinAmpl = np.log(np.array(Measurement.RadialAmpl) * np.sqrt(LinAmplDist))
221             Measurement.LinAmplData = [LinAmplDist, LinAmpl]
222             Measurement.AmplFitData = self.LinFitRegion(LinAmplDist, LinAmpl,\
223                                                         "Amplitude", MinDist, MaxDist,\
224                                                         Measurement.LockInFreq)
225             # plot the linear amplitude fit
226             Measurement.plotAmplFit(self.SweepData.PlotPath1D)
227             # fit the linearized phase
228             LinPhaseDist = np.array(Measurement.PhaseRadii)*self.RScale
229             LinPhase = np.array(Measurement.RadialPhase)
230             Measurement.LinPhaseData = [LinPhaseDist, LinPhase]
231             Measurement.PhaseFitData = self.LinFitRegion(LinPhaseDist, LinPhase,\
232                                                          "Phase", MinDist, MaxDist,\
233                                                          Measurement.LockInFreq)
234             # plot the linear phase fit
235             Measurement.plotPhaseFit(self.SweepData.PlotPath1D)
236             # compute the thermal diffusivity
237             Measurement.ThermalDiff = abs((np.pi*Measurement.LockInFreq)/\
238                                           (Measurement.AmplFitData[0][0] *\
239                                            Measurement.PhaseFitData[0][0]))
240             Measurement.StdThermalDiff = self.computeSAlpha(Measurement.LockInFreq,\
241                                                             Measurement.AmplFitData,\
242                                                             Measurement.PhaseFitData)
243             self.Sentinel.info("Thermal diffusivity:"+\
244                                " {} mm^2/s".format(Measurement.ThermalDiff)+\
245                                " for Lock-in Frequency:"+\
246                                " {} Hz".format(Measurement.LockInFreq))
247         # analyze multifrequency measurements (single frequency are also analyzed)
248         self.SweepData.MultiFreq = True
249         # compute the thermal diffusivity with the linear interpolation
250         self.SweepData.AlphaSlope = self.calcAlphaLinReg(self.SweepData)
251         self.SweepData.plotAlphaLinReg()
252         self.Sentinel.info("Thermal diffusivity from the linear regression: "+\
253                            " {} +/- {} mm^2/s".format(self.SweepData.AlphaSlope[0],\
```

```
                                                        self.SweepData.AlphaSlope[1]))
        self.SweepData.plotTDiffvsFreq()
        # save the data
        self.SweepData.saveData()
        # create the report
        self.SweepData.createReport()
        return True
```

A.3.3 Inherited Line Excitation LIT-Analysis Class

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.colors import LogNorm
from matplotlib.ticker import MaxNLocator
from scipy.optimize import curve_fit
from LIT_SweepData import LIT_SweepData
from LIT_Analyzer import LIT_Analyzer


class LIT_LineAnalyzer(LIT_Analyzer):
    """Class to analyze LIT measurements with a line excitation"""

    def __init__(self, ExpName, AutoRun):
        """
        Initialize LIT_Analyzer class

        Parameters
        ----------
        ExpName : str
           Name part used for all saved data files
        """
        LIT_Analyzer.__init__(self, ExpName, AutoRun)

    def plotTDiffDirection(self, Data):
        """
        Plots the thermal diffusivity vs. the direction

        Parameters
        ----------
           Data : list of SingleMeasData
        """
        # extract the thermal diffusivity vs. direction
        TDiffList = []
        StdTDiffList = []
        AngleList = []
        for D in Data:
            #forward
            TDiffList.append(D.ThermalDiff)
            StdTDiffList.append(D.StdThermalDiff)
            AngleList.append(D.CenterLine[2])
            #backward
            TDiffList.append(D.ThermalDiff)
            StdTDiffList.append(D.StdThermalDiff)
            AngleList.append(D.CenterLine[2] + np.pi)
        # plot in both directions (Angle, Angle + 180Deg)
        ax = plt.subplot(111, projection='polar')
        ax.errorbar(AngleList, TDiffList, yerr = StdTDiffList,\
                    fmt = ".", capsize=0)
        #ax.set_rlabel_position(-22.5)
        ax.grid(True)
        ax.set_title(self.ExpName + "\n TDiff / $mm^2/s$")
        plt.legend(loc = "upper left", numpoints = 1, frameon = False)
        plt.tight_layout()
        plt.savefig(self.ExpName + "_TDiff_Angle.png", dpi = 600)
```

```
56          plt.close()
57
58      def CalcLineData(self, Data, CenterLine, MaxDist, CenterOffset):
59          """
60          Computes the distance to the center line for each
61          pixel in a predefined area
62
63          Parameters
64          ----------
65          Data : numpy array
66              Contains the 2D image data
67          CenterLine : LaserLine object
68              Contains the location and direction of the line laser
69          CenterOffset : float,
70              Valid range on both sides from the center in millimeters
71          MaxDist : float
72              Valid range parallel to the center line in millimeters
73
74          Returns
75          -------
76          LinDist : list of float
77              List of all the linear distances
78          Y : list of float
79              List with the corresponding y-values
80          """
81          # find appropriate boundaries for perfomance
82          LinDist = []
83          Y = []
84          # convert to linear data, go through all pixels
85          for Line in range(0, len(Data[0]), 1):
86              for Row in range(0, len(Data[0][Line]), 1):
87                  Point = [Data[0][Line][Row], Data[1][Line][Row]]
88                  [LinearDist, OnLineDist] = CenterLine.getDistFromLine(Point)
89                  # if the point is inside the average box
90                  if abs(LinearDist * self.RScale) <= MaxDist and\
91                     abs(OnLineDist * self.RScale) <= CenterOffset:
92                      LinDist.append(LinearDist * self.RScale)
93                      Y.append(Data[2][Line][Row])
94          # Sort by increasing distance and bring order to the Universe
95          LinDist, Y = (list(t) for t in zip(*sorted(zip(LinDist, Y))))
96
97          return [LinDist, Y]
98
99      def SelectBoundaries(self, Measurement, MaxDist = 5, CenterOffset = 5):
100         """
101         Lets the user select appropriate boundaries for the linear
102         conversion
103
104         Parameters
105         ----------
106         Measurement : SinglePointMeas object
107                 Contains all raw and result data from a single point
108                 excitation measurement
109         MaxDist : float, default = 5 mm
110             Valid range parallel to the center line in millimeters
111         CenterOffset : float, default = 5 mm
112             Valid range on both sides from the center in millimeters
113         """
114         Measurement.CenterOffset = CenterOffset
115         Measurement.MaxDist = MaxDist
116         CenterLine = Measurement.CenterLine
117
118         def onMouseClick(event):
119             if event.xdata == None:
120                 return True
121             # compute the distance from the line
122             Point = [event.xdata/self.RScale, event.ydata/self.RScale]
123             [NewDist, NewCDist]= Measurement.CenterLine.getDistFromLine(Point)
```

```
124                 print("The New Distance from the line is: {}".format(NewDist/self.RScale))
125                 # check if the CenterOffset or MaxDist changes
126                 if NewDist < (Measurement.MaxDist/3/self.RScale):
127                     # close to the central line, change the CenterOffset
128                      Measurement.CenterOffset = NewCDist*self.RScale
129                      self.Sentinel.info("The New Center Offset is:"+\
130                           " {}".format(Measurement.CenterOffset))
131                 else:
132                      Measurement.MaxDist = NewDist*self.RScale
133                      self.Sentinel.info("The New Max Distance is:"+\
134                                         " {}".format(Measurement.MaxDist))
135
136                 # display the recieved data to the user
137                 self.Sentinel.info("You pressed Key: " + ascii(event.key) +\
138                   " at position in Data Koordinates: " + str([event.xdata, event.ydata]))
139                 # change the rectangle around the center
140                 Rect1.set_width(Measurement.CenterOffset*2)
141                 Rect1.set_height(Measurement.MaxDist*2)
142                 [CornerX, CornerY] = Measurement.getBoundaryCorners()
143                 Rect1.set_xy([CornerX*self.RScale, CornerY*self.RScale])
144                 plt.draw()  # redraw everything
145                 return True
146
147             fig = Measurement.getRawAmplCanvas()
148             # connect the MouseClickEvent with the Canvas
149             fig.canvas.mpl_connect('button_press_event', onMouseClick)
150             ax1 = fig.gca()
151             # necessary for limiting the line range to the image
152             [A, B] = CenterLine.getLineLimits()
153             ax1.plot([A[0]*self.RScale, B[0]*self.RScale],\
154                      [A[1]*self.RScale, B[1]*self.RScale], "w-", label = "Center Line")
155             ax1.plot([CenterLine.CenterPoint[0]*self.RScale],\
156                      [CenterLine.CenterPoint[1]*self.RScale],\
157                      "ro", label = "Center")
158             # plot the rectangle around the center
159             [CornerX, CornerY] = Measurement.getBoundaryCorners()
160             Rect1 = plt.Rectangle([CornerX*self.RScale, CornerY*self.RScale],\
161                                   Measurement.CenterOffset*2,\
162                                   Measurement.MaxDist*2,\
163                                  angle = CenterLine.RotAngle*180/np.pi, color = 'r', fill = False)
164             ax1.add_artist(Rect1)
165             ax1.set_xlim(0, CenterLine.XMax*self.RScale)
166             ax1.set_ylim(0, CenterLine.YMax*self.RScale)
167             plt.show()
168             return True
169
170         def FitAngleTDiff(self, SData):
171             """
172             Fits an Ellipse to the angle dependent thermal diffusivity data.
173
174             Parameters
175             ----------
176                 SData : Sweep Data object
177
178             Returns
179             -------
180                 EllipseData : list of float
181                     Contains the ellipse parameters (a, b, theta)
182             """
183             CList = SData.getCenterLineList()
184             [TDiff, StdTDiff] = SData.getTDiffList()
185             ForwardDir = [CL.HeatFlowAngles[0] for CL in CList]
186             BackwardDir = [CL.HeatFlowAngles[1] for CL in CList]
187             TDiff = np.array(TDiff + TDiff)
188             Direction = np.concatenate((np.array(ForwardDir),\
189                                         np.array(BackwardDir)))
190             if len(TDiff) == 1:
191                 return [TDiff[0], TDiff[0], 0]
```

```
        def f(theta, a, b, phi):
            return a*b/(np.sqrt((a*np.cos(theta - phi))**2 +\
                            (b*np.sin(theta - phi))**2))
        FitData = curve_fit(f, xdata = Direction, ydata = TDiff,\
                             p0 = [np.mean(TDiff), np.mean(TDiff), 0],\
                             bounds = ([0, 0, -np.pi],[100, 100, np.pi]))
        # if b > a
        if FitData[0][1] > FitData[0][0]:
            temp = FitData[0][1]
            FitData[0][1] = FitData[0][0]
            FitData[0][0] = temp
            FitData[0][2] = FitData[0][2] - np.pi/2
            # mirror orientation into [-90, 90] Deg
            if FitData[0][2] < -np.pi/2:
                FitData[0][2] = FitData[0][2] + np.pi
            elif FitData[0][2] > np.pi/2:
                FitData[0][2] = FitData[0][2] - np.pi
        self.Sentinel.info("Ellipse fitted to multi angle line measurement."+\
                           "Parameters (a, b, phi): {}".format(FitData[0]))

        return FitData[0]

    def convertToLinear(self, Measurement, nPixel = 5000):
        """
        Converts a 2D image into linear data by finding the line
        where the line laser hit the sample. The user then selects
        appropriate boundaries for the 2D to linear conversion.
        Each pixel is then asigned the shortest distence to the
        laser line.

        Parameters
        ----------
            Measurement : SinglePointMeas object
                Contains all raw and result data from a single point
                excitation measurement
            nPixel : int, default is 5000
               Number of pixels with the highest intensity
               used for the line detection
        """
        # Find center line
        Measurement.CenterLine.detectLine(Measurement.RawAmplData, nPixel)
        # compute the measurement ID for plots file names
        Measurement.ID = str(Measurement.LockInFreq).replace(".", "_")+\
                         "_Hz_"+\
                         str(int(Measurement.CenterLine.RotAngle*180/np.pi))+\
                         "_DEG"
        # plot the line detection data
        Measurement.CenterLine.plotLaserLineDetection(self.SweepData.PlotPath1D,\
                                                      self.ExpName, Measurement.ID,
                                                      Measurement.LockInFreq)
        # Select the boundaries
        if self.AutoRun == True:
                Measurement.CenterOffset = 5
                Measurement.MaxDist = Measurement.MaxDist*2
        else:
            self.SelectBoundaries(Measurement, Measurement.MaxDist*2)
        # convert the amplitude to distance from center line
        Measurement.LinAmplData = self.CalcLineData(Measurement.RawAmplData,\
                                                    Measurement.CenterLine,\
                                                    Measurement.MaxDist,\
                                                    Measurement.CenterOffset)
        # convert the phase to distance from center line
        [PhaseDist, LinearPhase] = self.CalcLineData(Measurement.RawPhaseData,\
                                                     Measurement.CenterLine,\
                                                     Measurement.MaxDist,\
                                                     Measurement.CenterOffset)
        Measurement.LinPhaseData = np.unwrap([PhaseDist, LinearPhase])
```

```
260         # plot the zoomed amplitude
261         Measurement.plotLineAmplZoom(self.SweepData.PlotPathZoom2D)
262         # plot the zoomed phase
263         Measurement.plotLinePhaseZoom(self.SweepData.PlotPathZoom2D)
264
265     def AnalyzeLineLITMeas(self, RawDataFolder, MinDist, MaxDist):
266         """
267         Analyzes the LIT measurement of a thin film
268         excited with a line source (laser, 3w line heater)
269         at one or multiple frequencies.
270
271         Parameters
272         ----------
273             RawDataFolder : Path object
274                 path to the folder containing all the ASCII raw data files
275             MinDist : float, >0 in [mm]
276                 default minimal distance used for the linear fit
277             MaxDist : float, >0 in [mm]
278                 default maximal distance used for the linear fit
279         """
280         # read in all the files from a sweep
281         self.RawDataFolderPath = RawDataFolder
282         self.SweepData = LIT_SweepData(self.RawDataFolderPath,\
283                                        self.RScale_Instrument,\
284                                        self.ExpName, "Line")
285         for Measurement in self.SweepData.SMeasList:
286             # save the default fit distances
287             Measurement.MinDist = MinDist
288             Measurement.MaxDist = MaxDist
289             # convert the 2D Data to 1D radial data
290             self.convertToLinear(Measurement)
291             # fit the linearized amplitude
292             Measurement.LinAmplData[1] = np.log(Measurement.LinAmplData[1])
293             Measurement.AmplFitData = self.LinFitRegion(Measurement.LinAmplData[0],\
294                                                         Measurement.LinAmplData[1],\
295                                                         "Amplitude", MinDist, MaxDist,\
296                                                         Measurement.LockInFreq)
297             # plot the linear amplitude fit
298             Measurement.plotAmplFit(self.SweepData.PlotPath1D)
299             # fit the linearized phase
300             Measurement.PhaseFitData = self.LinFitRegion(Measurement.LinPhaseData[0],\
301                                                          Measurement.LinPhaseData[1],\
302                                                          "Phase", MinDist, MaxDist,\
303                                                          Measurement.LockInFreq)
304             #plot the linear phase fit
305             Measurement.plotPhaseFit(self.SweepData.PlotPath1D)
306             # compute the thermal diffusivity
307             Measurement.ThermalDiff = abs((np.pi*Measurement.LockInFreq)/\
308                                           (Measurement.AmplFitData[0][0] *\
309                                            Measurement.PhaseFitData[0][0]))
310             Measurement.StdThermalDiff = self.computeSAlpha(Measurement.LockInFreq,\
311                                                             Measurement.AmplFitData,\
312                                                             Measurement.PhaseFitData)
313             self.Sentinel.info("Thermal diffusivity: {} mm^2/s".format(Measurement.ThermalDiff)+\
314                                " for Lock-in Frequency: {} Hz".format(Measurement.LockInFreq))
315         # multiangle  measurement
316         if len(self.SweepData.SMeasList) > 1:
317             self.SweepData.MultiAngle = True
318             # plot the thermal diffusivit vs. direction
319             self.SweepData.FitEllipse = self.FitAngleTDiff(self.SweepData)
320         # save the data
321         self.SweepData.saveData()
322         # plot the sweep results
323         self.SweepData.plotPolarTDiff()
324         self.SweepData.plotCenterLines()
325         self.SweepData.plotAngleTDiff()
326         # create the report
```

```
        self.SweepData.createReport()
        return True
```

A.3.4 Inherited Fiber LIT-Analysis Class

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.colors import LogNorm
from matplotlib.ticker import MaxNLocator
from scipy.optimize import curve_fit
from LIT_SweepData import LIT_SweepData
from LIT_Analyzer import LIT_Analyzer


class LIT_FiberAnalyzer(LIT_Analyzer):
    """Class to analyze LIT measurements of a single fiber"""
    def __init__(self, ExpName, AutoRun):
        """
        Initialize LIT_FiberAnalyzer class

        Parameters
        ----------
        ExpName : str
           Name used for all saved data files
        """
        LIT_Analyzer.__init__(self, ExpName, AutoRun)

    def SelectBoundaries(self, Measurement):
        """
        Shows the amplitude image with the center.
        The user can then select an appropriate maximal distance
        along the fiber.

