UNIVERSITÄT BAYREUTH

Bachelorarbeit

im Studiengang Mathematik
Lehrstuhl für angewandte Mathematik

Optimal Sensor Placement for linear Systems

eingereicht von: Maximilian Pfister

eingereicht am: 17.08.2012

Betreuer: Prof. Dr. Tobias Damm
Dr.-Ing. Ulrich Münz (Siemens AG)

Abstract

The aim of sensor placement is to observe the state of a dynamical system while using only a small part of the available output information. Thus, the observer does not need sensors at every possible node of the system. We use sensor placement because it is not practical for large-scale networks, such as power grids, to place sensors at each node. With an optimal sensor placement we obtain a subset of sensors which minimizes the observer error in comparison to any other subset of the same size. This means we generate an optimal observation with the given number of sensors.

We compute the observer error, for the linear dynamical systems we consider, with the H_2-norm of the observer error system. In this approach, we optimize both the subset of selected sensors and the observer gain matrix in parallel. The optimization problem is non-convex both in a constraint, which bounds the H_2-norm, as well as in the objective function which uses a ℓ_0-norm to count the used sensors. To obtain a semidefinite program, we first relax the ℓ_0-norm by an iterative reweighted ℓ_1-norm. Second, we use a reformulation of the H_2-norm with linear matrix inequalities to replace an occuring bilinear and therefore non-convex term.

We use this computationally efficient formulation of the sensor placement problem to derive three algorithms. Furthermore, existing algorithms, which do not use the convex reformulation of the optimization problem, were implemented. The algorithms are compared extensively relating to execution time, performance of the chosen sensors, and the applicability on a practical problem. The practical problem is a model of a high-voltage power grid with the aim to measure the phase angles and the frequencies at every node. The result of the comparison is that a algorithm with a greedy approach solves the optimization problem fast and usually with a good solution. However, this algorithm is problematic because the shortsighted greedy approach cannot exclude that a worst case solution is generated. The best results in general were produced by a novel approach made in this thesis. This novel algorithm iteratively solves the relaxed optimization problem and finds near-optimal sensor subsets.

Contents

1 Introduction

In this thesis, we consider the problem of observing the state of dynamical systems, such as energy networks, as accurate as possible with a small number of sensors. This problem is interesting since there exist large networks, where it is not cost-efficient to place a sensor at every node of the system. Consequently, we can not measure all the states of the system which results in an observer error. The aim of this thesis is to find an optimal subset of sensors, i.e. a subset, such that every other subset of the same size has a bigger observer error.

The problem of sensor selection appears in several areas of application like robotics, sensor placement for structures, target tracking, chemical plant control and wireless networks as listed in Joshi and Boyd [2009]. The field of application of this thesis are energy networks, especially high-voltage networks, as seen in the example of Section 4.4.

We consider the problem of choosing an optimal subset from among n potential sensors of a time-invariant linear dynamic system for state estimation subject to white input noise. Each sensor can measure one component of the output vector y. Thus the process of sensor selection reduces the available information for the observer. The sensor selection consequently has an influence on the observer error. The aim of this thesis is to choose an H_2-optimal subset, i.e. a subset that minimizes the H_2-norm of the observer error system. The H_2-norm describes the total output energy of the impulse response of the error system. A simple approach to evaluate the best k sensor subset would be to calculate the H_2-norm with an optimal observer for all possible $\binom{n}{k}$ combinations of subsets. But since $\binom{n}{k}$ grows rapidly with increasing n and k, this method is not practical. For example, with $n = 50$ potential sensors and $k = 25$ sensors to choose there are over 10^{14} possible tuples, so a sequential evaluation is obviously not possible.

In this thesis, we provide several algorithms, partly based on convex optimization, for a near optimal solution of the sensor selection problem. And we use one combinatorial algorithm with a branch and bound technique for an optimal solution in order to compare the performance of the relaxed algorithms to this optimal solution.

When comparing the execution time of the algorithms, it can be seen that the branch and bound algorithm is a NP-hard problem, while the other algorithms have a polynomial computational complexity.

The problem we consider consists of two components. First we have to choose a subset of sensors which have to be used. Second, we have to calculate an optimal observer gain matrix for these settings in order to evaluate the objective function. In this approach, we provide a setting that solves these two problems in parallel.

Various papers treat related problems to the one discussed here. In Mo et al. [2011], we find a general approach for sensor selection strategies for wireless sensor networks. However, the framework does not hold for our case since the strategy is chosen for a finite time horizon. This finiteness is used in the included manipulation of the optimization problem for a recursively defined equation of the state estimation. Given an infinite time horizon, we could not perform the same reformulations.

In Schuler et al. [2012], we find an almost dual problem to the observer design in this thesis. This paper with the title *Decentralized State Feedback Control for Interconnected Process Systems* seeks to minimize the number of measurement links between sensors and controllers and creates a convex optimization problem, that is similar to the problem derived in this thesis. Nevertheless, the derivation of the convex optimization problem is not applicable to the problem discussed here, because, on the one hand, the controller problem has slightly different bilinearities which have to be substituted for convex optimization, and on the other hand Schuler et al. [2012] use the H_∞-norm in contrast to the here applied H_2-norm.

The approach in this thesis starts with describing the system of the estimation error. By introducing a design matrix, we create the possibility to vary the setting of the problem, i.e. to punish estimation errors in certain states more than in others. In the next step, we use a LMI-characterization of the H_2-norm to reformulate this constraint to a semidefinite one. The mentioned constraint is relaxed in two different ways that afterwards result in different algorithms. The ℓ_0-norm for observer-sparsity in the objective is a non-convexity that is relaxed by an iterative weighted ℓ_1-norm.

The rest of the thesis is organized as follows. We present the problem formulation in Chapter 2. In Chapter 3, we describe convex relaxations of the optimization problem and the derived algorithms both for discrete-time and continuous-time systems. In Chapter 4, extensive comparison of the different algorithms is presented, including a runtime and a performance comparison as well as an example of an energy-network model to test the algorithms in praxis. The thesis concludes with a summary and an outlook in Chapter 5.

2 Problem Formulation

2.1 Problem Formulation for Discrete-Time Systems

Consider the following discrete-time dynamical system

$$\begin{aligned} x_{k+1} &= Ax_k + Bw_k && (2.1a) \\ y_k &= Cx_k, && (2.1b) \end{aligned}$$

where $k \in \mathbb{N}_0^+$, $x_k \in \mathbb{R}^n$ is the state of a system with n states and $y_k \in \mathbb{R}^m$, $m \leq n$, is the set of possible sensors. The unknown input $w_k \in \mathbb{R}^p$ is zero-mean white noise with unit variance. Assuming that the matrices $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, and $C \in \mathbb{R}^{m \times n}$ are known, we define a Luenberger observer.

$$\begin{aligned} \hat{x}_{k+1} &= A\hat{x}_k + L(y_k - \hat{y}_k) && (2.2a) \\ \hat{y}_k &= C\hat{x}_k, && (2.2b) \end{aligned}$$

where $L \in \mathbb{R}^{n \times m}$ describes the observer gain matrix that has to be designed. An optimal L could be found by using the principle of the Kalman filter, which uses the same setup as the one described here.

We can express the observer error $e = x - \hat{x}$:

$$\begin{aligned} e_{k+1} &= (A - LC)e_k + Bw_k && (2.3a) \\ z_k &= We_k. && (2.3b) \end{aligned}$$

The matrix $W \in \mathbb{R}^{n \times n}$ can be used as a design matrix when the desired accuracy of the observations of certain states differs or if a scaling has to be done. Elsewise W could simply be defined as an identity matrix.

Let us assume there exists a stabilizing L. Thus we can guarantee that the H_2-norm of the system exists and characterize the observer error with that norm, again as in the setup of Kalman filter. With the aim to minimize the number of used sensors with a bounded observer error we can define the following simplified optimization problem:

$$\begin{aligned} \min_{L} \quad & \text{"no. of sensors"} && (2.4a) \\ \text{s.t.} \quad & ||\Sigma_e(L)||_2^2 < \gamma, && (2.4b) \end{aligned}$$

where $\Sigma_e(L)$ is a short form for the error system depending on L as described in (2.3). $|| \cdot ||_2$ denotes the H_2-norm of the system and γ is a predefined upper bound to the squared H_2-norm.

The number of used sensors in (2.4) is the number of non-zero columns in L, i.e. if every element of a column of L is zero, the corresponding entry in y_k has no influence on the observer system, which means that we could remove the sensor belonging to y_k. Hence, it is the same optimization problem if we replace the "no. of sensors" in the objective of (2.4a) with a term that counts the non-zero columns of L.

Obviously, the accuracy of the observation will drop as the number of active sensors decreases because L has consequently a smaller degree of freedom.

$$\begin{aligned} \min_{L} \quad & \sum_{j=1}^{n} \Big|\Big| \sum_{i=1}^{n} |L_{ij}| \Big|\Big|_{\ell_0} && \text{(2.5a)} \\ \text{s.t.} \quad & ||\Sigma_e(L)||_2^2 < \gamma, && \text{(2.5b)} \end{aligned}$$

where $||\cdot||_{\ell_0}$ denotes the ℓ_0-norm, i.e., for $x \in \mathbb{R}$, $||x||_{\ell_0} = 1$ if $x \neq 0$ and $||x||_{\ell_0} = 0$ if $x = 0$.

In (2.5a) we use a predefined γ to bound the H_2-norm of the error system. Since it is difficult to choose an appropriate γ a priori, we can minimize a combination of the H_2-norm and the number of active sensors. Thus, we can look at a trade-off between the two objectives.

$$\begin{aligned} \min_{L,\gamma} \quad & \gamma + \alpha \sum_{j=1}^{n} \Big|\Big| \sum_{i=1}^{n} |L_{ij}| \Big|\Big|_{\ell_0} && \text{(2.6a)} \\ \text{s.t.} \quad & ||\Sigma_e(L)||_2^2 < \gamma, && \text{(2.6b)} \end{aligned}$$

where γ is a variable denoting the upper bound on the H_2-norm and α is a parameter to balance the mentioned trade-off.

2.2 Problem Formulation for Continuous-Time Systems

The continuous-time system, that is equivalent to the discrete-time system (2.1), is this:

$$\begin{aligned} \dot{x} &= Ax + Bw && \text{(2.7a)} \\ y &= Cx, && \text{(2.7b)} \end{aligned}$$

where the dimensions of the matrices A, B, and C are the same as in Section 2.1, but the entries of the matrices would be different due to the conversion of a discrete-time into a continuous-time system.

In the following, one can see the corresponding system of the observer error with an observer designed as in (2.2).

$$\begin{aligned} \dot{e} &= (A - LC)e + Bw && \text{(2.8a)} \\ z &= We, && \text{(2.8b)} \end{aligned}$$

This system does not differ much of the discrete-time model. We can remark a considerable change, however, when reformulating the H_2-norm in the optimization problem (2.5) or (2.6).

3 Convex Relaxation of the Optimization Problem

In this chapter, we consider the convex relaxation of the optimization problem. This includes the relaxation of the ℓ_0-objective with an iterative reweighted ℓ_1-norm. Additionally, we rewrite the upper bound on the H_2-performance of the system to remove bilinearities for discrete-time as well as for continuous-time systems. Furthermore, we derive algorithms to solve the optimization problem efficiently.

3.1 Convex Relaxation of the ℓ_0-Objective

The ℓ_0-norm as formulated in (2.6a) counts the non-zero columns of the observer gain matrix L. The result of this term is consequently an integer and therefore a discrete and non-convex term in the objective. Only very exhaustive combinatorial search leads to an optimal sparse solution when this formulation of the objective is applied. In practical use, this is not tractable because the problem is NP-hard and grows very fast with increasing size of the system. This can be seen in Chapter 4.

Our aim is to substitute the ℓ_0-norm with a convex function that is as closely related to the ℓ_0-norm as possible. Therefore, the convex envelope of the ℓ_0-norm looks suitable. This convex envelope is defined next.

Definition 1. The function f is defined as $f : \mathbb{X} \to \mathbb{R}$, where $\mathbb{X} \subseteq \mathbb{R}^n$. The convex envelope of f (on $\mathbb{X}$), written as f_{env}, is defined as the point-wise largest convex function g such that $g(x) \leq f(x)$ for all $x \in \mathbb{X}$.

Lemma 1 (As in Fazel [2002] and in Schuler et al. [2012])**.** The convex envelope of the function $f = ||x||_0 = \sum_{i=1}^n |\text{sign}(x_i)|$ on $\mathbb{X} = \{x \in \mathbb{R}^n \mid ||x||_\infty \leq 1\}$ is $f_{\text{env}}(x) = ||x||_1 = \sum_{i=1}^n |x_i|$.

So, the ℓ_1-norm is the convex envelope of the ℓ_0-norm for selective x. The restriction on x is not important in our case, because it is possible to do a scaling in the objective function. The values of L could, for example, be estimated with the size of the values of A.

As proposed in Candes et al. [2008], a reweighted ℓ_1-norm could be used to improve the results generated by the relaxed problem. We use the weights to counteract the size of the different variables considered in the objective. If we choose the weights inverse to the size of the variables, in our case the ℓ_1-norm of the columns of L, the weighted ℓ_1-norm coincides with the ℓ_0-norm. The weights ω_j are chosen as follows:

$$\omega_j := \begin{cases} 1/||L_{*j}||_{\ell_1} & \text{for } ||L_{*j}||_{\ell_1} \neq 0 \\ \infty & \text{for } ||L_{*j}||_{\ell_1} = 0 \end{cases} .$$

The weights can be assigned to the corresponding variables L_{*j} and we obtain the weighted ℓ_1-norm of the columns of L as

$$\sum_{j=1}^{n} \omega_j ||L_{*j}||_{\ell_1}.$$

It is not possible to change the weights during the computation of the optimization problem because the weights obviously depend on the optimization variable L. Candes et al. [2008] suggest to use an iterative procedure to adjust the weights according to the solution of the previous iteration.

Using the weights ω_j in the optimization problem presents another advantage. If knowledge about the system exists before the optimization, the weights could be adjusted a priori to improve the result.

Thus, the optimization problem (2.6) can be relaxed as

$$\begin{aligned} \min_{L,\gamma} \quad & \gamma + \alpha \sum_{j=1}^{n} \omega_j^{(\mu)} ||L_{*j}||_{\ell_1} \\ \text{s.t.} \quad & ||\Sigma_e(L)||_2^2 < \gamma, \end{aligned}$$

where μ is the iteration variable for the reweighting.

3.2 Reformulation of the H_2-Performance Constraint

Next, we consider the H_2-norm of a system that depends on an unknown observer. In our case, a complex reformulation is necessary to obtain a semidefinite program. The H_2-norm is transformed into the corresponding characterization with linear matrix inequalities (LMIs). Linear matrix inequalities appeared recently in several research areas. Efficient solvers were developed to solve such optimization problems. For example, one standard LMI solver is mentioned in Sturm [1999] and Lofberg [2004].

The theorem is written here for a general time-discrete system with transfer function $G(z) = \left[\begin{smallmatrix} A & B \\ C & 0 \end{smallmatrix}\right] := C(zI - A)^{-1}B$. The special cases we need will be considered after the theorem together with the necessary elimination of non-linear terms in the matrix inequalities.