        Parameters
        ----------
            Measurement : LIT_SinglePointMeas object
                contains all data from a single measurement
        """
        def onMouseClick(event):

            if event.xdata != None:
                # compute the radial distance to the center
                Measurement.MaxConvDist = int(abs(Measurement.Center[1]-\
                                                  event.ydata/self.RScale))
                self.Sentinel.info("New Max Dist selected:"+\
                                   " {}".format(Measurement.MaxDist))
            else:
                return True

            # display the recieved data to the user
            self.Sentinel.info("You pressed Key: " + ascii(event.key) +\
              " at position in Data Koordinates: " + str([event.xdata,\
                                                          event.ydata]))
            # change the position of the two horizontal boundary lines
            TopLine.set_ydata([(Measurement.Center[1] +\
                                Measurement.MaxConvDist)*self.RScale]*2)
            BottomLine.set_ydata([(Measurement.Center[1] -\
                                   Measurement.MaxConvDist)*self.RScale]*2)
            plt.draw()  # redraw everything
            return True

        # default maximal distance:
        Measurement.MaxConvDist = int(min(Measurement.RawAmplData[2].shape)/4)
        fig = Measurement.getRawAmplCanvas()
```

```
61          # connect the MouseClickEvent with the Canvas
62          fig.canvas.mpl_connect('button_press_event', onMouseClick)
63          ax1 = fig.gca()
64          XLimits = ax1.get_xlim()
65          ax1.plot([Measurement.Center[0]*self.RScale],\
66                   [Measurement.Center[1]*self.RScale], "ro")
67          ax1.plot(XLimits, [Measurement.Center[1]*self.RScale]*2, "w-")
68
69          # plot the boundary lines
70          TopLine, = ax1.plot(XLimits, [(Measurement.Center[1] +\
71                              Measurement.MaxConvDist)*self.RScale]*2, "w--")
72          BottomLine, = ax1.plot(XLimits, [(Measurement.Center[1] -\
73                              Measurement.MaxConvDist)*self.RScale]*2, "w--")
74          self.Sentinel.info("Click on the image to"+\
75                             " change the vertical distance")
76          plt.show()
77          return True
78
79      def FindCenterFromAmp(self, AmplData, nPoints = 100):
80          """
81          Determines the center of the fiber excitation.
82          The center is the average of the first nPoints
83          pixel with the largest amplitude.
84
85          Parameters
86          ----------
87          AmplData : numpy array
88              Contains the 2D amplitude image
89          nPoints : int
90              Number of points with the largest amplitude to use
91
92          Returns
93          -------
94          XCenter : int
95              Row index for the pixel with maximal ampltiude
96          YCenter : int
97              Line index for the pixel with maximal amplitude
98          """
99          FlatImage = []    # list with (Ampl, x, y)
100         for a in range(0, len(AmplData[0]), 1):
101             for b in range(0, len(AmplData[0][0]), 1):
102                 FlatImage.append([AmplData[2][a][b],\
103                                   AmplData[0][a][b],\
104                                   AmplData[1][a][b]])
105         FlatImage.sort()  # sort by amplitude
106         XCenter = int(round(np.mean([k[1] for k in FlatImage[-nPoints:]]), 0))
107         YCenter = int(round(np.mean([k[2] for k in FlatImage[-nPoints:]]), 0))
108         self.Sentinel.info("Center detected at: " +\
109                            str([XCenter, YCenter]))
110         return [XCenter, YCenter]
111
112     def getVerticalLines(self, Img, Xmin, Xmax, YCenter, MaxDist):
113         """
114         Extracts the x and y coordinates of the vertical lines
115         between Xmin and Xmax around YCenter +/- MaxDist from the
116         image.
117
118         Parameters
119         ----------
120             Img : Image Data 2D Numpy meshgrid arrays
121             Xmin : int
122                 Horizontal pixel index of the most left line
123             Xmax : int
124                 Horizontal pixel index of the most right line
125             YCenter : int
126                 Y position of the center pixel
127         Raises
128         ------
```

```
            ValueError if XPos is out of image bounds

        Returns
        -------
            YPosValues : list of float
                Pixel indices for the vertical line
            IntesityValues : list of float
                Values of the corresponding pixels
        """
        PointList = []
        for XIdx in range(Xmin, Xmax + 1, 1):
            for k in range(int(YCenter - MaxDist),\
                           int(YCenter + MaxDist)+1, 1):
                PointList.append([abs(k-YCenter)*self.RScale,\
                                  Img[2][k][XIdx]])
        # sort by distance
        PointList.sort()
        YPosValues = [P[0] for P in PointList]
        IntensityValues = [P[1] for P in PointList]
        return [YPosValues, IntensityValues]

    def extractFiberPixels(self, Meas, Width):
        """
        Identifies the vertical pixel line with the highest
        amplitude.

        Parameters
        ----------
            Meas : Single Fiber Measurement Object
                Contains all raw and processed data from a
                single fiber measurement
            Width : int
               Number of vertical pixel lines to extract
               on both sides of the center
        """
        # compute the measurement ID
        Meas.ID = str(Meas.LockInFreq).replace(".","_") + "_Hz"
        # find the maximum amplitude value
        Meas.Center = self.FindCenterFromAmp(Meas.RawAmplData)
        # let the user select the vertical bounadries
        if self.AutoRun == False:
            self.SelectBoundaries(Meas)
        else:
            Measurement.MaxConvDist = Measurement.MaxDist*2

        Xmin = Meas.Center[0] - Width
        Xmax = Meas.Center[0] + Width
        #extract the amplitude lines
        Meas.LinAmplData = self.getVerticalLines(Meas.RawAmplData,\
                                                 Xmin, Xmax,\
                                                 Meas.Center[1],\
                                                 Meas.MaxDist)
        # extract the phase lines
        [PhaseDist, LinearPhase] = self.getVerticalLines(Meas.RawPhaseData,\
                                                          Xmin, Xmax,\
                                                          Meas.Center[1],\
                                                          Meas.MaxDist)
        Meas.LinPhaseData = np.unwrap([PhaseDist, LinearPhase])
        # plot the zoomed amplitude
        Meas.plotFiberAmplZoom(self.SweepData.PlotPathZoom2D)
        # plot the zoomed phase
        Meas.plotFiberPhaseZoom(self.SweepData.PlotPathZoom2D)
        return True

    def AnalyzeFiberLITMeas(self, RawDataFolder, MinDist,\
                            MaxDist, Width = 0):
        """
        Analyzes the LIT measurement of a thin fiber
```

```
197         excited with a point source (line or point laser)
198         at one or multiple frequencies.
199 
200         Parameters
201         ----------
202             RawDataFolder : Path object
203                 path to the folder containing all the ASCII raw data files
204             MinDist : float, >0 in [mm]
205                 default minimal distance used for the linear fit
206             MaxDist : float, >0 in [mm]
207                 default maximal distance used for the linear fit
208             Width : int
209                 Number of vertical pixel lines to extract
210                 left and right off center
211                 default is 0 (only central line)
212         """
213         self.RawDataFolderPath = RawDataFolder
214         self.SweepData = LIT_SweepData(self.RawDataFolderPath,\
215                                        self.RScale_Instrument,\
216                                        self.ExpName, "Fiber")
217         for Measurement in self.SweepData.SMeasList:
218             # save the default fit distances
219             Measurement.MinDist = MinDist
220             Measurement.MaxDist = MaxDist
221             # extract the pixel lines with the fiber
222             self.extractFiberPixels(Measurement, Width)
223             ## fit the linearized amplitude
224             Measurement.LinAmplData[1] = np.log(Measurement.LinAmplData[1])
225             Measurement.AmplFitData = self.LinFitRegion(Measurement.LinAmplData[0],\
226                                                         Measurement.LinAmplData[1],\
227                                                         "Amplitude", MinDist, MaxDist,\
228                                                         Measurement.LockInFreq)
229             # plot the linear amplitude fit
230             Measurement.plotAmplFit(self.SweepData.PlotPath1D)
231             ## fit the linearized phase
232             Measurement.PhaseFitData = self.LinFitRegion(Measurement.LinPhaseData[0],\
233                                                          Measurement.LinPhaseData[1],\
234                                                          "Phase", MinDist, MaxDist,\
235                                                          Measurement.LockInFreq)
236             #plot the linear phase fit
237             Measurement.plotPhaseFit(self.SweepData.PlotPath1D)
238             # compute the thermal diffusivity
239             Measurement.ThermalDiff = abs((np.pi*Measurement.LockInFreq)/\
240                                           (Measurement.AmplFitData[0][0] *\
241                                            Measurement.PhaseFitData[0][0]))
242             Measurement.StdThermalDiff = self.computeSAlpha(Measurement.LockInFreq,\
243                                                             Measurement.AmplFitData,\
244                                                             Measurement.PhaseFitData)
245             self.Sentinel.info("Thermal diffusivity:"+\
246                                " {} mm^2/s".format(Measurement.ThermalDiff)+\
247                                " for Lock-in Frequency:"+\
248                                " {} Hz".format(Measurement.LockInFreq))
249         self.SweepData.AlphaSlope = self.calcAlphaLinReg(self.SweepData)
250         self.SweepData.plotAlphaLinReg()
251         self.Sentinel.info("Thermal diffusivity from the linear regression: "+\
252                            " {} +/- {} mm^2/s".format(self.SweepData.AlphaSlope[0],\
253                                                       self.SweepData.AlphaSlope[1]))
254         self.SweepData.plotTDiffvsFreq()
255         # save the data
256         self.SweepData.saveData()
257         # create the report
258         self.SweepData.createReport()
259         return True
```

A.3.5 Pixel Scale LIT-Analysis Class

```python
import logging
import matplotlib.pyplot as plt
from time import time
import pylatex
from datetime import datetime
import matplotlib.image as mpimg
from matplotlib.widgets import RectangleSelector
import numpy as np


class RScale_Analyzer():
    """ Class to analyze the RScale Image"""

    def __init__(self, ImgPath, LinePeriod, ExpName):
        """
        Create RScale_Analyzer object

        Parameters
        ----------
        ImgPath : Path object
            Path to the image with the reference wafer
        LinePeriod : float
            Periodicity of the lines on the reference wafer in [meters]
        ExpName : str
            Name used for all file saveings
        """
        self.Sentinel = logging.getLogger("Watchtower")
        self.ImgPath = ImgPath
        self.LinePeriod = LinePeriod
        self.ExpName = ExpName
        self.RScale = None
        self.DistList = None
        self.MeanDist = None
        # read in the RScale image
        self.Img = mpimg.imread(ImgPath.absolute())
        # processed data
        self.xAvg = []
        self.IAvg = []
        self.YCrossings = None
        self.ICrossings = None
        self.Middle = None
        self.AvgSelection = None
        # path to the plots
        self.IRImageProcessedPath = None
        self.AvgCrossPlotPath = None

    def AnalyzeRScale(self):
        """
        Determines the pixel size from the reference image.
        The lines on the reference are averaged. The pixel
        size is determined from the known line width and
        pixel number per line.
        """
        # let the user select a profile in the image
        self.SelectAvgRegion()
        # plot the profile and the RScale image with the selected region
        self.plotIRImage()
        self.plotAvgCrossection()
        # save the profile data
        self.saveRScaleResults()

        return self.RScale

    def line_select_callback(self, eclick, erelease):
        """eclick and erelease are the press and release events"""
```

```
66          x1, y1 = eclick.xdata, eclick.ydata
67          x2, y2 = erelease.xdata, erelease.ydata
68          self.AvgSelection = [[int(x1), int(y1)], [int(x2), int(y2)]]
69          # update the profile data
70          self.computeProfile()
71          self.AnalyzeProfile()
72          self.avg_Gray.set_xdata(self.yAvg)
73          self.avg_Gray.set_ydata(self.IAvg)
74          self.avg_line.set_xdata(self.yAvg)
75          self.avg_line.set_ydata(np.ones(len(self.yAvg))*self.Middle)
76          self.avg_crosspt.set_xdata(self.YCrossings)
77          self.avg_crosspt.set_ydata(self.ICrossings)
78          self.avg_ax.set_title("Pixelsize = " + str(round(self.RScale, 4)) +\
79                                " mm")
80          self.avg_ax.relim()
81          # update ax.viewLim using the new dataLim
82          self.avg_ax.autoscale_view()
83          asp = np.diff(self.avg_ax.get_xlim())[0] /\
84                np.diff(self.avg_ax.get_ylim())[0]
85          asp /= np.abs(np.diff(self.image_ax.get_xlim())[0] /\
86                        np.diff(self.image_ax.get_ylim())[0])
87          self.avg_ax.set_aspect(asp)
88          self.Sentinel.info("({}, {})".format(x1, y1)+\
89                             "--> ({}, {})".format(x2, y2))
90          self.Sentinel.info("The button you used was:"+\
91                             " {} {}".format(eclick.button, erelease.button))
92          return True
93
94      def SelectAvgRegion(self):
95          """
96          Plots the RScale Image in a matplotlib figure.
97          The use can select the area to average over by
98          click & release of a rectangle.
99          """
100         # initial selected area
101         XStart = int(self.Img.shape[1]/4)
102         XEnd = int(3*self.Img.shape[1]/4)
103         YStart = int(self.Img.shape[0]/4)
104         YEnd = int(3*self.Img.shape[0]/4)
105         self.AvgSelection = [[XStart, YStart],[XEnd, YEnd]]
106         self.computeProfile()
107         self.Sentinel.info("Computed the horizontaly averaged profile.")
108         self.AnalyzeProfile()
109         self.Sentinel.info("Analyzed the profile to detect the midpoints.")
110         # start the interactive plot
111         fig, (self.image_ax, self.avg_ax) = plt.subplots(1, 2, figsize=(16,9))
112         self.image_ax.imshow(self.Img, aspect="equal")
113         self.avg_Gray, = self.avg_ax.plot(self.yAvg, self.IAvg, "k.-", alpha = 0.5)
114         self.avg_line, = self.avg_ax.plot(self.yAvg,\
115                                           np.ones(len(self.yAvg))*self.Middle, "g-")
116         self.avg_crosspt, = self.avg_ax.plot(self.YCrossings, self.ICrossings, "ro")
117         self.avg_ax.set_xlabel("y-position / pixel")
118         self.avg_ax.set_ylabel("avg. Grayscale Value")
119         self.avg_ax.set_title("Pixelsize = " + str(round(self.RScale, 4)) + " mm")
120         self.image_ax.set_xlabel("x-position / pixel")
121         self.image_ax.set_ylabel("y-position / pixel")
122         self.image_ax.set_title(self.ImgPath)
123         # drawtype is 'box' or 'line' or 'none'
124         RS = RectangleSelector(self.image_ax, self.line_select_callback,
125                                drawtype = 'box', useblit=False,
126                                button=[1, 3],
127                                minspanx=5, minspany=5,
128                                spancoords='pixels',
129                                interactive=True)
130         RS.to_draw.set_visible(True)
131         fig.canvas.draw()
132         asp = np.diff(self.avg_ax.get_xlim())[0] /\
133               np.diff(self.avg_ax.get_ylim())[0]
```

```
        asp /= np.abs(np.diff(self.image_ax.get_xlim())[0] /\
                      np.diff(self.image_ax.get_ylim())[0])
        self.avg_ax.set_aspect(asp)
        # modify the initial position here
        RS.extents = (XStart, XEnd, YStart, YEnd)
        plt.connect('key_press_event', RS)
        plt.tight_layout()
        self.Sentinel.info("Click and relese to select averaging area")
        plt.show()
        return True

    def computeProfile(self):
        """
        Computes a vertical average of the selected region
        """
        self.yAvg = []
        self.IAvg = []
        for yPos in range(self.AvgSelection[0][1], self.AvgSelection[1][1], 1):
            self.yAvg.append(yPos)
            NewAvg = 0
            for xPos in range(self.AvgSelection[0][0], self.AvgSelection[1][0], 1):
                NewAvg += self.Img[yPos][xPos][0]  # last index for grayscale value
            PointN = abs(self.AvgSelection[1][1] - self.AvgSelection[0][1])
            self.IAvg.append(NewAvg / PointN)
        return True

    def AnalyzeProfile(self):
        """
        Computes the pixel size from the 1D profile
        """
        # calculate the middle
        self.Middle = np.mean(self.IAvg)
        # find all crossings of the middle line
        [self.YCrossings, self.ICrossings] =\
                          self.FindCrossings(self.yAvg, self.IAvg, self.Middle)
        # calculate the distances between the crossings
        self.DistList = self.calcCrossDist(self.YCrossings)
        # calculate the average distance
        self.MeanDist = np.mean(self.DistList)
        self.Sentinel.info("The Average distance is: " +\
                           str(round(self.MeanDist, 4)) + " Pixel")
        # calculate the pixel size
        self.RScale = self.LinePeriod / self.MeanDist * 1e3   # mm
        self.Sentinel.info("The Pixel Size is: " +\
                           str(round(self.RScale, 4)) + " mm / Pixel.")
        return True

    def plotIRImage(self):
        """
        Plots the 2D IR Image
        """
        fig, image_ax = plt.subplots(1, 1)
        image_ax.imshow(self.Img, aspect="equal")
        plt.xlabel("x-position / Pixel")
        plt.ylabel("y-position / Pixel")
        plt.title(self.ImgPath)
        # drawtype is 'box' or 'line' or 'none'
        RS = RectangleSelector(image_ax, self.line_select_callback,
                               drawtype = 'box', useblit=False,
                               button=[1, 3],
                               minspanx=5, minspany=5,
                               spancoords='pixels',
                               interactive=False)
        RS.to_draw.set_visible(True)
        fig.canvas.draw()
        RS.extents =  (self.AvgSelection[0][0],\
                       self.AvgSelection[1][0],\
                       self.AvgSelection[0][1],\
```