Lemma 2. (As in Rieber [2006]) Consider a time-discrete system with transfer function

$$G(z) = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} := C(zI - A)^{-1}B.$$

The following statements are equivalent:

(i) $||G(z)||_2^2 < \gamma$ and A is asymptotically stable.

(ii) There exists a positive definite symmetric solution P of the Lyapunov equation $A^TPA - P = -C^TC$ and $\text{trace}(B^TPB) < \gamma$.

(iii) There exists a positive definite symmetric solution X of the following LMI

$$\begin{pmatrix} -X + C^TC & A^TX \\ XA & -X \end{pmatrix} \prec 0$$

and $\text{trace}(B^TXB) < \gamma$.

The theorem is similar for continuos-time systems which is considered in Section 3.4.

Proof: (partly as in Smith [2010])

(i) $\Leftrightarrow$ (ii)

$||G(z)||_2^2 < \gamma$

$\Leftrightarrow \dfrac{1}{2\pi}\displaystyle\int_{-\pi}^{\pi} \text{trace}(G(e^{j\omega})G(e^{j\omega})^*)d\omega < \gamma$, which is the definition of the H_2-norm on the frequency domain. In the next step we use an equivalent definition on the time domain.

$\Leftrightarrow \displaystyle\sum_{t=0}^{\infty} \text{trace}(g(t)^T g(t)) < \gamma$, where $g(t)$ is the impulse response of the system $G(z)$.

The impulse response is defined as $g(t) := CA^tB$.

$\Leftrightarrow \displaystyle\sum_{t=0}^{\infty} \text{trace}[B^T(A^t)^T C^T CA^t B] < \gamma$ with the definition of the impulse response.

$\Leftrightarrow \text{trace}[B^T(\displaystyle\sum_{t=0}^{\infty}(A^t)^T C^T CA^t)B] < \gamma$

$\Leftrightarrow \text{trace}[B^T PB] < \gamma$, where P is the observability Gramian (possible, since A is stable) which is defined as $P = \displaystyle\sum_{t=0}^{\infty}(A^t)^T C^T CA^t$.

$\Leftrightarrow \text{trace}[B^T PB] < \gamma,\ P \succ 0$ and $A^T PA - P = -C^T C$

(ii) $\Rightarrow$ (iii)
Define the positive definite and symmetric matrix X with $\varepsilon > 0$:

$$X := \sum_{t=0}^{\infty}(A^t)^T \begin{bmatrix} C^T & \varepsilon I \end{bmatrix} \begin{bmatrix} C \\ \varepsilon I \end{bmatrix} A^t.$$

$\Rightarrow A^T XA - X + C^T C + \varepsilon^2 I = 0$ and as $\varepsilon \to 0$ it follows that $X \to P$ (continuosly).
$\text{trace}(B^T PB) < \gamma$ implies that there exist sufficiently small ε such that $\text{trace}(B^T XB) < \gamma$.
$\Rightarrow$ By choosing an appropriate $\varepsilon > 0$, the inequality $A^T XA - X + C^T C \prec 0$ is satisfied and $\text{trace}(B^T XB) < \gamma$.
$\Rightarrow$ The inequality can be written as $A^T X(X^{-1})XA - X + C^T C \prec 0$ and $\text{trace}(B^T XB) < \gamma$.
$\Rightarrow$ By using the Schur complement the inequality can be transformed into

$$\begin{pmatrix} -X + C^T C & A^T X \\ XA & -X \end{pmatrix} \prec 0$$

and $\text{trace}(B^T XB) < \gamma$.

(iii) $\Rightarrow$ (ii)

With Schur complement

$$\begin{pmatrix} -X + C^T C & A^T X \\ XA & -X \end{pmatrix} \prec 0$$

is equivalent to the inequality $A^T X A - X + C^T C \prec 0$.
The Lyapunov inequality implies that $G(z)$ is stable. Furthermore there exists a positive definite symmetric matrix Q such that $A^T X A - X + C^T C + Q = 0$.
$\Rightarrow A^T (X - P) A - (X - P) + Q = 0$
A is Hurwitz so $X - P = \sum_{t=0}^{\infty} (A^t)^T Q A^t \succ 0$.
$\Rightarrow ||G(z)||_2^2 = \text{trace}(B^T P B) \leq \text{trace}(B^T X B) < \gamma$

□

To simplify the solving of our optimization problem (2.6b), we need to reformulate the H_2-norm of the error system. The transfer function of the error system is

$$\Sigma_e(L)(z) = \begin{bmatrix} A - LC & B \\ W & 0 \end{bmatrix} := W(zI - (A - LC))^{-1} B.$$

Applying Theorem 2 shows that $||\Sigma_e(L)||_2^2 < \gamma$ is equivalent to

$$\begin{pmatrix} -X + W^T W & (A - LC)^T X \\ X(A - LC) & -X \end{pmatrix} \prec 0 \tag{3.1}$$

and $\text{trace}(B^T X B) < \gamma$ with X being a positive definite symmetric matrix.
The matrix inequality (3.1) is the same as

$$\begin{pmatrix} -X + W^T W & A^T X - C^T L^T X \\ XA - XLC & -X \end{pmatrix} \prec 0, \tag{3.2}$$

which will be considered to derive algorithm 1. We can alter (3.2) by pre- and post-multiplying with $\left(\begin{smallmatrix} I & 0 \\ 0 & X^{-1} \end{smallmatrix}\right)$. The resulting matrix inequality is:

$$\begin{pmatrix} -X + W^T W & A^T - C^T L^T \\ A - LC & -X^{-1} \end{pmatrix} \prec 0. \tag{3.3}$$

This formulation will be used to apply a cone complementarity linearization, which is the basis for Algorithm 2.

3.3 Algorithms for Optimal Sensor Placement for Discrete-Time Systems

In the following section, we derive two different algorithms to solve the ℓ_1-relaxed optimization problem. We use the previously mentioned H_2-reformulation to alter the optimization problem to a semidefinite program.

3.3.1 Algorithm with Substitution

The formulation we currently consider is

$$\begin{aligned}
\min_{\gamma,L,X} \quad & \gamma + \alpha \sum_{j=1}^{n} \omega_j^{(\mu)} ||L_{*j}||_{\ell_1} && \text{(3.4a)} \\
\text{s.t.} \quad & X = X^T \succ 0 && \text{(3.4b)} \\
& \text{trace}(B^T X B) < \gamma && \text{(3.4c)} \\
& \begin{pmatrix} -X + W^T W & A^T X - C^T L^T X \\ XA - XLC & -X \end{pmatrix} \prec 0. && \text{(3.4d)}
\end{aligned}$$

As we can see, γ, X, and L are optimization variables. The product $X \cdot L$ presents itself as bilinear term. Therefore, we cannot use this formulation for semidefinite programming. The novel approach in this thesis is to make use of the special appearence of the bilinearity. We can easily substitute the bilinear term $X \cdot L$ with a new optimization variable $\tilde{L} \in \mathbb{R}^{n \times m}$. The constraints with the applied substitution then look like:

$$\begin{aligned}
\min_{\gamma,L,\tilde{L},X} \quad & \gamma + \alpha \sum_{j=1}^{n} \omega_j^{(\mu)} ||L_{*j}||_{\ell_1} && \text{(3.5a)} \\
\text{s.t.} \quad & X = X^T \succ 0 && \text{(3.5b)} \\
& \text{trace}(B^T X B) < \gamma && \text{(3.5c)} \\
& \begin{pmatrix} -X + W^T W & A^T X - C^T \tilde{L}^T \\ XA - \tilde{L}C & -X \end{pmatrix} \prec 0 && \text{(3.5d)} \\
& \tilde{L} = XL && \text{(3.5e)}
\end{aligned}$$