```
                        self.AvgSelection[1][1])
        plt.tight_layout()
        self.IRImageProcessedPath = self.ImgPath.parent /\
                                    (self.ExpName + "_IRCalibImg.png")
        plt.savefig(str(self.IRImageProcessedPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("IR calibration image plotted and saved")
        return True

    def plotAvgCrossection(self):
        """
        Plots the 1D averaged crossection
        """
        fig = plt.figure()
        plt.plot(self.yAvg, self.IAvg, "k.-", alpha = 0.5)
        plt.plot(self.yAvg, np.ones(len(self.yAvg))*self.Middle, "g-")
        plt.plot(self.YCrossings, self.ICrossings, "ro")
        plt.xlabel("y-position / pixel")
        plt.ylabel("avg. Grayscale Value")
        plt.title("Pixel size = " + str(round(self.RScale, 4)) + " mm")
        plt.tight_layout()
        self.AvgCrossPlotPath = self.ImgPath.parent /\
                                (self.ExpName + "_CalibAvgCross.png")
        plt.savefig(str(self.AvgCrossPlotPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("RScale data plotted and saved")
        return True

    def calcCrossDist(self, Cross):
        """
        Computes the length of a line

        Parameters
        ----------
            Cross : list
                List of all the crossing x-positions

        Returns
        -------
            DistList : list
                List with all the next neighbour distances
        """
        DistList = []
        for Pos in range(1, len(Cross), 1):
            DistList.append(Cross[Pos] - Cross[Pos-1])
        return DistList

    def FindCrossings(self, x, y, Middle):
        """
        Detects the closest pixels to the center line

        Parameters
        ----------
            x,y : list of float
                Averaged horizontal profile of the line image
            Middle : float
                y-value of the central line

        Returns
        -------
            [CrossListX, CrossListY] : list of list of floats
                X and Y coordinates of the pixels at the crossings
        """
        CrossListX = []
        CrossListY = []
        for Pos in range(1, len(x), 1):
            # positive crossing
            if y[Pos-1] < Middle and y[Pos] >= Middle:
```

```
                CrossListX.append((x[Pos-1] + x[Pos])/2)
                CrossListY.append((y[Pos-1] + y[Pos])/2)
                continue
            # negative crossing
            if y[Pos-1] > Middle and y[Pos] <= Middle:
                CrossListX.append((x[Pos-1] + x[Pos])/2)
                CrossListY.append((y[Pos-1] + y[Pos])/2)
                continue

        return [CrossListX, CrossListY]

    def saveRScaleResults(self):
        """
        Saves the results from the RScale measurement to .txt
        """
        ResultFile = open(self.ExpName + "_RScaleResults.txt", "w")
        ResultFile.write("Pixel size for experiment: {}\n".format(self.ExpName))
        TStamp = datetime.fromtimestamp(time()).strftime('%Y-%m-%d %H:%M:%S')
        ResultFile.write("Processed on: {}\n".format(TStamp))
        ResultFile.write("Pixel size analysis from image: " +\
                         str(self.ImgPath.absolute()) + "\n")
        ResultFile.write("Average Pixel Size: " +\
                         str(round(self.RScale, 4)) + " mm / Pixel\n")
        ResultFile.write("Line Width: " + str(round(self.LinePeriod*1e3, 4)) +\
                         " mm\n")
        ResultFile.write("Average Distance: " +\
                         str(round(self.MeanDist, 4)) + " Pixel\n")
        ResultFile.write("# Distances found [Pixel]\n")
        for D in self.DistList:
            ResultFile.write(str(D) + "\n")
        ResultFile.close()
        self.Sentinel.info("RScale Result data saved.")
        return True

    def addLatexReport(self, doc):
        """
        Adds a one page report.

        The report contains the RScale image,
        averaged crossection and processed data

        Paramerets
        ----------
            doc : pylatex document
                Contains all the latex data

        Returns
        -------
            doc : pylatex document
                Contains all the latex data
        """
        # add Heading
        with doc.create(pylatex.MiniPage(align='c')):
            doc.append(pylatex.LargeText(pylatex.utils.bold(self.ExpName)))
            doc.append(pylatex.LineBreak())
            doc.append(pylatex.MediumText(pylatex.utils.bold("Pixel Scale Analysis")))
        # add the 2D IR Calibration Image
        with doc.create(pylatex.Figure(position = 'h!')) as IRImage:
            IRImage.add_image("\"" +\
                              str(self.IRImageProcessedPath.absolute())[:-4] + "\"")
        # add the average IR image crossection
        with doc.create(pylatex.Figure(position = 'h!')) as AvgImage:
            AvgImage.add_image("\"" +\
                               str(self.AvgCrossPlotPath.absolute())[:-4] + "\"")
        # add important fit parameters and results
        with doc.create(pylatex.Center()):
            with doc.create(pylatex.Section('Results', numbering = False)):
                with doc.create(pylatex.Tabular('lcl')) as table:
```

```
                    table.add_hline()
                    table.add_row(("Parameter", "Value", "Unit"))
                    table.add_hline()
                    table.add_row(("Original line period",\
                                   str(self.LinePeriod*1e6), "µm"))
                    table.add_row(("Pixel size",\
                                   str(round(self.RScale, 7)), " mm/Pixel"))
                    table.add_hline()
        doc.append(pylatex.utils.NoEscape(r'\pagebreak'))
        return do
```

A.3.6 Sweep LIT Data Class

```
import logging
from os import listdir
import datetime
import numpy as np
import pylatex
import matplotlib.pyplot as plt
from LIT_SinglePointMeas import LIT_SinglePointMeas
from LIT_SingleLineMeas import LIT_SingleLineMeas
from LIT_SingleFiberMeas import LIT_SingleFiberMeas


class LIT_SweepData():
    """ Class to handle all the data from a LIT sweep"""

    def __init__(self, RawDataFolder, RScaleInstr, ExpName, Type):
        """
        Initialize the LIT sweep data class

        Parameters
        ----------
        RawDataFolder : Path object
            Path to the folder with the amplitude and phase images
        RScaleInstr : RScaleAnalyzer object
            Contains all information about the RScale
        ExpName : str
            Name of the experiment, not the single measurement
        Type : str
            Type of single measurement used: "Point", "Line", "Fiber"
        """
        self.Sentinel = logging.getLogger("Watchtower")
        self.RawDataFolder = RawDataFolder
        self.ExpName = ExpName
        self.Type = Type
        self.RScaleInstrument = RScaleInstr
        self.RScale = RScaleInstr.RScale
        self.MultiFreq = False
        self.MultiAngle = False
        # lists for the sweep data
        self.FreqList = []
        self.AmplSlopeList = []
        self.PhaseSlopeList = []
        self.SMeasList = []
        self.AlphaSlope = None
        self.FitEllipse = None
        # find all amplitude and phase file pairs in the raw data folder
        self.FilePairs = self.getAmplPhaseFilePairs(self.RawDataFolder)
        # read in all the raw data files
        self.readRawData()
        # paths for all the plots
        self.MainFolder = RawDataFolder.resolve().parents[0]
        self.MainPlotFolder = self.MainFolder/"Plots"
        self.MainPlotFolder.mkdir(parents = True, exist_ok = True)
```

```
53          # create a folder for the 1D plots
54          self.PlotPath1D = self.MainPlotFolder/"1DPlots"
55          self.PlotPath1D.mkdir(parents = True, exist_ok = True)
56          # path for the zoomed in 2D amplitude and phase
57          self.PlotPathZoom2D = self.MainPlotFolder/"Zoom2DPlots"
58          self.PlotPathZoom2D.mkdir(parents = True, exist_ok = True)
59          # path for the raw 2D data
60          self.PlotPathRaw2D = self.MainPlotFolder/"Raw2DPlots"
61          self.PlotPathRaw2D.mkdir(parents = True, exist_ok = True)
62          # path for the LaTeX report files
63          self.LatexReportPath = self.MainFolder/"Latex Report"
64          self.LatexReportPath.mkdir(parents = True, exist_ok = True)
65          # sweep plot paths
66          self.AlphaLinRegPath = None
67          self.AlphaFreqPath = None
68          self.PolarPlotPath = None
69          self.CLinePlotPath = None
70          self.AngleTDiffPlotPath = None
71          # plot all the raw data files
72          self.plotRawData()
73
74      def getFreqList(self):
75          """
76          Returns a list of the frequencies for each measurement
77          """
78          self.FreqList = []
79          for M in self.SMeasList:
80              self.FreqList.append(M.LockInFreq)
81          return self.FreqList
82
83      def getCenterLineList(self):
84          """
85          Returns a list with all the center lines
86          """
87          self.CLineList = []
88          for M in self.SMeasList:
89              self.CLineList.append(M.CenterLine)
90          return self.CLineList
91
92      def getPhaseSlopeList(self):
93          """
94          Returns two lists (Value, STD) with the phase slope data
95          """
96          Slopes = []
97          Stds = []
98          for M in self.SMeasList:
99              Slopes.append(M.PhaseFitData[0][0])
100             Stds.append(M.PhaseFitData[1])
101         return [Slopes, Stds]
102
103     def getAmplSlopeList(self):
104         """
105         Returns two lists (Value, STD) with the amplitude slope data
106         """
107         Slopes = []
108         Stds = []
109         for M in self.SMeasList:
110             Slopes.append(M.AmplFitData[0][0])
111             Stds.append(M.AmplFitData[1])
112         return [Slopes, Stds]
113
114     def getTDiffList(self):
115         """
116         Returns two lists with the thermal diffusivity and its error
117         """
118         TDiff = []
119         StdTDiff = []
120         for M in self.SMeasList:
```

```
            TDiff.append(M.ThermalDiff)
            StdTDiff.append(M.StdThermalDiff)
        return [TDiff, StdTDiff]

    def plotAlphaLinReg(self):
        """
        Plots the linear fit for a multi frequency measurement
        """
        [AlphaLinReg, SAlphaLinReg, SlopeProd, SProdError, LinearSlope] = self.AlphaSlope
        fig = plt.figure()
        plt.errorbar(self.FreqList, abs(SlopeProd), fmt = "bo",\
                     yerr = SProdError, markersize = 5, label = "Measured")
        plt.plot([0, max(self.FreqList)], [0, max(self.FreqList) * LinearSlope],\
                 "k-", label = "Linear Fit")
        plt.xlabel("Frequency / Hz")
        plt.ylabel("Slope Product / $1/mm^2$")
        plt.title(r"$\alpha$ = " + str(round(AlphaLinReg, 3)) +\
                  "+/-" + str(round(SAlphaLinReg, 3)) + " $mm^2/s$ (" +\
                  str(round(SAlphaLinReg/AlphaLinReg*100, 2)) + "% rel. std)")
        plt.legend(loc = "upper left", numpoints = 1, frameon = False)
        plt.tight_layout()
        self.AlphaLinRegPath = self.MainFolder / (self.ExpName + "_AlphaSlopeFit.png")
        plt.savefig(str(self.AlphaLinRegPath.absolute()), dpi = 600)
        plt.close()
        return True

    def plotTDiffvsFreq(self):
        FreqList = self.getFreqList()
        [TDiff, StdTDiff] = self.getTDiffList()
        # plot the data and the fit
        fig = plt.figure()
        plt.errorbar(FreqList, TDiff, fmt = "bo",\
                     yerr = StdTDiff, markersize = 5, label = "Measured")
        plt.plot([0, max(FreqList)], [np.mean(TDiff)]*2, "k-", label = "Mean")
        plt.plot([0, max(FreqList)], [np.mean(TDiff) + np.std(TDiff)]*2,\
                 "k--", label = "+/- Sigma")
        plt.plot([0, max(FreqList)], [np.mean(TDiff) - np.std(TDiff)]*2,\
                 "k--")
        plt.ylim(np.mean(TDiff) - 3*np.std(TDiff),\
                 np.mean(TDiff) + 3*np.std(TDiff))
        plt.xlabel("Frequency / Hz")
        plt.ylabel("Thermal Diffusivity / mm^2/s")
        plt.title(r"$\alpha$ = " + str(round(np.mean(TDiff), 3)) +\
                  "+/-" + str(round(np.std(TDiff), 3)) + " $mm^2/s$ (" +\
                  str(round(np.std(TDiff)/np.mean(TDiff)*100, 2)) + "% rel. std)")
        plt.legend(loc = "upper left", numpoints = 1, frameon = False)
        plt.tight_layout()
        self.AlphaFreqPath = self.MainFolder / (self.ExpName + "_AlphavsFreq.png")
        plt.savefig(str(self.AlphaFreqPath.absolute()), dpi = 600)
        plt.close()
        return True

    def plotPolarTDiff(self):
        """
        Plots the thermal diffusivity with error bars direction dependent
        in a polar coordinate system.
        """
        [TDiff, StdTDiff] = self.getTDiffList()
        CLineList = self.getCenterLineList()
        ForwardDir = [CL.HeatFlowAngles[0] for CL in CLineList]
        BackwardDir = [CL.HeatFlowAngles[1] for CL in CLineList]
        theta = np.linspace(-np.pi, np.pi, 1000)
        EllipsePlot =  self.FitEllipse[0] * self.FitEllipse[1]/\
                                (np.sqrt((self.FitEllipse[0]*\
                                 np.cos(theta - self.FitEllipse[2]))**2 +\
                                (self.FitEllipse[1]*\
                                 np.sin(theta - self.FitEllipse[2]))**2))
        # plot in both directions (Angle, Angle + 180Deg)
```

```
        fig = plt.figure()
        ax = plt.subplot(111, projection='polar')
        ax.plot(theta, EllipsePlot, "k--")
        ax.errorbar(ForwardDir, TDiff, yerr = StdTDiff,\
                    fmt = "o", c = "C0", capsize=0, markersize = 10)
        ax.errorbar(BackwardDir, TDiff, yerr = StdTDiff,\
                    fmt = "o", c = "C1", capsize=0, markersize = 10)
        #ax.set_rlabel_position(-22.5)
        ax.set_rmax(max(TDiff)*1.2)
        ax.set_rmin(0)
        ax.set_xticks(np.array([-135, -90, -45, 0, 45, 90, 135, 180])/180*np.pi)
        ax.set_thetalim(-np.pi, np.pi)
        ax.grid(True)
        ax.set_title(self.ExpName + " TDiff / $mm^2/s$", pad = 20)
        #plt.legend(loc = "upper left", numpoints = 1, frameon = False)
        plt.tight_layout()
        self.PolarPlotPath = self.MainFolder / (self.ExpName + "_PolarPlot.png")
        plt.savefig(str(self.PolarPlotPath.absolute()), dpi = 600)
        plt.close()
        return True

    def plotCenterLines(self):
        """
        Plots all center points and lines together to visualize the rotation offset.
        """
        CLineList = self.getCenterLineList()
        Directions = [CL.RotAngle for CL in CLineList]
        CP_X = [CL.CenterPoint[0]*self.RScale for CL in CLineList]
        CP_Y = [CL.CenterPoint[1]*self.RScale for CL in CLineList]
        ArrowLength = 10
        fig = plt.figure()
        plt.plot(CP_X, CP_Y, "ro", label = "Centers")
        plt.plot([np.mean(CP_X)], [np.mean(CP_Y)], "bo", label = "Rotation Center")
        ax = plt.gca()
        ax.set_aspect("equal")

        # plot all the laser lines with 60 pixel length
        for CL in CLineList:
            LMin = np.array(CL.CenterPoint) + 30 * np.array(CL.Direction)
            LMax = np.array(CL.CenterPoint) - 30 * np.array(CL.Direction)
            plt.plot([LMin[0]*self.RScale, LMax[0]*self.RScale],\
                     [LMin[1]*self.RScale, LMax[1]*self.RScale], "k-", lw = 1)
            EndX = ArrowLength*np.cos(CL.HeatFlowAngles[0])
            EndY = ArrowLength*np.sin(CL.HeatFlowAngles[0])
            ax.arrow(CL.CenterPoint[0]*self.RScale, CL.CenterPoint[1]*self.RScale,\
                     EndX*self.RScale, EndY*self.RScale, head_width=2*self.RScale,\
                     head_length=4*self.RScale, fc='k', ec='k')
            EndX = ArrowLength*np.cos(CL.HeatFlowAngles[1])
            EndY = ArrowLength*np.sin(CL.HeatFlowAngles[1])
            ax.arrow(CL.CenterPoint[0]*self.RScale, CL.CenterPoint[1]*self.RScale,\
                     EndX*self.RScale, EndY*self.RScale, head_width=2*self.RScale,\
                     head_length=4*self.RScale, fc='k', ec='k')
        ax.set_xlabel("x-position / mm")
        ax.set_ylabel("y-position / mm")
        # create a square image around the center point
        YLimits = ax.get_ylim()
        XLimits = ax.get_xlim()
        Extent = max([abs(XLimits[0] - np.mean(CP_X)),\
                      abs(XLimits[1] - np.mean(CP_X)),\
                      abs(YLimits[0] - np.mean(CP_Y)),\
                      abs(YLimits[1] - np.mean(CP_Y))])*1.1
        ax.set_xlim([np.mean(CP_X) - Extent,\
                     np.mean(CP_X) + Extent])
        ax.set_ylim([np.mean(CP_Y) - Extent,\
                     np.mean(CP_Y) + Extent])
        #plt.legend(loc = "upper right")
        plt.tight_layout()
        self.CLinePlotPath = self.MainFolder / (self.ExpName + "_CenterLines.png")
```

```
257         plt.savefig(str(self.CLinePlotPath.absolute()), dpi = 600)
258         plt.close()
259         return True
260 
261     def plotAngleTDiff(self):
262         """
263         Plots the thermal diffusiviy vs. the angle in a cartesian coordinate system
264         """
265         [TDiff, StdTDiff] = self.getTDiffList()
266         CLineList = self.getCenterLineList()
267         ForwardDir = [CL.HeatFlowAngles[0]*180/np.pi for CL in CLineList]
268         BackwardDir = [CL.HeatFlowAngles[1]*180/np.pi for CL in CLineList]
269         theta = np.linspace(-np.pi, np.pi, 1000)
270         EllipsePlot =  self.FitEllipse[0] * self.FitEllipse[1]/\
271                                 (np.sqrt((self.FitEllipse[0]*\
272                                  np.cos(theta - self.FitEllipse[2]))**2 +\
273                                (self.FitEllipse[1]*\
274                                  np.sin(theta - self.FitEllipse[2]))**2))
275         fig = plt.figure()
276         plt.plot(theta*180/np.pi, EllipsePlot, "k--", label = "Ellipse Fit")
277         plt.errorbar(ForwardDir, TDiff, yerr = StdTDiff,\
278                      fmt = "o", c = "C0", capsize = 0,\
279                      markersize = 10, label = "Forward")
280         plt.errorbar(BackwardDir, TDiff, yerr = StdTDiff,\
281                      fmt = "o", c = "C1", capsize = 0,\
282                      markersize = 10, label = "Backward")
283         plt.xlabel("Direction / Deg")
284         plt.ylabel("Thermal Diffusivity / $mm^2/s$")
285         plt.ylim(0, max(TDiff) * 1.1)
286         plt.xlim(-180, 180)
287         plt.legend(loc = "lower right")
288         plt.tight_layout()
289         self.AngleTDiffPlotPath = self.MainFolder / (self.ExpName + "_AngleTDiff.png")
290         plt.savefig(str(self.AngleTDiffPlotPath.absolute()), dpi = 600)
291         plt.close()
292         return True
293 
294     def plotRawData(self):
295         """
296         Plots all the single measurements into the
297         PlotPathRaw2D folder
298         """
299         for RawData in self.SMeasList:
300             RawData.plotRawAmplitude(self.PlotPathRaw2D)
301             RawData.plotRawPhase(self.PlotPathRaw2D)
302         return True
303 
304     def getAmplPhaseFilePairs(self, Folder):
305         """
306         Extracts all amplitude and phase file pairs
307         from the specified folder. Amplitude files have _Amp.txt at
308         the end. Phase files have _Phase.txt at the end. Pairs share
309         an identical file name bevore the _Amp.txt or _Phase.txt part.
310 
311         Parameters
312         ----------
313         Folder : Path
314             Path to the folder with all raw data files (.txt)
315 
316         Raises
317         ------
318         ValueError if Folder is not a directory
319 
320         Returns
321         -------
322         FilePairs : list
323             List of all amplitude and phase file pairs
324         """
```

```
325         if Folder.is_dir() == False:
326             ErrorMsg = "The path: {}".format(Folder.absolute())+\
327                        " is not a directory."
328             self.Sentinel.error(ErrorMsg)
329             assert ValueError(ErrorMsg)
330         FilePairs = []
331         # go through all the files
332         for File in Folder.iterdir():
333             # if there is an amplitude file
334             if File.is_file() and File.stem[-4:] == "_Amp":
335                 PhaseFile = Folder/(File.stem[:-4] + "_Phase.txt")
336                 # check if the corresponding phase file exists
337                 if PhaseFile.is_file() == True:
338                     FilePairs.append([File, PhaseFile])
339         self.Sentinel.info("Found {} Amplitude".format(len(FilePairs))+\
340                            " & Phase file pairs.")
341         return FilePairs
342 
343     def readRawData(self):
344         """
345         Reads in all the amplitude and phase file pairs
346         Differentiates between the different measurement types:
347         "Point", "Line", "Fiber"
348 
349         Raises
350         ------
351              ValueError if Type is not in ["Point", "Line", "Fiber"]
352         """
353         # determine the right type of single measurement class
354         if self.Type == "Point":
355             DataClass = LIT_SinglePointMeas
356         elif self.Type == "Line":
357             DataClass = LIT_SingleLineMeas
358         elif self.Type == "Fiber":
359             DataClass = LIT_SingleFiberMeas
360         else:
361             ErrorMsg = "The Type : {}".format(self.Type)+ " is not recognized."+\
362                        "Use one of: [Point, Line, Fiber]"
363             self.Sentinel.Error(ErrorMsg)
364             raise ValueError(ErrorMsg)
365         # read in all file pairs with this measurement class
366         self.SMeasList = []    # empty the single measurement list
367         for FPair in self.FilePairs:
368                 self.SMeasList.append(DataClass(FPair[0], FPair[1],\
369                                                 self.RScale, self.ExpName))
370         self.Sentinel.info("All raw data files of type: {} are loaded.".format(self.Type))
371         return True
372 
373     def saveData(self):
374         """
375         Saves all the results from the analysis into one .txt file.
376         """
377         FilePath = self.MainFolder / (self.ExpName + "_Results.txt")
378         DataFile = open(str(FilePath.absolute()), "w")
379         # write the header
380         DataFile.write("Experiment Name: " + self.ExpName + "\n")
381         DataFile.write("Processed on: {:%d-%m-%Y %H:%M:%S}".format(datetime.datetime.now()) + "\n")
382         DataFile.write("Measurement type: {}\n".format(self.Type))
383         # save the ellipse fit data too
384         if self.Type == "Line":
385             DataFile.write("Major Axis thermal diffusivity (mm^2/s):"+\
386                            " {}\n".format(round(self.FitEllipse[0], 5)))
387             DataFile.write("Minor Axis thermal diffusivity (mm^2/s):"+\
388                            " {}\n".format(round(self.FitEllipse[1], 5)))
389             DataFile.write("Direction of the major axis (degree):"+\
390                            " {}\n".format(round(self.FitEllipse[2]*180/np.pi, 5)))
391         DataFile.write("#Result Data:")
392         DataFile.write("FileName \t LockInFrequency / Hz \t Direction 1 / DEG \t"
```

```
                        "Direction 2 / DEG \t TDiff / mm^2/s \t Sigma TDiff /"
                        " mm^2/s \t LeftPhaseBound / mm \t RightPhaseBound / mm \t "
                        "Phase Slope / rad/mm\t Sigma Phase Slope rad/mm \t Phase y-axis /"
                        " rad \t LeftAmplBound / mm \t RightAmplBound \t "+\
                        "Ampl Slope \t Sigma Ampl Slope \t Ampl y-axis\n")
        for M in self.SMeasList:
            RString = M.AmplPath.stem[:-4] + "\t"
            RString += str(M.LockInFreq) + "\t"
            if self.Type == "Line":
                RString += str(M.CenterLine.RotAngle*180/np.pi-90) + "\t"
                RString += str(M.CenterLine.RotAngle*180/np.pi+90) + "\t"
            else:
                RString += "NA" + "\t"
            RString += str(M.ThermalDiff) + "\t"
            RString += str(M.StdThermalDiff) + "\t"
            RString += str(M.PhaseFitData[2][0][0]) + "\t"
            RString += str(M.PhaseFitData[2][1][0]) + "\t"
            RString += str(M.PhaseFitData[0][0]) + "\t"
            RString += str(M.PhaseFitData[1]) + "\t"
            RString += str(M.PhaseFitData[0][1]) + "\t"
            RString += str(M.AmplFitData[2][0][0]) + "\t"
            RString += str(M.AmplFitData[2][1][0]) + "\t"
            RString += str(M.AmplFitData[0][0]) + "\t"
            RString += str(M.AmplFitData[1]) + "\t"
            RString += str(M.AmplFitData[0][1]) + "\n"
            DataFile.write(RString)
        DataFile.close()
        self.Sentinel.info("Saved all results to: " + str(FilePath))
        return True

    def createReport(self):
        """
        Creates a LaTeX report of all the result and the raw data.
        """
        # define the document
        DocPath = self.LatexReportPath / (self.ExpName + "_Report")
        doc = pylatex.Document(str(DocPath.absolute()),
                        documentclass = "article",
                        document_options = ["a4paper", "pdftex", "12pt"],
                        geometry_options = {"left"   : "3cm",
                                            "right"  : "3cm",
                                            "top"    : "3cm",
                                            "bottom" : "3cm"})
        header = pylatex.PageStyle("header")
        # Create left header
        with header.create(pylatex.Head("L")):
            header.append("University of Bayreuth")
        # Create center header
        with header.create(pylatex.Head("C")):
            header.append("LIT Report")
        # Create right header
        with header.create(pylatex.Head("R")):
            header.append(str(datetime.date.today()))
        # Create center footer
        with header.create(pylatex.Foot("C")):
            header.append(pylatex.simple_page_number())
        doc.preamble.append(header)
        doc.change_document_style("header")
        #create the first page with the main results
        # add Heading
        with doc.create(pylatex.MiniPage(align='c')):
            doc.append(pylatex.LargeText(pylatex.utils.bold("Report for: " + self.ExpName)))
            doc.append(pylatex.LineBreak())
            doc.append(pylatex.LargeText(pylatex.utils.bold("Measurment Type: " + self.Type)))
        if self.Type == "Line":
            with doc.create(pylatex.Figure(position='h!')) as FitPlots:
                with doc.create(pylatex.SubFigure(position='c',
                        width=pylatex.utils.NoEscape(r'0.5\linewidth'))) as FitAmplPlot:
```

```
461                     FitAmplPlot.add_image("\"" + str(self.PolarPlotPath.absolute())[:-4] + "\"",
462                                             width = pylatex.utils.NoEscape(r'\linewidth'))
463                 with doc.create(pylatex.SubFigure(position='c',
464                         width=pylatex.utils.NoEscape(r'0.5\linewidth'))) as FitPhasePlot:
465                     FitPhasePlot.add_image("\"" + str(self.CLinePlotPath.absolute())[:-4] + "\"",
466                                              width = pylatex.utils.NoEscape(r'\linewidth'))
467             with doc.create(pylatex.Figure(position = 'h!')) as LinRegImage:
468                 LinRegImage.add_image("\"" + str(self.AngleTDiffPlotPath.absolute())[:-4] + "\"",\
469                                         width = pylatex.utils.NoEscape(r'0.6\linewidth'))
470             # add important fit parameters and results
471             with doc.create(pylatex.Center()):
472                 with doc.create(pylatex.Section('Results', numbering = False)):
473                     with doc.create(pylatex.Tabular('ccl')) as table:
474                         table.add_hline()
475                         table.add_row(("Frequency / Hz", "Direction / Deg",\
476                                        "Thermal Diffusivity / mm^2/s"))
477                         table.add_hline()
478                         for M in self.SMeasList:
479                             FirstAngle = int(M.CenterLine.HeatFlowAngles[0]*180/np.pi)
480                             OppositeAngle = int(M.CenterLine.HeatFlowAngles[1]*180/np.pi)
481                             Angles = str(FirstAngle) + "°; "+ str(OppositeAngle) + "°"
482                             DataTuple = (str(M.LockInFreq), Angles,\
483                                          str(round(M.ThermalDiff, 5)) + " +/- " +\
484                                          str(round(M.StdThermalDiff, 8)) + " (" +\
485                                          str(round(M.StdThermalDiff/M.ThermalDiff*100,4))+\
486                                          " % rel.)")
487                             table.add_row(DataTuple)
488                         table.add_hline()
489                         table.add_row(("Major Axis / mm^2/s",\
490                                        "Minor Axis / mm^2/s",\
491                                        "Orientation / Deg"))
492                         table.add_hline()
493                         print(self.FitEllipse)
494                         table.add_row((str(round(self.FitEllipse[0], 5)),\
495                                        str(round(self.FitEllipse[1], 5)),\
496                                        str(int(self.FitEllipse[2]*180/np.pi))))
497                         table.add_hline()
498                         table.add_row(("Anisotropy Ratio", "", ""))
499                         table.add_hline()
500                         table.add_row((str(round(self.FitEllipse[0]/\
501                                                  self.FitEllipse[1], 5)), "", ""))
502                         table.add_hline()
503         elif self.Type == "Point":
504             # include the alpha lin fit plot
505             with doc.create(pylatex.Figure(position = 'h!')) as LinRegImage:
506                 LinRegImage.add_image("\"" + str(self.AlphaLinRegPath.absolute())[:-4] + "\"",\
507                                         width = pylatex.utils.NoEscape(r"0.6\textwidth"))
508             with doc.create(pylatex.Figure(position = 'h!')) as AlphaVSFreqImage:
509                 AlphaVSFreqImage.add_image("\"" + str(self.AlphaFreqPath.absolute())[:-4] + "\"",\
510                                         width = pylatex.utils.NoEscape(r"0.6\textwidth"))
511             # add important fit parameters and results
512             with doc.create(pylatex.Center()):
513                 with doc.create(pylatex.Section('Results', numbering = False)):
514                     with doc.create(pylatex.Tabular('cc')) as table:
515                         table.add_hline()
516                         table.add_row(("Frequency / Hz", "Thermal Diffusivity /"+\
517                                        " mm^2/s"))
518                         table.add_hline()
519                         for M in self.SMeasList:
520                             DataTuple = (str(M.LockInFreq),\
521                                          str(round(M.ThermalDiff, 5)) + " +/- " +\
522                                          str(round(M.StdThermalDiff, 8)) + " (" +\
523                                          str(round(M.StdThermalDiff/M.ThermalDiff*100,2))+\
524                                          " % rel.)")
525                             table.add_row(DataTuple)
526                         table.add_hline()
527         elif self.Type == "Fiber":
528             # include the alpha lin fit plot
```

```
            with doc.create(pylatex.Figure(position = 'h!')) as LinRegImage:
                LinRegImage.add_image("\"" + str(self.AlphaLinRegPath.absolute())[:-4] + "\"",\
                                      width = pylatex.utils.NoEscape(r"0.6\textwidth"))
            with doc.create(pylatex.Figure(position = 'h!')) as AlphaVSFreqImage:
                AlphaVSFreqImage.add_image("\"" + str(self.AlphaFreqPath.absolute())[:-4] + "\"",\
                                      width = pylatex.utils.NoEscape(r"0.6\textwidth"))
            # add important fit parameters and results
            with doc.create(pylatex.Center()):
                with doc.create(pylatex.Section('Results', numbering = False)):
                    with doc.create(pylatex.Tabular('cc')) as table:
                        table.add_hline()
                        table.add_row(("Frequency / Hz", "Thermal Diffusivity /"+\
                                       " mm^2/s"))
                        table.add_hline()
                        for M in self.SMeasList:
                            DataTuple = (str(M.LockInFreq),\
                                         str(round(M.ThermalDiff, 5)) + " +/- " +\
                                         str(round(M.StdThermalDiff, 8)) + " (" +\
                                         str(round(M.StdThermalDiff/M.ThermalDiff*100,2))+\
                                         " % rel.)")
                            table.add_row(DataTuple)
                        table.add_hline()
        doc.append(pylatex.utils.NoEscape(r"\pagebreak"))
        # add the pixel calibration procedure
        doc = self.RScaleInstrument.addLatexReport(doc)
        # include the analyzed data from all measurements
        for M in self.SMeasList:
           doc =  M.addLatexReport(doc)
        doc.generate_tex()
        return True
```

A.3.7 Single LIT Measurement Data Class

```
import logging
import numpy as np
import pylatex
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.colors import LogNorm
from matplotlib.ticker import MaxNLocator

class LIT_SingleMeas():
    """ Class to handle all the data from a single measurement"""

    def __init__(self, AmplPath, PhasePath, RScale, ExpName):
        """
        Initialize the SingleFreqData class

        Parameters
        ----------
        AmplPath : Path
            Path to the file containing the 2D amplitude data for the sample
        PhasePath : Path
            Path to the file containing the 2D phase data for the sample
        RScale : float
            Conversion factor from pixel to m in units of [m / Pixel]
        ExpName : str
            Name of the experiment, not the single measurement
        """
        self.Sentinel = logging.getLogger("Watchtower")
        self.AmplPath = AmplPath
        self.PhasePath = PhasePath
        self.ExpName = ExpName
        self.RScale = RScale
        self.MinDist = None     # minimal
```

```
        self.MaxDist = None     # and maximal distance for lin fits
        self.ID = None
        # read in the raw 2Ddata and remove negative amplitudes
        self.RawAmplData = self.getRawDataLIT(self.AmplPath, "Amplitude")
        self.RawPhaseData = self.getRawDataLIT(self.PhasePath, "Phase")
        self.RawAmplData[2] = np.abs(self.RawAmplData[2])
        self.LockInFreq = self.RawAmplData[3]
        # data for the single frequency linear fits
        self.AmplFitData = None
        self.PhaseFitData = None
        self.LinAmplData = None
        self.LinPhaseData = None
        self.ThermalDiff = None
        self.StdThermalDiff = None
        # paths to the different plots
        self.AmplFitPlotPath = None
        self.PhaseFitPlotPath = None
        self.AmplZoomPlotPath = None
        self.PhaseZoomPlotPath = None

    def getRawAmplCanvas(self):
        """
        Creates the figure and axis with the default amplitude
        plot. Can be extended by different other functions later.