Obviously, the equation $\tilde{L} = XL$ is still a bilinear term. So, the variable L has to be removed completely from the optimization problem. That is L has to be replaced with $\tilde{L}$ in the objective function:

$$\begin{aligned}
\min_{\gamma,\tilde{L},X} \quad & \gamma + \alpha \sum_{j=1}^{n} \omega_j^{(\mu)} ||\tilde{L}_{*j}||_{\ell_1} && \text{(3.6a)} \\
\text{s.t.} \quad & X = X^T \succ 0 && \text{(3.6b)} \\
& \text{trace}(B^T X B) < \gamma && \text{(3.6c)} \\
& \begin{pmatrix} -X + W^T W & A^T X - C^T \tilde{L}^T \\ XA - \tilde{L}C & -X \end{pmatrix} \prec 0. && \text{(3.6d)}
\end{aligned}$$

Remark 1. There are two characteristics, why this approach holds:

1. The substitution keeps the degrees of freedom, as the number of independent parameters is not reduced.
2. The relevant structure of L is kept in $\tilde{L}$, this means that the jth column in L is a zero column if and only if the jth column of $\tilde{L}$ is a zero column because the matrix X is positive definite.

Thus, we have a semidefinite program that is independent of L. The algorithm to solve the problem of sensor selection is then:

Algorithm 1. Algorithm with substitution: Relaxed sensor placement algorithm with linearization via substitution for discrete-time systems

1. Set the iteration count μ to zero and choose an appropriate value for the parameter α. Initialize the weights vector $\omega^{(0)}$ to $\omega_j^{(0)} = 1$ for $j = 1, \ldots, n$ and choose a sufficiently small $\varepsilon > 0$.
2. Solve the optimization problem (3.6).
3. Update the weights:
$$\omega_j^{(\mu+1)} = \frac{1}{||\tilde{L}_{*j}||_1 + \varepsilon}, \qquad j = 1, \ldots, n.$$
4. Terminate the iteration on convergence with $L = X^{-1}\tilde{L}$. Otherwise increase μ by 1 and return to Step 2.
5. Improve the choice of L by solving the optimization problem:
$$\min_{\gamma,\tilde{L},X} \quad \gamma \tag{3.7a}$$
$$\text{s.t.} \quad X = X^T \succ 0 \tag{3.7b}$$
$$\operatorname{trace}(B^T X B) < \gamma \tag{3.7c}$$
$$\begin{pmatrix} -X + W^T W & A^T X - \tilde{C}^T \tilde{L}^T \\ XA - \tilde{L}\tilde{C} & -X \end{pmatrix} \prec 0, \tag{3.7d}$$
where $\tilde{C} := C$ with predefined zero lines. The jth line is set to zero if the jth sensor was removed in the previous solution of Step 2 (i.e. if $||L_{*j}||_1 \ll \varepsilon$).

The weights vector $\omega^{(0)}$ could also be initialized as a zero vector in Step 2 of the algorithm. Thus, the first iteration would yield to a optimal observer gain matrix with no zero columns in L and $\omega^{(1)}$ would counteract the actual size of the column in an optimal non-sparse observer. With the current implementation it is possible that a sensor is set to zero in the first iteration because the column of L is smaller than the other columns. This is in general not equivalent to a bad performance of the sensor.

In Step 3 of Algorithm 1, $\tilde{L}$ or L could be used to perform the update on the weights. The reason why $\tilde{L}$ is used here is that we want to counteract the size of $\tilde{L}$ in the objective. As mentioned in Remark 1, it is equivalent to reduce either the non-zero columns of L or $\tilde{L}$.

In Step 3 of Algorithm 1, we introduced the variable ε to improve the reweighting with $\omega^{(\mu)}$ and avoid numerical problems, such as dividing by zero, as proposed in Candes et al. [2008].

Step 5 of Algorithm 1 is necessary, since the observer, that is calculated in the optimization problem (3.6), has a suboptimal observer gain matrix, because even the reweighted ℓ_1-norm in the objective minimizes the absolute value of the entries of $\tilde{L}$. After Step 5, the exact H_2-norm of the observer error system can be calculated.

The inequation $||L_{*j}||_1 \ll \varepsilon$ in Step 5 is equivalent to the statement that the jth column is counteracted with a maximum weight. This means that the corresponding sensor is removed. Therefore, the jth sensor is not considered when computing the improvement of L.

Remark 2. As written in equation (2.5), it is also possible to choose a constant γ and remove γ from the objective function. We choose the other way because it is on the one hand more difficult to estimate an appropriate γ before starting the optimization, and on the other hand the algorithm has no incentive to choose the best subset of k sensors, if another subset of k sensors also satisfies the bound on the H_2-norm.

3.3.2 Algorithm with Cone Complementarity Linearization

As mentioned before, we will consider the matrix inequality (3.3) to perform a cone complementarity linearization. Here, you can see the matrix inequality again:

$$\begin{pmatrix} -X + W^T W & A^T - C^T L^T \\ A - LC & -X^{-1} \end{pmatrix} \prec 0. \tag{3.8}$$

The following Lemma is necessary for formulating the algorithm with cone complementarity linearization.

Lemma 3. With $X \succ 0$ the following statements hold:

(i) The following two expressions are equivalent:

a) $KX = I$

b) $\text{trace}(KX) = n$ and $\begin{pmatrix} K & I \\ I & X \end{pmatrix} \succeq 0$

(ii) $\begin{pmatrix} K & I \\ I & X \end{pmatrix} \succeq 0 \Rightarrow \text{trace}(KX) \geq n.$

Proof:
Ad (i): Under condition of $X \succ 0$ the following steps are valid:
$\begin{pmatrix} K & I \\ I & X \end{pmatrix} \succeq 0$
$\Leftrightarrow$ Reformulation of the LMI with the Schur complement: $K - IX^{-1}I \succeq 0$
$\Leftrightarrow K - X^{-1} \succeq 0$. Postmultiplying with X leads to the next matrix inequality.
$\Leftrightarrow KX - I \succeq 0$
With this reformulation the proof of "a)$\Rightarrow$b)" is obvious since $KX = I$ implies $\text{trace}(KX) = \text{trace}(I) = n$ and $KX - I \succeq 0$ holds.

For the proof of "b)$\Rightarrow$a)" we will look at a characteristic of the matrix inequality $KX - I \succeq 0$ and use the premise $\text{trace}(KX) = \text{trace}(I) = n$.
$\text{trace}(KX - I) = 0$ is valid, because of $\text{trace}(KX) = \text{trace}(I) = n$.
$\Rightarrow KX - I$ has all eigenvalues equal to zero. Because assuming that $KX - I$ has a positive eigenvalue, which is possible since $KX - I \succeq 0$, would result in $KX - I$ having a negative eigenvalue so that the sum of all eigenvalues is zero again. That leads to a contradiction to the statement $KX - I \succeq 0$.

Ad (ii): Under condition of $X \succ 0$ the following step is valid:
$\begin{pmatrix} K & I \\ I & X \end{pmatrix} \succeq 0 \Leftrightarrow KX - I \succeq 0$ follows from the first steps of the proof of (i).
Since the trace of a matrix is the sum of its eigenvalues, it results that $\text{trace}(KX) \geq \text{trace}(I) = n$.

$\square$

The lemma is vital for our approach to the second algorithm because we now have the possibility to introduce a new variable K that behaves as the inverse of X. The LMI mentioned in Lemma 3 can be used as a constraint, while trace(KX) can be part of the objective function. With minimizing trace(KX), we will receive K as the inverse of X because trace$(KX) \geq n$ and $K = X^{-1}$ at the minimum, where trace$(KX) = n$.