        Returns
        -------
            fig : matplotlib.figure object
                Contains the figure with the plot
        """
        fig = plt.figure(tight_layout = True)
        ax1 = plt.subplot(aspect = "equal")
        plot1 = ax1.pcolormesh(self.RawAmplData[0]*self.RScale,\
                               self.RawAmplData[1]*self.RScale,\
                               self.RawAmplData[2],
                               norm=LogNorm(vmin=self.RawAmplData[2].min(),\
                                            vmax=self.RawAmplData[2].max()),\
                               cmap = "viridis")
        ax1.set_title("Amplitude / mK, f = " + str(self.LockInFreq) + " Hz")
        ax1.set_aspect("equal")
        ax1.set_xlabel("x-position / mm")
        ax1.set_ylabel("y-position / mm")
        ax1.yaxis.set_major_locator(MaxNLocator(integer=True))
        ax1.xaxis.set_major_locator(MaxNLocator(integer=True))
        fig.colorbar(plot1, ax = ax1, fraction=0.046, pad=0.04)
        return fig

    def getRawPhaseCanvas(self):
        """
        Creates the figure and axis with the default phase
        plot. Can be extended by different other functions later.

        Returns
        -------
            fig : matplotlib.figure object
                Contains the figure with the plot
        """
        fig = plt.figure(tight_layout = True)
        ax1 = plt.subplot(aspect = "equal")
        plot1 = ax1.pcolormesh(self.RawPhaseData[0]*self.RScale,
                               self.RawPhaseData[1]*self.RScale,\
                               self.RawPhaseData[2]*180/np.pi, cmap = "bwr")
        ax1.set_title("Phase / DEG, f = " + str(self.LockInFreq) + " Hz")
        ax1.set_aspect("equal")
        ax1.set_xlabel("x-position / mm")
        ax1.set_ylabel("y-position / mm")
        ax1.yaxis.set_major_locator(MaxNLocator(integer=True))
        ax1.xaxis.set_major_locator(MaxNLocator(integer=True))
```

```
101         fig.colorbar(plot1, ax = ax1, fraction=0.046, pad=0.04,\
102                      ticks = [-180, -90, 0, 90, 180])
103         return fig
104 
105     def getRawDataLIT(self, Path, Type):
106         """
107         Reads the 2D ASCII images from the files
108 
109         Parameters
110         ----------
111         Path : Path
112             Path to the file containing the 2D data
113         Type : str
114             Either "Amplitude" or "Phase"
115         """
116         DataFile = open(str(Path.absolute()), "r")
117         datalines = DataFile.readlines()
118         DataFile.close()
119         PixelData = []
120         XPixel = []
121         YPixel = []
122         LockInFreq = float(datalines[9][20:-1])
123         # for rad <-> deg conversion
124         ConvFactor = 1
125         if Type == "Phase":
126             ConvFactor = np.pi / 180
127         for L in range(50,len(datalines)-1,1):
128             TempLine = datalines[L].split(";")
129             XLine = []
130             YLine = []
131             PixelLine = []
132             for R in range(0,len(TempLine)-1,1):
133                 PValue = float(TempLine[R])*ConvFactor+0.001
134                 if Type == "Amplitude":
135                     PixelLine.append(max([0.001, PValue]))
136                 else:
137                     PixelLine.append(PValue)
138                 XLine.append(R)
139                 YLine.append(L-50)
140 
141             PixelData.append(PixelLine)
142             XPixel.append(XLine)
143             YPixel.append(YLine)
144 
145         return [np.array(XPixel), np.array(YPixel), np.array(PixelData), LockInFreq]
146 
147     def plotRawAmplitude(self, PlotPathRaw2D):
148         """
149         Plots the 2D amplitude data into the PlotPathRaw2D folder
150 
151         Parameters
152         ----------
153             PlotPathRaw2D : Path object
154                 path to the folder for the raw 2D plots
155         """
156         fig = self.getRawAmplCanvas()
157         PlotPath = PlotPathRaw2D / (str(self.AmplPath.stem[:-4])+\
158                                     "_Raw2DAmpl.png")
159         plt.savefig(str(PlotPath.absolute()), dpi = 600)
160         plt.close()
161         self.Sentinel.info(str(self.AmplPath.stem[:-4])+\
162                            " 2D Raw Amplitude plotted")
163         return True
164 
165     def plotRawPhase(self, PlotPathRaw2D):
166         """
167         Plots the 2D phase data into the PlotPathRaw2D folder
168 
```

```
        Parameters
        ----------
            PlotPathRaw2D : Path object
                path to the folder for the raw 2D plots
        """
        fig = self.getRawPhaseCanvas()
        PlotPath = PlotPathRaw2D / (str(self.PhasePath.stem[:-6])+\
                                    "_Raw2DPhase.png")
        plt.savefig(str(PlotPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info(str(self.PhasePath.stem[:-6])+\
                           " 2D Raw Phase plotted")
        return True

    def plotAmplFit(self, PlotPath1D):
        """
        Plots the linear amplitude fit into the PlotPath1D

        Parameters
        ----------
            PlotPath1D : Path object
                path to the folder for the 1D plots
        """
        # unpack the data
        EndPosList = self.AmplFitData[2]
        Left = EndPosList[0][1]
        Right = EndPosList[1][1]
        LinDist = self.LinAmplData[0]
        LinAmpl = self.LinAmplData[1]
        # plot the linear fit
        fig = plt.figure()
        ax = plt.subplot()
        ax.plot(LinDist, LinAmpl, ".", markersize = 1)
        X_Line = [LinDist[Left], LinDist[Right]]
        Y_Line = self.AmplFitData[0][0] * np.array(X_Line) +\
                 self.AmplFitData[0][1]
        # create two boundary dummy plots where the data is updated later on
        YExtent = ax.get_ylim()
        LeftBound, = ax.plot([LinDist[Left]]*2, YExtent, "k--")
        RightBound, = ax.plot([LinDist[Right]]*2, YExtent, "k--")
        # create a plot for the linear fit
        FitGraph, = ax.plot(X_Line, Y_Line, "r-", lw = 2, label = "Linear Fit")
        ax.set_xlabel("Distance / mm")
        ax.set_ylabel("Linearized Amplitude")
        ax.set_title("Amplitude, f = " + str(self.LockInFreq) + " Hz")
        ax.set_ylim(-2)
        plt.tight_layout()
        self.AmplFitPlotPath = PlotPath1D /\
        (self.ExpName + "_" + self.ID + "_AmplFit.png")
        plt.savefig(str(self.AmplFitPlotPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Linear amplitude fit for: " + self.ExpName +\
                           "_" + self.ID + " plotted.")
        return True

    def plotPhaseFit(self, PlotPath1D):
        """
        Plots the linear phase fit into the PlotPath1D

        Parameters
        ----------
            PlotPath1D : Path object
                path to the folder for the 1D plots
        """
        # unpack the data
        EndPosList = self.PhaseFitData[2]
        Left = EndPosList[0][1]
        Right = EndPosList[1][1]
```

```
237         LinDist = self.LinPhaseData[0]
238         LinPhase = self.LinPhaseData[1]
239         fig = plt.figure()
240         ax = plt.subplot()
241         ax.plot(LinDist, LinPhase, ".", markersize = 1)
242         # find tight y-limits for points between -2pi and 2pi
243         NewYMin = min([P for P in LinPhase if P < 2*np.pi and P > -2*np.pi])
244         NewYMax = max([P for P in LinPhase if P < 2*np.pi and P > -2*np.pi])
245         # plot the linear fit
246         X_Line = [LinDist[Left], LinDist[Right]]
247         Y_Line = self.PhaseFitData[0][0] * np.array(X_Line) +\
248                  self.PhaseFitData[0][1]
249         # create two boundary dummy plots where the data is updated later on
250         YExtent = ax.get_ylim()
251         LeftBound, = ax.plot([LinDist[Left]]*2, YExtent, "k--")
252         RightBound, = ax.plot([LinDist[Right]]*2, YExtent, "k--")
253         # create a plot for the linear fit
254         FitGraph, = ax.plot(X_Line, Y_Line, "r-", lw = 2, label = "Linear Fit")
255         ax.set_xlabel("Distance / mm")
256         ax.set_ylabel("Phase / rad")
257         y_label = [r"$-2\pi$", r"$-\pi$", r"$0$", r"$+\pi$", r"$+2\pi$"]
258         ax.set_yticks([-2*np.pi, -np.pi, 0, np.pi, 2*np.pi])
259         ax.set_yticklabels(y_label, fontsize=20)
260         ax.set_ylim(NewYMin, NewYMax)
261         ax.set_title("Phase, f = " + str(self.LockInFreq) + " Hz")
262         plt.tight_layout()
263         self.PhaseFitPlotPath = PlotPath1D/\
264         (self.ExpName + "_" + self.ID + "_PhaseFit.png")
265         plt.savefig(str(self.PhaseFitPlotPath.absolute()), dpi = 600)
266         plt.close()
267         self.Sentinel.info("Linear phase fit for: " + self.ExpName +\
268                            "_" + self.ID + " plotted.")
269         return True
270 
271     def addLatexReport(self, doc):
272         """
273         Adds a one page report.
274 
275         The report contains the amplitude and phase
276         raw data, the linear fits, and important analysis
277         parameters and results to the provided document doc.
278 
279         Paramerets
280         ----------
281             doc : pylatex document
282                 Contains all the latex data
283 
284         Returns
285         -------
286             doc : pylatex document
287                 Contains all the latex data
288         """
289         # add Heading
290         with doc.create(pylatex.MiniPage(align='c')):
291             doc.append(pylatex.LargeText(pylatex.utils.bold(self.ExpName)))
292             doc.append(pylatex.LineBreak())
293             LatexID =  "f = " + str(round(self.LockInFreq, 3)) + " Hz"
294             if self.Type == "Line":
295                 FirstAngle = int(self.CenterLine.HeatFlowAngles[0]*180/np.pi)
296                 OppositeAngle = int(self.CenterLine.HeatFlowAngles[1]*180/np.pi)
297                 LatexID += ", d = " + str(FirstAngle) + "° and "+ str(OppositeAngle) + "°"
298             doc.append(pylatex.MediumText(pylatex.utils.bold(LatexID)))
299         # add the amplitude and phase 2D raw selection data
300         with doc.create(pylatex.Figure(position='h!')) as ZoomPlots:
301             with doc.create(pylatex.SubFigure(position='b',
302                     width=pylatex.utils.NoEscape(r'0.5\linewidth'))) as ZoomAmplPlot:
303                 ZoomAmplPlot.add_image("\"" + str(self.AmplZoomPlotPath.absolute())[:-4] + "\"",
304                                        width = pylatex.utils.NoEscape(r'\linewidth'))
```

```
            with doc.create(pylatex.SubFigure(position='b',
                    width=pylatex.utils.NoEscape(r'0.5\linewidth'))) as ZoomPhasePlot:
                ZoomPhasePlot.add_image("\"" + str(self.PhaseZoomPlotPath.absolute())[:-4] + "\"",
                                        width = pylatex.utils.NoEscape(r'\linewidth'))
        # add the linearized amplitude and phase fits
        with doc.create(pylatex.Figure(position='h!')) as FitPlots:
            with doc.create(pylatex.SubFigure(position='c',
                    width=pylatex.utils.NoEscape(r'0.5\linewidth'))) as FitAmplPlot:
                FitAmplPlot.add_image("\"" + str(self.AmplFitPlotPath.absolute())[:-4] + "\"",
                                      width = pylatex.utils.NoEscape(r'0.9\linewidth'))
            with doc.create(pylatex.SubFigure(position='c',
                    width=pylatex.utils.NoEscape(r'0.5\linewidth'))) as FitPhasePlot:
                FitPhasePlot.add_image("\"" + str(self.PhaseFitPlotPath.absolute())[:-4] + "\"",
                                        width = pylatex.utils.NoEscape(r'0.9\linewidth'))
        # add important fit parameters and results
        with doc.create(pylatex.Center()):
            with doc.create(pylatex.Section('Parameters and Results', numbering = False)):
                with doc.create(pylatex.Tabular('lcl')) as table:
                    table.add_hline()
                    table.add_row(("Parameter", "Value", "Unit"))
                    table.add_hline()
                    table.add_row(("Frequency", str(self.LockInFreq), "Hz"))
                    FileID = self.AmplPath.stem[:-4].split("Results")[1][1:]
                    table.add_row(("File", FileID, ""))
                    if self.Type == "Line":
                        FirstAngle = int(self.CenterLine.HeatFlowAngles[0]*180/np.pi)
                        OppositeAngle = int(self.CenterLine.HeatFlowAngles[1]*180/np.pi)
                        Angles = str(FirstAngle) + "°; "+ str(OppositeAngle) + "°"
                        table.add_row(("Direction", Angles, "Deg"))
                    table.add_row(("Pixel Size", str(round(self.RScale, 7)), "mm/Pixel"))
                    table.add_row(("Amplitude Slope", str(round(self.AmplFitData[0][0], 4)) +\
                                   " +/- " + str(round(self.AmplFitData[1], 5)), "1/mm"))
                    table.add_row(("Phase Slope", str(round(self.PhaseFitData[0][0], 4))+\
                                   " +/- " + str(round(self.PhaseFitData[1], 5)), "1/mm"))
                    table.add_row(("Left Fit Boundary Amplitude",\
                                   str(round(self.AmplFitData[2][0][0], 2)), "mm"))
                    table.add_row(("Right Fit Boundary Amplitude",\
                                   str(round(self.AmplFitData[2][1][0], 2)), "mm"))
                    table.add_row(("Left Fit Boundary Phase",\
                                   str(round(self.PhaseFitData[2][0][0], 2)), "mm"))
                    table.add_row(("Right Fit Boundary Phase",\
                                   str(round(self.PhaseFitData[2][1][0], 2)), "mm"))
                    table.add_row(("Thermal Diffusivity",\
                                   str(round(self.ThermalDiff, 5)) + " +/- " +\
                                   str(round(self.StdThermalDiff, 8)),\
                                   "mm^2/s"))
                    table.add_row(("Thermal Wavelength (TPD)",\
                                   str(int(np.sqrt(self.ThermalDiff*1e6/self.LockInFreq/np.pi))),\
                                   "µm"))
                    table.add_hline()
        doc.append(pylatex.utils.NoEscape(r'\pagebreak'))
        return doc
```

A.3.8 Inherited LIT Point Measurement Data Class

```
import logging
from LIT_SingleMeas import LIT_SingleMeas
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.colors import LogNorm
from matplotlib.ticker import MaxNLocator

class LIT_SinglePointMeas(LIT_SingleMeas):
    """ Class to handle all the data from a single
```

```
11          point excitation measurement"""
12
13      def __init__(self, AmplPath, PhasePath, RScale, ExpName):
14          """
15          Initialize the SinglePointMeas class
16
17          Parameters
18          ----------
19          AmplPath : Path
20              Path to the file containing the 2D amplitude data for the sample
21          PhasePath : Path
22              Path to the file containing the 2D phase data for the sample
23          RScale : float
24              Conversion factor from pixel to m in units of [m / Pixel]
25          ExpName : str
26              Name of the experiment, not the single measurement
27          """
28          self.Sentinel = logging.getLogger("Watchtower")
29          LIT_SingleMeas.__init__(self, AmplPath, PhasePath, RScale, ExpName)
30          self.Center = None    # Contains the data of the center pixel
31          self.MaxR = None
32          AmplRadii = None
33          RadialAmpl = None
34          PhaseRadii = None
35          RadialPhase = None
36          self.Type = "Point"
37
38      def PlotRadialAmplZoom(self, PlotPathZoom2D):
39          """
40          Plots the amplitude with the center and the circular area
41          used for averaging. The image is centered and zoomed around
42          the averaging area.
43
44          Parameters
45          ----------
46              PlotPathZoom2D : Path object
47                  Path to the folder for the zoomed 2d plots
48          """
49          fig = self.getRawAmplCanvas()
50          ax1 = fig.gca()
51          ax1.plot([self.Center[0]*self.RScale],\
52                   [self.Center[1]*self.RScale], "ro")
53          S = self.RawPhaseData[2].shape
54          Left = max([(self.Center[0] - self.MaxR*1.3), 0])
55          Right = min([(self.Center[0] + self.MaxR*1.3), S[1]])
56          Bottom = max([(self.Center[1] - self.MaxR*1.3), 0])
57          Top = min([(self.Center[1] + self.MaxR*1.3), S[0]])
58          ax1.set_xlim(Left*self.RScale, Right*self.RScale)
59          ax1.set_ylim(Bottom*self.RScale, Top*self.RScale)
60          # plot the circle around the center
61          circle1 = plt.Circle(np.array(self.Center[0:2])*self.RScale,
62                               self.MaxR*self.RScale,\
63                               color = 'r', fill = False)
64          ax1.add_artist(circle1)
65          plt.tight_layout()
66          self.AmplZoomPlotPath = PlotPathZoom2D /\
67          (self.ExpName + "_" + self.ID + "_AmplZoom.png")
68          plt.savefig(str(self.AmplZoomPlotPath.absolute()), dpi = 600)
69          plt.close()
70          self.Sentinel.info("Zoomed 2D Amplitude for {} ".format(self.ExpName) +\
71                             " at f = {}".format(round(self.LockInFreq, 3)) + " plotted" )
72          return True
73
74      def PlotRadialPhaseZoom(self, PlotPathZoom2D):
75          """
76          Plots the phase with the center and the circular area
77          used for averaging. The image is centered and zoomed around
78          the averaging area.
```

```
        Parameters
        ----------
            PlotPathZoom2D : Path object
                Path to the folder for the zoomed 2d plots
        """
        fig = self.getRawPhaseCanvas()
        ax1 = fig.gca()
        ax1.plot([self.Center[0]*self.RScale],\
                 [self.Center[1]*self.RScale], "ro")
        S = self.RawPhaseData[2].shape
        Left = max([(self.Center[0] - self.MaxR*1.3), 0])
        Right = min([(self.Center[0] + self.MaxR*1.3), S[1]])
        Bottom = max([(self.Center[1] - self.MaxR*1.3), 0])
        Top = min([(self.Center[1] + self.MaxR*1.3), S[0]])
        ax1.set_xlim(Left*self.RScale, Right*self.RScale)
        ax1.set_ylim(Bottom*self.RScale, Top*self.RScale)
        # plot the circle around the center
        circle1 = plt.Circle(np.array(self.Center[0:2])*self.RScale,
                             self.MaxR*self.RScale,\
                             color = 'r', fill = False)
        ax1.add_artist(circle1)
        plt.tight_layout()
        self.PhaseZoomPlotPath = PlotPathZoom2D /\
        (self.ExpName + "_" + self.ID + "_PhaseZoom.png")
        plt.savefig(str(self.PhaseZoomPlotPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Zoomed 2D Phase for {} ".format(self.ExpName) +\
                           " at f = {}".format(round(self.LockInFreq, 3)) + " plotted" )
        return True
```

A.3.9 Inherited LIT Line Measurement Data Class

```
import logging
from LIT_SingleMeas import LIT_SingleMeas
from LIT_LaserLine import LIT_LaserLine
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.colors import LogNorm
from matplotlib.ticker import MaxNLocator


class LIT_SingleLineMeas(LIT_SingleMeas):
    """ Class to handle all the data from a single
        line excitation measurement"""

    def __init__(self, AmplPath, PhasePath, RScale, ExpName):
        """
        Initialize the SingleLineMeas class

        Parameters
        ----------
        AmplPath : Path
            Path to the file containing the 2D amplitude data for the sample
        PhasePath : Path
            Path to the file containing the 2D phase data for the sample
        RScale : float
            Conversion factor from pixel to m in units of [m / Pixel]
        ExpName : str
            Name of the experiment, not the single measurement
        """
        self.Sentinel = logging.getLogger("Watchtower")
        LIT_SingleMeas.__init__(self, AmplPath, PhasePath, RScale, ExpName)
        self.CenterLine = LIT_LaserLine()
```

```
33          self.CenterOffset = 1
34          self.MaxDist = 1
35          self.Type = "Line"
36
37      def getBoundaryCorners(self):
38          """
39          Computes the cornes for the boundary rectangle
40
41          Returns
42          -------
43             [CornerX, CornerY] : list of float
44                 [(x,y), (x,y)] points for the corners of the rectangle
45          """
46          CornerX = self.CenterLine.CenterPoint[0]-\
47                    (np.cos(self.CenterLine.RotAngle)*self.CenterOffset/self.RScale -\
48                     np.sin(self.CenterLine.RotAngle)*self.MaxDist/self.RScale)
49          CornerY = self.CenterLine.CenterPoint[1]-\
50                    (np.sin(self.CenterLine.RotAngle)*self.CenterOffset/self.RScale +\
51                     np.cos(self.CenterLine.RotAngle)*self.MaxDist/self.RScale)
52          self.Sentinel.debug("New boundary corners: {}, {}".format(CornerX, CornerY))
53          return [CornerX, CornerY]
54
55      def plotLineAmplZoom(self, PlotPathZoom2D):
56          """
57          Plots a zoom of the selected linear conversion amplitude region.
58          The location of the line laser is overlayed.
59
60          Parameters
61          ----------
62              PlotPathZoom2D : Path object
63                  Path to the folder for the zoomed 2D plots
64          """
65          fig = self.getRawAmplCanvas()
66          ax1 = fig.gca()
67          # plot the center line into the image
68          [A, B] = self.CenterLine.getLineLimits()
69          ax1.plot([A[0]*self.RScale, B[0]*self.RScale],\
70                   [A[1]*self.RScale, B[1]*self.RScale], "w-", label = "Center Line")
71          ax1.plot([self.CenterLine.CenterPoint[0]*self.RScale],\
72                   [self.CenterLine.CenterPoint[1]*self.RScale],\
73                   "ro", label = "Center")
74          # plot the rectangle around the center
75          [CornerX, CornerY] = self.getBoundaryCorners()
76          Rect1 = plt.Rectangle([CornerX*self.RScale, CornerY*self.RScale], self.CenterOffset*2,\
77                                self.MaxDist*2,\
78                                angle = self.CenterLine.RotAngle*180/np.pi,\
79                                color = 'r', fill = False)
80          ax1.add_artist(Rect1)
81          ax1.set_xlim(0, self.CenterLine.XMax*self.RScale)
82          ax1.set_ylim(0, self.CenterLine.YMax*self.RScale)
83          self.AmplZoomPlotPath = PlotPathZoom2D /\
84          (self.ExpName + "_" + self.ID + "_AmplZoom.png")
85          plt.savefig(str(self.AmplZoomPlotPath.absolute()), dpi = 600)
86          plt.close()
87          self.Sentinel.info("Zoomed 2D Amplitude for {} ".format(self.ExpName) +\
88                             " at f = {}".format(round(self.LockInFreq, 3)) + " plotted")
89          return True
90
91      def plotLinePhaseZoom(self, PlotPathZoom2D):
92          """
93          Plots a zoom of the selected linear conversion phase region.
94          The location of the line laser is overlayed.
95
96          Parameters
97          ----------
98              PlotPathZoom2D : Path object
99                  Path to the folder for the zoomed 2D plots
100         """
```

```
        fig = self.getRawPhaseCanvas()
        ax1 = fig.gca()
        # necessary for limiting the line range to the image
        [A, B] = self.CenterLine.getLineLimits()
        ax1.plot([A[0]*self.RScale, B[0]*self.RScale],\
                 [A[1]*self.RScale, B[1]*self.RScale], "w-", label = "Center Line")
        ax1.plot([self.CenterLine.CenterPoint[0]*self.RScale],\
                 [self.CenterLine.CenterPoint[1]*self.RScale], "ro", label = "Center")
        # plot the rectangle around the center
        [CornerX, CornerY] = self.getBoundaryCorners()
        Rect1 = plt.Rectangle([CornerX*self.RScale, CornerY*self.RScale], self.CenterOffset*2,\
                              self.MaxDist*2,\
                              angle = self.CenterLine.RotAngle*180/np.pi,\
                              color = 'r', fill = False)
        ax1.add_artist(Rect1)
        ax1.set_xlim(0, self.CenterLine.XMax*self.RScale)
        ax1.set_ylim(0, self.CenterLine.YMax*self.RScale)
        self.PhaseZoomPlotPath = PlotPathZoom2D /\
        (self.ExpName + "_" + self.ID + "DEG_PhaseZoom.png")
        plt.savefig(str(self.PhaseZoomPlotPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Zoomed 2D Phase for {} ".format(self.ExpName) +\
                           " at f = {}".format(round(self.LockInFreq, 3)) + " plotted")
        return True
```

A.3.10 Inherited LIT Fiber Measurement Data Class

```
import logging
from LIT_SingleMeas import LIT_SingleMeas
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.colors import LogNorm
from matplotlib.ticker import MaxNLocator

class LIT_SingleFiberMeas(LIT_SingleMeas):
    """ Class to handle all the data from a single
       fiber measurement"""

    def __init__(self, AmplPath, PhasePath, RScale, ExpName):
        """
        Initialize the SingleFiberMeas class

        Parameters
        ----------
        AmplPath : Path
            Path to the file containing the 2D amplitude data for the sample
        PhasePath : Path
            Path to the file containing the 2D phase data for the sample
        RScale : float
            Conversion factor from pixel to m in units of [m / Pixel]
        ExpName : str
            Name of the experiment, not the single measurement
        """
        self.Sentinel = logging.getLogger("Watchtower")
        LIT_SingleMeas.__init__(self, AmplPath, PhasePath, RScale, ExpName)
        self.Type = "Fiber"
        self.Center = []
        self.MaxDist = None
        self.MaxConvDist = None

    def plotFiberAmplZoom(self, PlotPathZoom2D):
        """
        Plots a zoom of the selected pixel line amplitude region.
        The location of the line laser is overlayed.

```

```
        Parameters
        ----------
            PlotPathZoom2D : Path object
                Path to the folder for the zoomed 2D plots
        """
        fig = self.getRawAmplCanvas()
        ax1 = fig.gca()
        XLimits = ax1.get_xlim()
        # plot the center
        ax1.plot([self.Center[0]*self.RScale],\
                 [self.Center[1]*self.RScale], "ro", label = "Center")
        ax1.plot(XLimits, [self.Center[1]*self.RScale]*2, "w-")
        # plot the boundary lines
        ax1.plot(XLimits, [(self.Center[1] + self.MaxDist)*self.RScale]*2, "w--")
        ax1.plot(XLimits, [(self.Center[1] - self.MaxDist)*self.RScale]*2, "w--")
        ax1.set_xlim((self.Center[0] - self.MaxDist*1.2)*self.RScale,\
                     (self.Center[0] + self.MaxDist*1.2)*self.RScale)
        ax1.set_ylim((self.Center[1] - self.MaxDist*1.2)*self.RScale,\
                     (self.Center[1] + self.MaxDist*1.2)*self.RScale)
        self.AmplZoomPlotPath = PlotPathZoom2D /\
        (self.ExpName + "_" + self.ID + "_AmplZoom.png")
        plt.savefig(str(self.AmplZoomPlotPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Zoomed 2D Amplitude for {} ".format(self.ExpName) +\
                           " at f = {}".format(round(self.LockInFreq, 3)) + " plotted")
        return True

    def plotFiberPhaseZoom(self, PlotPathZoom2D):
        """
        Plots a zoom of the selected pixel line phase region.
        The location of the line laser is overlayed.

        Parameters
        ----------
            PlotPathZoom2D : Path object
                Path to the folder for the zoomed 2D plots
        """
        fig = self.getRawPhaseCanvas()
        ax1 = fig.gca()
        XLimits = ax1.get_xlim()
        # plot the center
        ax1.plot([self.Center[0]*self.RScale],\
                 [self.Center[1]*self.RScale], "ro", label = "Center")
        ax1.plot(XLimits, [self.Center[1]*self.RScale]*2, "w-")
        # plot the boundary lines
        ax1.plot(XLimits, [(self.Center[1] + self.MaxDist)*self.RScale]*2, "w--")
        ax1.plot(XLimits, [(self.Center[1] - self.MaxDist)*self.RScale]*2, "w--")
        ax1.set_xlim((self.Center[0] - self.MaxDist*1.2)*self.RScale,\
                     (self.Center[0] + self.MaxDist*1.2)*self.RScale)
        ax1.set_ylim((self.Center[1] - self.MaxDist*1.2)*self.RScale,\
                     (self.Center[1] + self.MaxDist*1.2)*self.RScale)
        self.PhaseZoomPlotPath = PlotPathZoom2D /\
        (self.ExpName + "_" + self.ID + "DEG_PhaseZoom.png")
        plt.savefig(str(self.PhaseZoomPlotPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Zoomed 2D Phase for {} ".format(self.ExpName) +\
                           " at f = {}".format(round(self.LockInFreq, 3)) + " plotted")
        return True
```

A.3.11 LIT Laser Line Class

```
import logging
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
```

```
5   from matplotlib.colors import LogNorm
6   from matplotlib.ticker import MaxNLocator
7
8
9   class LIT_LaserLine:
10      """ Manages data from the line laser location """
11
12      def __init__(self):
13          self.CenterPoint = [0,0]
14          self.Direction = [1,1]
15          self.RotAngle = 0
16          self.HeatFlowAngles = [0, np.pi]
17          self.PointList = []
18          self.LineFit = []
19          self.nPixel = 0
20          self.Image = []
21          self.Sentinel = logging.getLogger("Watchtower")
22
23      def getLineLimits(self):
24          """
25          Computes the boundary points for the laser line at
26          the edges of the image
27
28          Return
29          ------
30             [A,B] : list of [x,y] points in pixel
31                first and last point on the laser line in the image
32          """
33          LamTop = min([self.CenterPoint[0]/self.Direction[0],\
34                        self.CenterPoint[1]/self.Direction[1]])
35          LamBottom = min([(self.XMax - self.CenterPoint[0])/self.Direction[0],\
36                           (self.YMax - self.CenterPoint[1])/self.Direction[1]])
37          A = np.array(self.CenterPoint) - LamTop * np.array(self.Direction)
38          B = np.array(self.CenterPoint) + LamBottom * np.array(self.Direction)
39          self.Sentinel.debug("New center line limits computed: {}".format([A, B]))
40          return [A, B]
41
42      def getDistFromLine(self, Point):
43          """
44          Computes the distance of the Point to the Line
45
46          The line is defined in vector form as
47          starting point + normalized direction vector
48          The point is given by its (x,y) coordinates
49
50          Parameters
51          ----------
52          Point : (x,y) float
53              X and Y coordinates of the point
54
55          Returns
56          -------
57          LinDist : float
58              shortest distance between the point and the line
59          CenterDist : float
60              distance from the closest intersection on the line to the center
61          """
62          P1 = self.CenterPoint
63          P2 = [self.CenterPoint[0] + self.Direction[0],\
64                self.CenterPoint[1] + self.Direction[1]]
65          Dist = abs(self.Direction[1]*Point[0]-self.Direction[0]*Point[1] +\
66                     P2[0]*P1[1] - P2[1]*P1[0])
67          CenterDist = np.sqrt(abs((self.CenterPoint[0]-Point[0])**2 +\
68                                   (self.CenterPoint[1]-Point[1])**2 -\
69                                    Dist**2))
70          return [Dist, CenterDist]
71
72      def getMaxPoints(self, Image, n):
```

```
        """
        Extracts the first n pixels with the highest amplitude value

        Parameters
        ----------
        Image : list of 2D np.arrays [X, Y, Ampl]
        n : int
            Number of points to use for the analysis

        Returns
        -------
        [XList, YList] : list of [x] and [y] points
        """
        FlatImage = []    # list with (Ampl, x, y)
        for a in range(0, len(Image[0]), 1):
            for b in range(0, len(Image[0][0]), 1):
                FlatImage.append([Image[2][a][b],\
                                  Image[0][a][b],\
                                  Image[1][a][b]])
        FlatImage.sort()  # sort by amplitude
        XList = [k[1] for k in FlatImage[-n:]]
        YList = [k[2] for k in FlatImage[-n:]]
        return [XList, YList]

    def detectLine(self, Image, n):
        """
        Determines the center line for a line excitation measurement

        n points with the highest amplitude are used to determine the
        line. The center is the average of all points. The line is
        determined by a linear fit to the (x,y) data.