Applying Lemma 3 to the matrix inequality (3.8), the optimization problem (2.6) results in:

$$\begin{aligned}
\min_{\gamma,L,X,K} \quad & \gamma + \alpha \sum_{j=1}^{n} \omega_j^{(\mu)} ||L_{*j}||_1 + \beta \text{ trace}(XK) && \text{(3.9a)} \\
\text{s.t.} \quad & X = X^T \succ 0 && \text{(3.9b)} \\
& \text{trace}(B^T X B) < \gamma && \text{(3.9c)} \\
& \begin{pmatrix} -X + W^T W & A^T - L^T C \\ A - LC & -K \end{pmatrix} \prec 0 && \text{(3.9d)} \\
& \begin{pmatrix} K & I \\ I & X \end{pmatrix} \succeq 0, && \text{(3.9e)}
\end{aligned}$$

where β is another parameter for adjusting the objective function. As we can see, linearizing the matrix inequality resulted in a bilinear term in the objective function. Our aim is to have semidefinite program so we need to replace the bilinear part. We could use a linearization method as seen in El Ghaoui et al. [1997] to solve such a problem. At a given point (X_{old}, K_{old}), a linear approximation of trace(XK) is

$$\phi_{lin}(X, K) = constant + \text{ trace}(XK_{old} + X_{old}K).$$

The optimization problem for the algorithm with cone complementarity linearization (CCL) is then:

$$\begin{aligned}
\min_{\gamma,L,X,K} \quad & \gamma + \alpha \sum_{j=1}^{n} \omega_j^{(\mu)} ||L_{*j}||_1 + \beta \text{ trace}(XK_{old} + X_{old}K) && \text{(3.10a)} \\
\text{s.t.} \quad & X = X^T \succ 0 && \text{(3.10b)} \\
& \text{trace}(B^T X B) < \gamma && \text{(3.10c)} \\
& \begin{pmatrix} -X + W^T W & A^T - L^T C \\ A - LC & -K \end{pmatrix} \prec 0 && \text{(3.10d)} \\
& \begin{pmatrix} K & I \\ I & X \end{pmatrix} \succeq 0. && \text{(3.10e)}
\end{aligned}$$

Algorithm 2. Algorithm with CCL: Relaxed sensor placement algorithm with cone complementarity linearization for discrete-time systems

1. Find a feasible solution for X in (3.4) and set $X_{old} = X$. If there are none, exit. Set the outer iteration count μ and the inner iteration count k to zero.
2. Set $K_{old} = X_{old}^{-1}$ and choose appropriate values for the parameters α, β, γ and δ. Initialize the weights vector $\omega^{(0)}$ to $\omega_j^{(0)} = 1$ for $j = 1, \dots, n$ and choose a sufficiently small $\varepsilon > 0$.
3. Solve the optimization problem (3.10).

4. If $|\mathrm{trace}(XK) - n| < \delta$, set $k = 0$ and go to Step 5, else set $k = k + 1$, $X_{old} = X$ and $K_{old} = K$ and go to Step 3.
5. Update the weights:

$$\omega_j^{(\mu+1)} = \frac{1}{||L_{*j}||_1 + \varepsilon}, \qquad j = 1, \dots, n.$$

6. Terminate the outer iteration on convergence. Otherwise increase μ by 1 and return to Step 3.
7. Improve the choice of L by solving the optimization problem (3.7). Define $\tilde{C} := C$ with predefined constant zero lines. The jth line is set to zero if the jth sensor was removed in the previous solution of Step 3 (i.e. if $||L_{*j}||_1 \ll \varepsilon$).

Remark 3. In Step 1 of Algorithm 2, there are several possibilities to find a feasible solution in (3.4). You could perform one iteration of (3.6) and use the generated X. In this case you could make the first update for the weights vector afterwards.

Whereas a faster method is to find X by calculating a non-sparse Kalman filter and the corresponding Lyapunov matrix of the observer error system. But in this case the matrix X has to change much during the next iterations to remove sensors. This means that a lot of iterations are needed because the linearization method punishes any changes in X and K. Therefore, the algorithm removes sensors only very slowly.

The objective function of this algorithm is divided up in three different parts, which are the bound on the H_2-norm γ, the ℓ_1-relaxation of the number of sensors, and the punishing term for the inverse matrix of X. These three terms have totally different functions in the optimization problem and have a diverse scale for each term. Therefore, it is hard to find the right parameterization for the terms.

Depending on the application of the algorithm, it is possible to reduce the complexity of the objective function. If only a reduced accuracy is needed for the result, one could take the term $\beta\ \mathrm{trace}(XK_{old} + X_{old}K)$ from the objective function and use it as a constraint. With an appropriate constant ζ the constraint could look like:

$$\mathrm{trace}(XK_{old} + X_{old}K) < 2n + \zeta.$$

As mentioned in Remark 3, a very small $\mathrm{trace}(XK_{old} + X_{old}K)$ slows down the removal of sensors. Therefore, it could even be an advantage to put this term into the constraints and allow greater inaccuracy in order to have a result with less sensors.

3.4 Algorithm for Optimal Sensor Placement for Continuous-Time Systems

In this section, we derive the approach for the algorithm with substitution for continuous-time systems. The algorithm is very similar to the one for discrete-time systems but the LMI characterization of the H_2-norm is different since the Lyapunov equation is of another form.

The following Lemma is like Lemma 2, but it is not necessary to show the second equivalence since the substitution even works when applied on the variables in the Lyapunov inequality.

Furthermore, in this thesis the cone complementarity linearization is used only for discrete-time systems as in the literature. Consequently, the augmented LMI for continuous-time systems is of no use for our approach.

Lemma 4. (As in Rieber [2006]) Consider a continuous-time system with transfer function

$$G(s) = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} := C(sI - A)^{-1}B.$$

The following two statements are equivalent:

(i) $||G(s)||_2^2 < \gamma$ and A is asymptotically stable.

(ii) There exists a positive definite symmetric solution P of the Lyapunov equation $A^TP + PA = -C^TC$ and $\text{trace}(B^TPB) < \gamma$.

Proof: (partly as in Smith [2010])

$||G(s)||_2^2 < \gamma$

$\Leftrightarrow \frac{1}{2\pi}\int_{-\infty}^{\infty} \text{trace}(G(j\omega)G(j\omega)^*)d\omega < \gamma$, which is the definition of the H_2-norm on the frequency domain. In the next step we use an equivalent definition on the time domain.

$\Leftrightarrow \int_0^{\infty} \text{trace}(g(t)^Tg(t))dt < \gamma$, where $g(t)$ is the impulse response of the system $G(s)$.

$\Leftrightarrow \text{trace}[B^T(\int_0^{\infty} e^{A^Tt}C^TCA^tdt)B] < \gamma$

$\Leftrightarrow \text{trace}[B^TPB] < \gamma$, where P is the observability Gramian (possible, since A is stable) which is defined as $P = \int_{t=0}^{\infty} e^{A^Tt}C^TCe^{At}dt$.

$\Leftrightarrow \text{trace}[B^TPB] < \gamma,\ P \succ 0$ and $A^TP + PA = -C^TC$

$\square$

We now consider the optimization problem (previously described in (2.6))

$$\min_{L,\gamma} \quad \gamma + \alpha \sum_{j=1}^{n} || \sum_{i=1}^{n} |L_{ij}| \, ||_{\ell_0} \tag{3.11a}$$

$$\text{s.t.} \quad ||\Sigma_e(L)||_2^2 < \gamma \tag{3.11b}$$

and apply the optimization problem to a continuous-time system (as described in (2.8))

$$\dot{e} = (A - LC)e + Bw \tag{3.12a}$$

$$z = We. \tag{3.12b}$$

Using Lemma 4, the resulting optimization problem is the following:

$$\begin{aligned}
\min_{\gamma,L,P} \quad & \gamma + \alpha \sum_{j=1}^{n} \omega_j^{(\mu)} ||L_{*j}||_{\ell_1} && \text{(3.13a)}\\
\text{s.t.} \quad & P = P^T \succ 0 && \text{(3.13b)}\\
& \mathrm{trace}(B^T P B) < \gamma && \text{(3.13c)}\\
& A^T P + PA - C^T L^T P - PLC + I = 0, && \text{(3.13d)}
\end{aligned}$$

where P is the solution of the Lyapunov equation as in Lemma 4 (ii). The constraint (3.13d) could also be written as a LMI, if the matrix P is replaced by a matrix X_{cont} which is defined as the continuous-time equivalent of the matrix X mentioned in the proof of Lemma 2.
We could now perform the substitution $\tilde{L} = PL$ as in Section 3.3.1. Consequently, Remark 1 holds also for this case and we replace L in the objective function with $\tilde{L}$. This results in the following optimization problem:

$$\begin{aligned}
\min_{\gamma,\tilde{L},P} \quad & \gamma + \alpha \sum_{j=1}^{n} \omega_j^{(\mu)} ||\tilde{L}_{*j}||_{\ell_1} && \text{(3.14a)}\\
\text{s.t.} \quad & P = P^T \succ 0 && \text{(3.14b)}\\
& \mathrm{trace}(B^T P B) < \gamma && \text{(3.14c)}\\
& A^T P + PA - C^T \tilde{L}^T - \tilde{L} C + I = 0. && \text{(3.14d)}
\end{aligned}$$

The algorithm with substitution is then formulated as:

Algorithm 3. Algorithm with substitution: Relaxed sensor placement algorithm with linearization via substitution for continuous-time systems

1. Set the iteration count μ to zero and choose an appropriate value for the parameter α. Initialize the weights vector $\omega^{(0)}$ to $\omega_j^{(0)} = 1$ for $j = 1, \ldots, n$ and choose a sufficiently small $\varepsilon > 0$.
2. Solve the optimization problem (3.14).
3. Update the weights:

$$\omega_j^{(\mu+1)} = \frac{1}{||\tilde{L}_{*j}||_1 + \varepsilon}, \qquad j = 1, \ldots, n.$$

4. Terminate the iteration on convergence with $L = P^{-1}\tilde{L}$. Otherwise increase μ by 1 and return to Step 2.
5. Improve the choice of L by solving the optimization problem:

$$\begin{aligned}
\min_{\gamma,\tilde{L},P} \quad & \gamma && \text{(3.15a)}\\
\text{s.t.} \quad & P = P^T \succ 0 && \text{(3.15b)}\\
& \mathrm{trace}(B^T P B) < \gamma && \text{(3.15c)}\\
& A^T P + PA - \tilde{C}^T \tilde{L}^T - \tilde{L} \tilde{C} + I = 0, && \text{(3.15d)}
\end{aligned}$$

   where $\tilde{C} := C$ with predefined zero lines. The jth line is set to zero if the jth sensor was removed in the previous solution of Step 2 (i.e. if $||L_{*j}||_1 \ll \varepsilon$).

The comments concerning $\tilde{L}$, L, ε, and ω made about the algorithm with substitution for discrete-time systems are also applicable to this algorithm.

4 Comparison of Different Algorithms

In this chapter, the algorithms of Chapter 3 are compared with state of the art algorithms. One is a greedy approach in two different versions as seen in Bach et al. [2010]. The other algorithm uses a branch and bound technique to provide an optimal solution which is also presented in the first part of this chapter. In the second part of the chapter, the algorithms are compared in terms of computation time and H_2-performance of the chosen subset of sensors. Additionally, the algorithms have to solve an application example consisting of a model of a power grid.

4.1 State of the art Algorithms

In the following, we discuss three algorithms that use the original optimization problem without any relaxation. Hence, these algorithms cannot use interior point methods as the two algorithms mentioned before.

These algorithms do not optimize the chosen subset of sensors and the observer gain matrix in parallel. In fact, the algorithms compute the H_2-norm of different subsets and compare this H_2-performance. Hence, there is no change in difficulty of the computation when the algorithms are implemented for discrete-time or continuous-time models.

4.1.1 Greedy Algorithm

The first state of the art algorithm, which is described, is a greedy approach that is very intuitive. The algorithm starts with all sensors and compares all subsets with $n-1$ sensors. After turning-off the worst sensor the algorithm consequently searches for the worst sensor of the remaining subset. Consequently, the greedy algorithm finds subsets of every size.

This greedy algorithm generates a fast solution and is able to give an overview of how many sensors are necessary for which H_2 performance. But as the algorithm removes all the sensors separately, it could not account for correlation between certain sensor subsets. This could lead to a very poor performance.

The algorithm searches for a subset of k sensors, where k is a predefined number with $k < n$. This means that the algorithm has to remove $n - k$ sensors.

Define: $J(k_1, k_2, \dots, k_m)$ is the H_2-norm of the error system with an optimal reduced observer which uses the sensors $k_1, k_2, \dots, k_m$ with $k_1, k_2, \dots, k_m \in \{1, 2, \dots, n\}$. The evaluation of this function could be done with the predefined system matrix C, in which one replaces every column except the columns $k_1, \dots, k_m$ with zero columns in order to remove the corresponding sensors. One way to evaluate the function is to solve the optimization problem (4.1) that is presented below.

$$\begin{aligned} \min_{\gamma,\tilde{L},X} \quad & \gamma && (4.1a)\\ \text{s.t.} \quad & X = X^T \succ 0 && (4.1b)\\ & \operatorname{trace}(B^T X B) < \gamma && (4.1c)\\ & \begin{pmatrix} -X + W^T W & A^T X - C^T \tilde{L}^T \\ XA - \tilde{L}C & -X \end{pmatrix} \prec 0 && (4.1d) \end{aligned}$$

Algorithm 4. Greedy algorithm: Greedy approach to the sensor selection problem as in Bach et al. [2010].

1. Initialize $M := \{1, 2, \ldots, n\}$ and iteration variable $i := 1$.
2. Compute $m = \arg\min_l J(M \setminus \{l\})$ with $l \in M$.
3. Define $M := M \setminus m$.
4. If $i = n - k$, end the algorithm with the solution M. If not, increase i by one and go back to Step 2.

M is the determined subset of sensors and $J(M)$ the corresponding value of the H_2 norm.

The greedy algorithm could also be implemented backwards. This means, the algorithm starts with zero sensors and then iteratively adds the best sensor. This implementation of the algorithm is called reverse greedy algorithm in the following.

4.1.2 Branch and Bound Algorithm

The next algorithm is a combinatorial algorithm, which finds the optimal subset of k sensors, where k is a predefined number with $k < n$. The algorithm is called branch and bound algorithm, because it arranges the sensors in branches and looks for an upper bound at the beginning of the algorithm. If the algorithm finds a good upper bound, it is possible that a lot of branches, i.e. tuples of sensors, will not have to be evaluated. The exact algorithm as it was implemented during the writing of this thesis and more details about the algorithm can be found in Narendra and Fukunaga [1977].

4.2 Runtime Comparison

For comparison discrete-time dynamical systems are generated randomly with B, C, and, W as identity matrices. The dynamic matrix A is chosen randomly with each entry uniformly, independently, and identically distributed on $[0, 1]$ and 80% zero elements to have a sparse structure as it is common in praxis for example when considering power grids.