        Parameters
        ----------
        n : int
            Number of points to use for the analysis
        Image : list of 2D np.arrays [X, Y, Ampl]
        """
        self.nPixel = n
        self.Image = Image
        self.XMax = Image[0][0][-1]
        self.YMax = Image[1][-1][0]
        # extract the first n points with the highest amplitude
        self.PointList = self.getMaxPoints(Image, n)
        # the center is the mean of all pixels
        self.CenterPoint = [np.mean(self.PointList[0]),\
                            np.mean(self.PointList[1])]
        # the slope comes from a linear fit
        self.LineFit = np.polyfit(self.PointList[0], self.PointList[1], 1)
        Length = np.sqrt(1+self.LineFit[0]*self.LineFit[0])  # sqrt(1+m^2)
        self.Direction = [1/Length, self.LineFit[0]/Length]
        self.RotAngle = np.arctan2(self.Direction[1], self.Direction[0])
        self.HeatFlowAngles = [self.RotAngle+np.pi/2, self.RotAngle-np.pi/2]
        self.Sentinel.info("Center line detected. Center at: {}".format(self.CenterPoint)+\
                           " Rotation angle: {}".format(self.RotAngle*180/np.pi))
        return True

    def plotLaserLineDetection(self, PlotPath1D, ExpName, ID, LockInFreq):
        """
        Plots the linear fit of the pixels with the highest amplitudes.
        The center and the line are overlayed.

        Parameters
        ----------
            PlotPath1D : Path object
                Path to the folder for the plot
            ExpName : str
                Name for the experiment
```

```
            ID : str
                ID for the current measurement
            LockInFreq : float
                Frequency of the lock in detection
        """
        xLine = np.linspace(min(self.PointList[0]),\
                            max(self.PointList[0]), 10)
        yLine = self.LineFit[0] * xLine + self.LineFit[1]
        fig = plt.figure()
        ax = plt.subplot()
        ax.plot(self.PointList[0], self.PointList[1], "k.",\
                label = "Threshold Pixel")
        ax.plot(xLine, yLine, "-", lw = 2, label = "Fit Line")
        ax.plot([self.CenterPoint[0]], [self.CenterPoint[1]],\
                "ro", label = "Center")
        self.Image
        ax.set_title("Amplitude, f = " + str(LockInFreq) + " Hz")
        ax.set_xlabel("horizontal position / Pixel")
        ax.set_ylabel("vertical position / Pixel")
        ax.set_xlim(0, self.XMax)
        ax.set_ylim(0, self.YMax)
        PlotPath = PlotPath1D /\
        (ExpName + "_" + ID + "_AmplLineFit.png")
        plt.savefig(str(PlotPath.absolute()), dpi = 600)
        plt.close()
        self.Sentinel.info("Laser Line detection plotted for measurement: "+\
                           ExpName + "_" + ID)
        return True
```

A.3.12 Rigol Arbitrary Signal Generation

```
from matplotlib import pylab as plt
import numpy as np

"""
Simple program to generate a RAF file comprised of the sum of a set
of sinusoids.
"""
# DAHBORAD
ExpName = "BK185"
PointN = 2**12      # number of support points
FreqList = [0.2, 0.4, 0.6, 0.8, 1] # Hz
AmplList = [np.sqrt(FreqList[0]/f) for f in FreqList]
PhaseList = [0]*len(FreqList)

# Setting Plot Parameters
params = {'axes.labelsize': 16,
          'font.size': 16,
          'font.family': "arial",
          'legend.fontsize': 16,
          'xtick.labelsize': 16,
          'ytick.labelsize': 16,
          'text.usetex': False,
          'figure.figsize': [4.5 * 1.4, 4.5]}
plt.rcParams.update(params)

def plotSignal(Signal, ExpName):
    fig = plt.figure()
    plt.plot(np.r_[0:len(Signal)], Signal, ".")
    plt.xlabel("Sample ID")
    plt.ylabel("Signal")
    plt.title(ExpName)
    plt.tight_layout()
    plt.savefig(ExpName + "_Signal.png", dpi = 600)
    plt.savefig(ExpName + "_Signal.eps")
```

```
35      plt.show()
36
37      return True
38
39  def to_raf(Signal, FileName):
40      """
41      Convert the signal (already the correct length)
42      to a RAF file.
43      """
44      # Shift and convert the signal to the range (-1, 1)
45      Signal = Signal - Signal.min()
46      Signal = Signal/Signal.max()
47      # plot the signal
48      plotSignal(Signal, FileName)
49      # convert to 16 bit integers
50      Signal = (Signal* int("3fff", 16)).astype('int16')
51      # Write the signal as binary.
52      fp = open(FileName + ".RAF", "w")
53      Signal.tofile(fp)
54
55      return True
56
57  def SumSignal(t, PhaseList, FreqList, AmplList):
58      I = 0
59      for k in range(0, len(FreqList), 1):
60        I += AmplList[k] * np.cos(2*np.pi*FreqList[k]*t + PhaseList[k])
61
62      return I
63
64  #MAIN
65  t = np.r_[0:PointN]
66  Signal = SumSignal(t/PointN/min(FreqList), PhaseList, FreqList, AmplList)
67  Signal = np.array(Signal, dtype=float)
68  to_raf(Signal, ExpName)
```

A.4 Particle Sizer

A.4.1 Main GUI

```
function varargout = GUI_ParticleSizer(varargin)
% GUI_PARTICLESIZER MATLAB code for GUI_ParticleSizer.fig
%      GUI_PARTICLESIZER, by itself, creates a new GUI_PARTICLESIZER or raises the existing
%      singleton*.
%
%      H = GUI_PARTICLESIZER returns the handle to a new GUI_PARTICLESIZER or the handle to
%      the existing singleton*.
%
%      GUI_PARTICLESIZER('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_PARTICLESIZER.M with the given input arguments.
%
%      GUI_PARTICLESIZER('Property','Value',...) creates a new GUI_PARTICLESIZER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_ParticleSizer_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_ParticleSizer_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_ParticleSizer

% Last Modified by GUIDE v2.5 31-Jan-2017 15:06:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_ParticleSizer_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_ParticleSizer_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%CHECKED
function GUI_ParticleSizer_OpeningFcn(hObject, ~, handles, varargin)
%clc;
format long;
movegui('center');  % move the GUI to the center of the Screen upon start
% Choose default command line output for GUI_ParticleSizer
handles.output = hObject;
handles.CurrentSelectedFile = 1; % File last selected in the Table
handles.ResultFileName = 'EnterFileName';  % Name of the Result File
handles.NewFolderSelected = false; % Is a new batch of files loaded?
handles.LastSelectedFolder = ''; % which was the last folder selected?
handles.ResultData = {};
%%% SaveFile, FileName, ParticlesFound, Avg.Diameter
handles.FileTable = {};
% Update handles structure
guidata(hObject, handles);
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = GUI_ParticleSizer_OutputFcn(~, ~, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CHECKED
function FolderSelect_Callback(hObject, ~, handles) %#ok<DEFNU>
% if a folder was selected previously, clear all data
if  isempty(handles.FileTable) == false
    handles.ResultData = {};
    handles.FileTable = {};
    guidata(hObject, handles);
end
% open the UI for folder Selection and get a folder from the User
SelectedFolder = uigetdir(handles.LastSelectedFolder);
% get the pixel size from the edit field
PixelSizeFieldEntry = get(handles.PixelSizeEntry, 'String');
PixelSize = checkPixelSize(str2double(PixelSizeFieldEntry));
% check if a folder was retrieved from the UI
if SelectedFolder == false
    set(handles.FolderNameDisp, 'String', 'No Folder Selected');
    return; % exit, there is nothing more to do
else
    % if a folder was selected, display the path
    set(handles.FolderNameDisp, 'String', SelectedFolder);
    % and save the folder
    handles.LastSelectedFolder = SelectedFolder;
end
%%% extract all the .tif and .jpeg images from the selected folder
% structure for the Images
ImageStruct = struct('FileName', 'none',...
                     'PathString', 'none',...
                     'Type', 'none',...
                     'ImageData', 0,...
                     'PixelSize', 1,...
                     'CenterList', [],...
                     'RadiiList', [],...
                     'Polarity', 'dark',...
                     'Sensitivity', 0.9,...
                     'EdgeThreshold', 0.2,...
                     'DetectionMethod', 'phasecode',...
                     'R_min', 10,...
                     'R_max', 30);
FoundImage = false; % check later if any Images were found
% display a waitbar to indicate the process
h = waitbar(0,'Importing Images. Please wait.');
% get a list of all the files in the selected folder
Files = dir(SelectedFolder);
%%% iterate over the files and select the .tiffs and .jpgs
for Index = 1:length(Files)
    % if it is a directory -> skip it
    if Files(Index).isdir == true
        continue;
    end
    % get the path, name and type (extension of the current file)
    [pathstr, name, ext] = fileparts(fullfile(SelectedFolder,...
                                              Files(Index).name));
    % if its a .tiff file of a .jpg, add it to the list
    if or(strcmp(ext,'.tif'), strcmp(ext,'.jpg'))
        FoundImage = true; % remember that an Image was found
        NewImageStruct = struct(ImageStruct);
        % add it to the File table list
        handles.FileTable = [handles.FileTable; {false, name, 0, 0.0}];
        % structure for the Images
        NewImageStruct.FileName = name;
        NewImageStruct.PathString = fullfile(pathstr,[name ext]);
```

```
        NewImageStruct.Type = ext;
        NewImageStruct.ImageData = imread(NewImageStruct.PathString);
        NewImageStruct.PixelSize = PixelSize;
        % append the new Image structure to the ResultData
        handles.ResultData = [handles.ResultData, NewImageStruct];
    end
    % update the waitbar
    waitbar(Index/length(Files));
end
% close the wait bar
close(h);
% if no images were found return to initial state
if FoundImage == false
    msgbox('No Images were found in the spezified folder.')
    handles.NewFolderSelected = false;
    handles.ResultData = {}; % delete the data
    handles.FileTable = {};
    guidata(hObject, handles);
    set(handles.DataTable, 'Data', handles.FileTable); % update the uitable
    cla(handles.ImageDisplay);  % delete the image
    return   % exit the current callback
else
    % indicate that a new folder was selected for the find circle function
    handles.NewFolderSelected = true;
    guidata(hObject, handles);
end
guidata(hObject, handles);
% display the files in the table
set(handles.DataTable, 'Data', handles.FileTable);
% clear the axis
cla(handles.ImageDisplay);
% display the current image in the table
imshow(handles.ResultData{1}.ImageData);
drawnow;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%CHECKED
function PixelSizeEntry_Callback(hObject, ~, handles) %#ok<DEFNU>
% get the pixel size from the edit field
PixelSize = checkPixelSize(str2double(get(hObject,'String')));
% set the pixel Size for all Images Present
for k = 1:length(handles.ResultData)
   handles.ResultData{k}.PixelSize = PixelSize;
end
% use the pixel size to adjust the avg diameters in the DataTable
% change the average diameters in all entries
for I = 1:length(handles.ResultData)
    MeanDia = mean((handles.ResultData{I}.RadiiList)*PixelSize*2);
    if handles.FileTable{I,3} ~= 0
        handles.FileTable{I,4} = MeanDia;
    end
end
guidata(hObject, handles);
% replace the data in the uitable with the updated data
set(handles.DataTable, 'Data', handles.FileTable);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%CHECKED
function PixelSizeEntry_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FindCircles_Callback(hObject, ~, handles)
% get the new Polarity
NewPolarity = lower(get(handles.ObjPol_Group, 'SelectedObject'));
```

```
% get the new sensitivity value
NewSensitivity = str2double(get(handles.SensEntry,'String'));
% get the new edge threshold value
NewEdgeThreshold = str2double(get(handles.EdgeThreshEntry,'String'));
% get the new detection method
Method = get(handles.DetMeth_Group, 'SelectedObject');
if strcmp(Method.String, 'Phase Code')
    NewDetectMethod = 'phasecode';
elseif strcmp(Method.String, 'Two Stage')
    NewDetectMethod = 'twostage';
end
% get the new R_MIN value
NewR_min = str2double(get(handles.D_MinEntry,'String'))/2;
% get the new R_MAX value
NewR_max = str2double(get(handles.D_MaxEntry,'String'))/2;
% apply the current parameters only to the selected file
% get the index of the currently selected image
CurrImage = handles.CurrentSelectedFile;
% Insert the new fit Parameters into the ResultData structure
handles.ResultData{CurrImage}.Polarity = NewPolarity.String;
handles.ResultData{CurrImage}.Sensitivity = NewSensitivity;
handles.ResultData{CurrImage}.EdgeThreshold = NewEdgeThreshold;
handles.ResultData{CurrImage}.DetectionMethod = NewDetectMethod;
handles.ResultData{CurrImage}.R_min = NewR_min;
handles.ResultData{CurrImage}.R_max = NewR_max;
% Indicate to the user that the circle search is in progress
h = msgbox('Circle Detection in Progress, please wait.',...
           'Circle Detection');
% detect the circles in the current image
[centers, radii] = findCircles(handles.ResultData{CurrImage});
% remove the progress notification
close(h);
%if no circles are found, leave the function call
if isempty(radii)
  warndlg('No circles found!', 'Circle Detection');
  return ;
end
% save the centers /radii and DefaultParameters in the ResultData
handles.ResultData{CurrImage}.CenterList = centers;
handles.ResultData{CurrImage}.RadiiList = radii;
% update the info on Particle Count and avg radius in the DataTable
handles.FileTable{CurrImage,1} = true;
handles.FileTable{CurrImage,3} = length(radii);
handles.FileTable{CurrImage,4} = round(mean(radii)*...
                        handles.ResultData{CurrImage}.PixelSize) * 2;
% put the ResultData into the gui data for global access
guidata(hObject, handles);
% get the end position of the scroll bar
jScrollpane = findjobj(handles.DataTable);                % get the handle of the table
scrollMax = jScrollpane.getVerticalScrollBar.getValue;  % get the end position of the scroll
% display the filed data in the table
set(handles.DataTable, 'Data', handles.FileTable);
% put the ResultData into the gui data for global access
guidata(hObject, handles);
drawnow;
% set scroll bar to the previous entry
jScrollpane.getVerticalScrollBar.setValue(scrollMax);
drawnow;
% clear the axis
cla(handles.ImageDisplay);
% display the current image in the table
imshow(handles.ResultData{CurrImage}.ImageData);
% overlay the found circles in green
viscircles(handles.ResultData{CurrImage}.CenterList,...
           handles.ResultData{CurrImage}.RadiiList,...
           'Color', 'g', 'LineStyle', '-', 'LineWidth', 1);
drawnow;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FileNameEntry_Callback(hObject, ~, handles) %#ok<DEFNU>
% Get the Filename from the user input
handles.ResultFileName = get(hObject,'String');
guidata(hObject, handles);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FileNameEntry_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function SaveData_Callback(~, ~, handles) %#ok<DEFNU>
% save the checkmark selected data to the results file
SaveDataToFile(handles);
% give a message as a response to the user
msgbox('Data export sucessfull!', 'Save');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function EdgeThreshEntry_Callback(hObject, ~, handles) %#ok<DEFNU>
%%% The Edge Threshold has been changed:
% get the new input value
NewEdgeThreshold = str2double(get(hObject,'String'));
% check if its valid, else return with no action
if checkEdgeThreshold(NewEdgeThreshold) == false
    return;
end
% set the new edge threshold
handles.ResultData{handles.CurrentSelectedFile}.EdgeThreshold =...
                                                        NewEdgeThreshold;
guidata(hObject, handles);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%CHECKED
function EdgeThreshEntry_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function SensEntry_Callback(hObject, ~, handles) %#ok<DEFNU>
%%% The Sensitivity has been changed:
% get the new input value
NewSensitivity = str2double(get(hObject,'String'));
% check if its valid
if checkSensitivity(NewSensitivity) == false
    % return to previous state with no change
    return
end
% set the new sensitivity
handles.ResultData{handles.CurrentSelectedFile}.Sensitivity =...
                                                        NewSensitivity;
guidata(hObject, handles);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function SensEntry_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function DataTable_CellSelectionCallback(hObject, eventdata, handles) %#ok<DEFNU>
% if the Indices are empty, return to main
```