The continuous-time systems were generated by converting the randomly generated discrete-time systems to a continuous-time model. The loss of the sparse structure during the converting process was not considered in this comparison.

Algorithms 1, 2 and 3 have objective functions in which the number of chosen sensors is controlled only indirectly by chosing the parameters α and β. For the test, the parameters are set so that about one fourth of the sensors were removed. However for these algorithms, the paramterization has little influence on the execution time if we assume convergence of the algorithms. Although a bad set of parameters could easily result in divergence, especially when handling the algorithm with CCL, and thus to no useful solution.

The branch and bound algorithm and the greedy algorithms had to choose subsets of $\frac{n}{2}$ sensors. In contrast to the other algorithms, the execution time is dependent on the number of chosen sensors, since both start with all sensors and consequently remove sensors. When choosing very few sensors, the branch and bound algorithm has a very large execution time. The algorithm could even be outperformed by an algorithm, which tries all possible tuples, because the upper bound used in the branch and bound algorithm would be too big and therefore nearly useless.

The formulation of the optimization problem for the algorithms with substitution and the algorithm with CCL is a semidefinite programming problem, since the constraints are linear matrix inequalities (see also in Vandenberghe and Boyd [1999]) and the objective is a convex function and can even be recast as a linear function. This allows us to use interior point methods which have polynomial complexity as can be seen in Sturm [1999]. The reweighted ℓ_1 relaxation usually needs only few steps to converge when the parameter ε is chosen appropriate (as in Candes et al. [2008]).

In the (reverse) greedy algorithm, the number of computations of an optimal observer gain matrix and the corresponding H_2-norm increases with n^2, where n is the number of states, assuming that it has to reduce a certain percentage of the senors. The optimal observer gain matrix could be evaluated as a Kalman filter, which has the complexity of n^3. So the greedy algorithm also has polynomial complexity.

The reverse greedy algorithm computes the optimal observer gain matrix as often as the greedy algorithm when choosing a subset of $\frac{n}{2}$ sensors. Hence, the execution time of these two greedy algorithms is the same. In the following, we do not distinguish between these two algorithms when considering the runtime.

The branch and bound algorithm is NP-hard because at a worst-case behavior it has to compute more than all possible subsets of k sensors, which are $\binom{n}{k}$. For this reason the branch and bound algorithm was mainly used here to have an optimal subset in order to compare the performance of the other algorithms. This comparison can be seen in the next section.

For each size, we generated 10 random examplary systems as described above. Every algorithm had to find the optimal sensors for all these systems. In Figure 4.1 we see the execution time in seconds. The displayed execution time is the average of the different computations.

The dashed lines show the greedy algorithm and the branch and bound algorithm with a slightly changed implementation. While the solid lines were produced with algorithms that used an implementation of the `kalman` function in Matlab, the dashed lines solved the optimization problem (4.1), which was implemented with SeDuMi (Sturm [1999]) and YALMIP (Lofberg [2004]). Both of the implementations achieve the same results, but the implementation with SeDuMi and YALMIP needs more execution time and allows a better comparison with the other three algorithms that also use SeDuMi and YALMIP.

As we can see, the greedy algorithm, which uses the `kalman` function method of Matlab, is the fastest algorithm. The algorithm with substitution and the algorithm with CCL are very similar, but the algorithm with CCL needs more execution time, which could be the effect of the second loop in the algorithm. Moreover, the algorithm with CCL quite often diverged which was not considered for the execution time here.

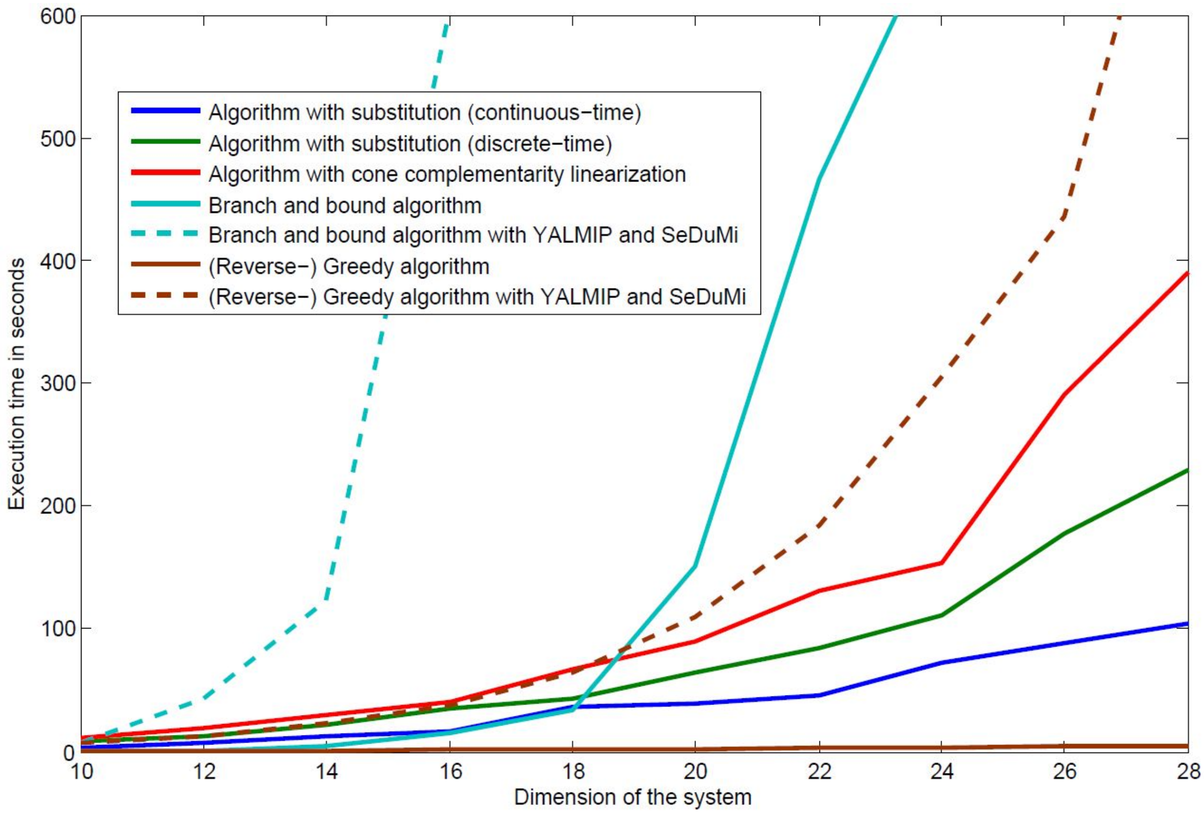

Figure 4.1: Comparison of the execution time for different dimensions of the system

The algorithm with substitution, which solves problems with continuous-time systems, is faster than its discrete-time equivalent which could easily be explained by the smaller size of the constraint.

Since the algorithm with substitution outperforms the greedy algorithm with the SeDuMi implementation, it could be possible that a more advanced implementation of the algorithm with substitution shows a similar execution time as the greedy algorithm.

The largest execution time was needed by the two implementations of the branch and bound algorithm. Its execution time rises quickly with increasing dimension of the system as the NP-hardness of the problem suggests.

4.3 Performance Comparison

To test the performance of the relaxed and the greedy algorithms, we compared the generated results with the global optimum evaluated by the branch and bound algorithm. In this comparison the algorithms solved 100 generated problems, which were computed as described in the section before. Each of the generated systems had twelve states.

The algorithms with substitution and the algorithm with CCL were parametrized ten times and solved ten optimization problems with each parameterization. The greedy algorithms were set up, so that they chose subsets with the size from one sensor to the size of eleven sensors.