```
if isempty(eventdata.Indices)
    return;
end
% get the Row Index in the uiTable
RowIndex = eventdata.Indices(1);
% update the Current Selection in the global variable
handles.CurrentSelectedFile = RowIndex;
guidata(hObject, handles);
% clear the axis
cla(handles.ImageDisplay);
% display the current image
imshow(handles.ResultData{RowIndex}.ImageData);
% overlay the found circles in green
viscircles(handles.ResultData{RowIndex}.CenterList,...
           handles.ResultData{RowIndex}.RadiiList,...
           'Color', 'g', 'LineStyle','-', 'LineWidth', 1);
%%% update the displayed circle detection parameters
if length(handles.ResultData{RowIndex}.RadiiList) ~= 0
    % Object Polarity
    if strcmp(handles.ResultData{RowIndex}.Polarity, 'bright')
        set(handles.radiobutton1, 'Value', 1.0);
    elseif strcmp(handles.ResultData{RowIndex}.Polarity, 'dark')
        set(handles.radiobutton2, 'Value', 1.0);
    end
    % Sensitivity
    set(handles.SensEntry,'String',...
        num2str(handles.ResultData{RowIndex}.Sensitivity));
    % Edge Threshold
    set(handles.EdgeThreshEntry,'String',...
        num2str(handles.ResultData{RowIndex}.EdgeThreshold));
    % Detection Method
    if strcmp(handles.ResultData{RowIndex}.DetectionMethod, 'phasecode')
        set(handles.radiobutton3, 'Value', 1.0);
    elseif strcmp(handles.ResultData{RowIndex}.DetectionMethod, 'twostage')
        set(handles.radiobutton4, 'Value', 1.0);
    end
    % Minimal Diameter
    set(handles.D_MinEntry,'String',...
        num2str(handles.ResultData{RowIndex}.R_min*2));
    % Maximal Diameter
    set(handles.D_MaxEntry,'String',...
        num2str(handles.ResultData{RowIndex}.R_max*2));
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function DataTable_CellEditCallback(hObject, ~, handles) %#ok<DEFNU>
% read in the boolean values from the first row and save them
handles.FileTable = get(hObject, 'data');
guidata(hObject, handles);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function D_MinEntry_Callback(hObject, ~, handles) %#ok<DEFNU>
%%% The minimal search radius has been changed:
NewR_min = str2double(get(hObject,'String'));
% check if its valid
if NewR_min <= 0
    return;
end
% set the new r_min
handles.ResultData{handles.CurrentSelectedFile}.R_min = NewR_min;
% update the gui data
guidata(hObject, handles);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function D_MinEntry_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function D_MaxEntry_Callback(hObject, ~, handles)
%%% The maximal search radius has been changed:
% get the new input value
NewR_max = str2double(get(hObject,'String'));
% check if its valid
if NewR_max <= 0
    return;
end
% set the new r_min
handles.ResultData{handles.CurrentSelectedFile}.R_max = NewR_max;
guidata(hObject, handles);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function D_MaxEntry_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function ObjPol_Group_SelectionChangedFcn(hObject, ~, handles) %#ok<DEFNU>
%%% Object Polarity has been changed:
% get the new Polarity
NewPolarity = get(handles.ObjPol_Group, 'SelectedObject');
% set the new Polarity in the Results data structure
handles.ResultData{handles.CurrentSelectedFile}.Polarity =...
                                                lower(NewPolarity.String);
guidata(hObject, handles);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function DetMeth_Group_SelectionChangedFcn(hObject, ~, handles) %#ok<DEFNU>
%%% Detection Method has been changed:
% get the new detection method
NewMethod = get(handles.DetMeth_Group, 'SelectedObject');
if strcmp(NewMethod.String, 'Phase Code')
    handles.ResultData{handles.CurrentSelectedFile}.DetectionMethod =...
elseif strcmp(NewMethod.String, 'Two Stage')
    handles.ResultData{handles.CurrentSelectedFile}.DetectionMethod =...
                                                            'twostage';
end
% update the GUI data
guidata(hObject, handles);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes during object creation, after setting all properties.
function ImageDisplay_CreateFcn(~, ~, ~) %#ok<DEFNU>
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes during object creation, after setting all properties.
function DataTable_CreateFcn(~, ~, ~) %#ok<DEFNU>
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes during object creation, after setting all properties.
function FolderNameDisp_CreateFcn(~, ~, ~) %#ok<DEFNU>
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.4.2 Detect Circles

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function performes the circle detection on a single image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ImageCenters, ImageRadii] = findCircles(Image)
SPart = 0; % cut off for the bottom info table
[height, width, ~] = size(Image.ImageData);
%%% For SEM Images search only in the upper 89% of the image.
if strcmp(Image.Type, '.tif')
   SPart = floor(height*0.89);
%%% for TEM Images, look in the upper 90% of the image.
elseif strcmp(Image.Type, '.jpg')
   SPart = floor(height*0.90);
end
%%% find the circles
[centers, radii] = imfindcircles(Image.ImageData(1:SPart,:),...
[floor(Image.R_min/Image.PixelSize),...
ceil(Image.R_max/Image.PixelSize)],...
'ObjectPolarity', Image.Polarity,...
'Sensitivity', Image.Sensitivity,...
'Method', Image.DetectionMethod,...
'EdgeThreshold', Image.EdgeThreshold);
%%% remove circles which are outside of the image!?
%% also if they are close to the boundary
PartBound = ceil((Image.R_max + Image.R_min)/2/Image.PixelSize * 0.7);
ImageCenters = [];
ImageRadii = [];
for k = 1:length(centers)
    if centers(k, 1) < width &&...
       centers(k, 1) > PartBound &&...
       centers(k, 2) < SPart - PartBound &&...
       centers(k, 2) > PartBound
        ImageCenters = [ImageCenters; centers(k,:)];
        ImageRadii = [ImageRadii, radii(k)];

    end
end
end
```

A.4.3 Histogram Plot Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function creates a histogram plot from the selected data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotHistogram(Diameters, FileNamePath, FileName)
    MyFontSize = 16;
    % make a histogram of all the radii
    tempFig = figure('Visible', 'off');
    set(gca,'fontsize', MyFontSize);
    h = histfit(Diameters);
    disp(h);
    xlabel('Diameter [nm]', 'FontSize', MyFontSize);
    ylabel('Probability Density [1/nm]', 'FontSize', MyFontSize);
    MyTitleString = strcat('D =', {' '}, num2str(round(mean(Diameters))), ' +/-', {' '},...
                    num2str(ceil(std(Diameters))), ' nm (',...
            num2str(round(std(Diameters)/mean(Diameters)*100, 2)), " %)");
    title(MyTitleString, 'FontSize', MyFontSize);
    legend([' NPart = ', num2str(length(Diameters))], 'Gauss Fit');
    % save the histogram in the folder with the images
    print(tempFig,fullfile(FileNamePath,[FileName, '_Hist']),'-dpng');
    % close the temporary figure
```

```
    close(tempFig);
end
```

A.4.4 Plot Detected Particles

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function overlays the circles found in every Image and saves into
%%% the same folder
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotImagesWithCircles(handles)
Results = handles.ResultData;
[Folder, ~, ~] = fileparts(Results{1}.PathString);
for R = 1:length(Results)
    if handles.FileTable{R,1} == true
    fig = figure(); % create a new figure
    set(fig, 'visible', 'off'); % make the figure invisible
    % display the current image in the table
    imshow(Results{R}.ImageData);
    % overlay the found circles in red
    viscircles(Results{R}.CenterList, Results{R}.RadiiList,...
               'Color', 'g', 'LineStyle', '-', 'LineWidth', 1);
    [~, FileName, ~] = fileparts(Results{R}.PathString);
    saveas(gca, fullfile(Folder, [FileName, '_CircDet']), 'png');
    clf;
    close();
    end
end
```

A.4.5 Save The Global Results

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function saves all the selected data to a result file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function SaveDataToFile(handles)
Results = handles.ResultData;
FileName = handles.ResultFileName;
SelectList = handles.FileTable;
% check if the Pixel size was entered correctly!
if Results{1}.PixelSize == 1
    % ask the user if the pixel size is correct
    button = questdlg(['The Pixel Size is set to ',...
        num2str(PixelSize), ' nm.',...
        ' Are you sure this is the right pixel size?'], 'PixelSize');
    % if not or cancel do nothing and return to main GUI
    if strcmp(button, 'Cancel')
        return;
    elseif strcmp(button, 'No')
        return;
    end
end
% check if a FileName was entered by the user
if strcmp(FileName, 'EnterFileName')
    % ask the user if the pixel size is correct
    button = questdlg(['The file name appears to be unchanged.',...
    ' Are you sure you want to use the default file name?'],...
    'Filename');
    % if not or cancel do nothing and return to main GUI
    if strcmp(button, 'Cancel')
        return;
    elseif strcmp(button, 'No')
        return;
```

```
    end
end
% display a waitbar to indicate the process
WaitbarHandle = waitbar(0,'Saveing Data. Please wait.');
%% save one file per image with its circle positions
for FilePos = 1:length(Results)
    % update the waitbar
    waitbar(FilePos/(length(Results)*2));
    if SelectList{FilePos, 1} == true
        SaveSingleImgData(handles, FilePos);
    end
end
%% Save one file with a list of all radii
% extract the Selected Folder path
[pathstr, ~, ~] = fileparts(Results{1}.PathString);
% create and open a new file called FileName.txt
DataFile = fopen(fullfile(pathstr,[FileName, '.txt']), 'W');
% write the header
fprintf(DataFile, 'ParticleSizer Results\r\n');
fprintf(DataFile, ['Data Processed on: ',...
                              datestr(datetime('now')), '\r\n']);
fprintf(DataFile, ['Global Pixel Size: ',...
    num2str(Results{1}.PixelSize), ' nm/Pixel\r\n']);
fprintf(DataFile, '# List of all Diameters [nm]\r\n');
AllDiaList = [];
% write every checked file data separately
for R = 1:length(Results)
    % update the waitbar
    waitbar((R + length(Results))/(length(Results)*2));
    if SelectList{R}(1) == true
        for C = 1:length(Results{R}.RadiiList)
            CurrDia = Results{R}.RadiiList(C)*Results{R}.PixelSize*2;
            AllDiaList = [CurrDia, AllDiaList];
            fprintf(DataFile, [num2str(CurrDia),'\r\n']);
        end
    end
end
% write the average and std of all radii
MeanDia = mean(AllDiaList);
STDDia = std(AllDiaList);
fprintf(DataFile, ['The particles have a diameter of: ',...
    num2str(MeanDia), ' +/- ', num2str(STDDia), ' nm (',...
    num2str(STDDia/MeanDia*100), ' %)\r\n']);
% close the file
fclose(DataFile);
% plot a histogramm of all radii with a gauss fit
plotHistogram(AllDiaList, pathstr, FileName)
% plot all images with the detected circles overlayed
plotImagesWithCircles(handles);
% close the wait bar
close(WaitbarHandle);
end
```

A.4.6 Save The Individual Image Particle Coordinates

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function saves the detected cicles for a single image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function SaveSingleImgData(handles, FilePos)
Results = handles.ResultData;
% extract the Selected Folder path
[pathstr, name, ~] = fileparts(Results{FilePos}.PathString);
% create and open a new file called FileName.txt
DataFile = fopen(fullfile(pathstr,[name, '_PosData.txt']), 'W');
% write the header
```

```
fprintf(DataFile, 'ParticleSizer Results\r\n');
fprintf(DataFile, ['Data Processed on: ',...
                                datestr(datetime('now')), '\r\n']);
fprintf(DataFile, ['Global Pixel Size: ',...
    num2str(Results{FilePos}.PixelSize), ' nm/Pixel\r\n']);
% write separate header for each image
fprintf(DataFile, ['Filename: ', Results{FilePos}.FileName, '\r\n']);
fprintf(DataFile, ['Particle Count: ',...
               num2str(length(Results{FilePos}.RadiiList)), '\r\n']);
fprintf(DataFile, ['Average Particle Diameter: ',...
num2str(2 * mean(Results{FilePos}.RadiiList) * Results{FilePos}.PixelSize),...
' +/- ', num2str(2 * std(Results{FilePos}.RadiiList) *...
Results{FilePos}.PixelSize), ' nm ( ',...
num2str(std(Results{FilePos}.RadiiList) / mean(Results{FilePos}.RadiiList) * 100), '%)\r\n']);
% search parameters used
fprintf(DataFile, ['Polarity: ', Results{FilePos}.Polarity, '\r\n']);
fprintf(DataFile, ['Sensitivity: ',...
                  num2str(Results{FilePos}.Sensitivity), '\r\n']);
fprintf(DataFile, ['Edge Threshold: ',...
                num2str(Results{FilePos}.EdgeThreshold), '\r\n']);
fprintf(DataFile, ['Method: ',...
                     Results{FilePos}.DetectionMethod, '\r\n']);
fprintf(DataFile, ['Search Radius_Min: ',...
                        num2str(Results{FilePos}.R_min),...
                        ' nm\r\n']);
fprintf(DataFile, ['Search Radius_Max: ',...
                        num2str(Results{FilePos}.R_max),...
                        ' nm\r\n']);
fprintf(DataFile, '#Radius [nm]\tX[nm]\tY[nm]\r\n');
% write the circles found in the image
% mirror at the x-axis and shift into the first quadrant of
% a cartesian coordinate system:
% the center is at the bottom left corner
[height, ~, ~] = size(Results{FilePos}.ImageData);
for C = 1:length(Results{FilePos}.RadiiList)
    CurrRadius = Results{FilePos}.RadiiList(C)*Results{FilePos}.PixelSize;
    CurrXPos = Results{FilePos}.CenterList(C,1)*Results{FilePos}.PixelSize;
    CurrYPos = (-Results{FilePos}.CenterList(C,2) +...
            2*height)*Results{FilePos}.PixelSize;
    fprintf(DataFile, [num2str(CurrRadius), '\t', num2str(CurrXPos),...
        '\t', num2str(CurrYPos), '\r\n']);
end
% close the file
fclose(DataFile);
end
```

A.4.7 Check The Edge Threshold

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function checks if the Edge Threshold is between 0 and 1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function TestResult = checkEdgeThreshold(EdgeThreshold)
    if EdgeThreshold <= 0
        errordlg(['The Edge Threshold: ', num2str(EdgeThreshold),...
            ' is too low. Please choose a value between 0 and 1.']);
        TestResult = false;
    elseif EdgeThreshold > 1
        errordlg(['The Edge Threshold: ', num2str(EdgeThreshold),...
            ' is too high. Please choose a value between 0 and 1.']);
        TestResult = false;
    else
        TestResult = true;
    end
end
```

A.4.8 Check The Pixel Size

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function checks if the pixel size is valid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function newSize = checkPixelSize(Size)
if Size > 0
    newSize = Size;
else
    newSize = 1;
    errordlg(['A Pixel Size of ', num2str(Size), ' is invalid!',...
        ' Please enter a Pixel Size > 0.'],'Pixel Size');
end
```

A.4.9 Check The Sensitivity

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function checks if the Sensitivity is between 0 and 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function TestResult = checkSensitivity(Sensitivity)
    if Sensitivity <= 0
        errordlg(['The Sensitivity: ', num2str(Sensitivity),...
            ' is too low. Please choose a value between 0 and 1.']);
        TestResult = false;
    elseif Sensitivity > 1
        errordlg(['The Sensitivity: ', num2str(Sensitivity),...
            ' is too high. Please choose a value between 0 and 1.']);
        TestResult = false;
    else
        TestResult = true;
    end
end
```

Eidesstattliche Versicherungen und Erklärungen

§8 Satz 2 Nr. 3 PromO Fakultät

Hiermit versichere ich eidesstattlich, dass ich die Arbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt habe (vgl. Art. 64 Abs 1 Satz 6 BayHSchG).

§8 Satz 2 Nr. 3 PromO Fakultät

Hiermit erkläre ich, dass ich die Dissertation nicht bereits zur Erlangung eines akademischen Grades eingereicht habe und dass ich nicht bereits diese oder eine gleichartige Doktorprüfung endgültig nicht bestanden habe.

§8 Satz 2 Nr. 4 PromO Fakultät

Hiermit erkläre ich, dass ich Hilfe von gewerblichen Promotionsberatern bzw. –vermittlern oder ähnlichen Dienstleistern weder bisher in Anspruch genommen habe noch künftig in Anspruch nehmen werde.

§8 Satz 2 Nr. 7 PromO Fakultät

Hiermit erkläre ich mein Einverständnis, dass die elektronische Fassung der Dissertation unter Wahrung meiner Urheberrechte und des Datenschutzes einer gesonderten Überprüfung unterzogen werden kann.

§8 Satz 2 Nr. 8 PromO Fakultät

Hiermit erkläre ich mein Einverständnis, dass bei Verdacht wissenschaftlichen Fehlverhaltens Ermittlungen durch universitätsinterne Organe der wissenschaftlichen Selbstkontrolle stattfinden können.

Bayreuth, 17. Dezember 2019

Bernd Kopera

Colophon

This thesis was typeset with LaTeX 2_ε in the TeXstudio integrated development environment. I used the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc. You can download the *Clean Thesis* style at http://cleanthesis.der-ric.de/. I created the vector graphics with Corel Draw X7. I used the programming language Python 3 for data analysis together with the Matplotlib library for data visualization.

Bernd Friedrich Alfred Kopera

Building Scientific Tools:
From Heat Transport to Particulate Systems

Dissertation, December 17, 2019

University of Bayreuth
Physical Chemistry I (PC I)
Faculty of Biology, Chemistry & Earth Sciences
Universitätsstraße 30
95447 Bayreuth