No. of chosen sensors	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$
Perf. gap of the alg. with substitution	–	3.65%	0.58%	1.19%	1.47%	0.86%	1.17%	0.23%	0.86%	–	–	–
No. of solutions with k sensors	0	1	4	16	27	18	21	6	2	0	0	0
Perf. gap of the alg. with CCL	–	–	–	–	0.00%	0.15%	0.00%	0.20%	0.09%	0.00%	0.00%	0.00%
No. of solutions with k sensors	0	0	0	0	1	4	4	19	17	13	5	32
Perf. gap of the greedy alg.	7.03%	2.76%	1.40%	0.50%	0.21%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	–
No. of solutions with k sensors	10	10	10	10	10	10	10	10	10	10	10	0
Perf. gap of the reverse greedy alg.	0.00%	1.39%	0.85%	1.21%	0.85%	1.60%	1.19%	0.95%	0.48%	0.00%	0.00%	–
No. of solutions with k sensors	10	10	10	10	10	10	10	10	10	10	10	0

Table 4.1: Performance comparison of the different algorithms

Table 4.1 shows the performance gap of the chosen subsets of sensors in comparison with the optimal subset of the same size chosen by the branch and bound algorithm. The performance gap is defined as the percentaged difference of the H_2-norms of the observation error system and we compare the suboptimal subset with an optimal subset of sensors. The performance gap is shown in dependence on the number of the chosen sensors. Additionally, we can see how often the algorithms chose a subset of that size. This is in particular interesting for the algorithm with CCL and for the algorithm with substitution because the parameterization can influence the number of chosen sensors only indirectly.

The algorithm with substitution and the algorithm with CCL had five attempts each, in which the algorithms did not converge, this information is left out in Table 4.1 so that the table is easier to read.

The algorithm with substitution for continuous-time systems showed similar results as the algorithm with substitution for discrete-time systems. Hence, they are not considered separately in this section.

The algorithm with CCL has a good performance with a maximal average performance gap of 0.2%. Yet, as we can see, the algorithm did not manage to choose small subsets of sensors with the chosen parameters. This observation was made in several other tests, too. The algorithm with CCL seems to be numerically instable when trying to remove many sensors. This could be due to the cone complementarity linearization in (3.10a) which always tries to keep the Lyapunov matrix close to the one in the iteration before (i.e. trace$(XK_{old} + X_{old}K)$ is smallest if $X = K_{old}^{-1}$ and $K = X_{old}^{-1}$). This way a change in the system is being punished and a change in the error system is in particular an observer which uses one sensor less.

Furthermore, we can interpret the 32 computations, in which the algorithm with CCL found subsets with twelve sensors, also as no useful solutions since no new information about the sensor placement problem was provided. So altogether the algorithm with CCL terminated 37 times without generating a useful solution. In summary, one can say that this algorithm is hard to parameterize.

The greedy algorithm has the advantage that one can decide a priori how many sensors are selected. Moreover, the greedy algorithm is not iterative, so the computation cannot end because of divergence. As one can see, the greedy algorithm often finds the optimal solution if few sensors have to be removed. But the performance gap increases when the algorithm has to choose less sensors.

At these generated systems, the reverse greedy algorithm shows a good performance also for small subsets. When considering systems that are instable, the algorithm could have problems to choose the first sensor because the observer error system could also be instable when using only one sensor.

The performance gap of the algorithm with substitution is similar for every size of the subset. Moreover, we see that parameters can be found so that the algorithm with substitution chooses a small subset of sensors. The approach of the algorithm with substitution is similar to the approach with cone complementarity linearization but seems to avoid the complicated choice of parameters and the difficulties with convergence when we search for a little number of sensors.

4.4 Application Example

In this section, a high-voltage power grid with seven nodes is considered where exclusively synchronous generators are connected. The aim is to observe the phase angles and phase velocities, i.e. frequencies, at the different nodes. For that purpose phase measurement units (PMUs), which can measure the phase angle but not the phase velocity, are installed at certain nodes. Since these PMUs are very expensive, as few as possible should be used.

The grid is of the form that is presented in Figure 4.2.

The corresponding swing equation that describes the behavior of the phase angles θ_i in a coordinate system rotating with 50 Hz is the following:

$$M_i\ddot{\theta}_i + D_i\dot{\theta}_i + D_i\frac{\sum_{j=1}^{7} p_j}{\sum_{j=1}^{7} D_j} = p_i - \sum_{\substack{j=1\\ j\neq i}}^{7} \frac{u_i u_j}{x_{ij}} \cdot \sin(\theta_i - \theta_j) \quad \text{for } i = 1, \dots, 7, \tag{4.2}$$

where θ_i is the phase angle at the different nodes, u_i the voltage, x_{ij} the impedance of the line from node i to node j, p_i the positive or negative power input at the node i, D_i the damping at the synchronous machines, and M_i the moment of inertia of each generator.

We consider a normalized power grid model with normalized voltages $u_i = u_j = 1$ for all i, j.

The parameters for the grid are chosen randomly out of the following intervals according to Kundur et al. [1994]. The intervals for the parameters are: $M_i \in [5, 10]$, $D_i \in [1, 2]$, $p_i \in [-1, 1]$, and $x_{ij} \in [0.1, 1]$ for all i, j.

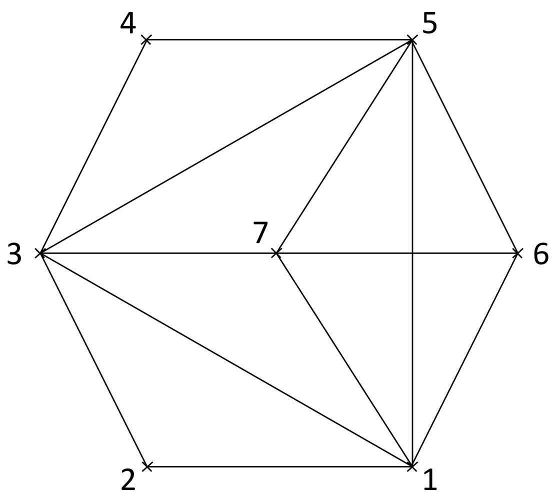

Figure 4.2: Symbolic illustration of the grid

To make the algorithms applicable, equation (4.2) needs to be linearized and the equation has to be written as a system of ordinary differential equations of first order. Furthermore, the model has to be converted from the displayed continuous-time to a discrete-time model.

We will linearize the equation by using a Taylor approximation of first order around the steady state of the model. To get to know the steady state phase angles θ^* we do a load flow calculation by solving the following equation ($u_i = u_j = 1$):

$$D_i \frac{\sum_{j=1}^7 p_j}{\sum_{j=1}^7 D_j} = p_i - \sum_{\substack{j=1 \\ j\neq i}}^{7} \frac{1}{x_{ij}} \cdot \sin(\theta_i^* - \theta_j^*) \quad \text{for } i = 1, \dots, 7, \tag{4.3}$$

which is the same as equation (4.2) under the assumption that $\ddot{\theta}_i = \dot{\theta}_i = 0$ for all $i \in \{1, \dots, 7\}$. The term $D_i \frac{\sum_{j=1}^7 p_j}{\sum_{j=1}^7 D_j}$ is balancing the equation in case of a non-zero overall power input i.e. $\sum_{i=1}^7 p_i \neq 0$.

The Taylor approximation around the calculated steady state θ^* is then:

$$M_i \triangle\ddot{\theta}_i + D_i \triangle\dot{\theta}_i + D_i \frac{\sum_{j=1}^7 p_j}{\sum_{j=1}^7 D_j} = p_i - \sum_{\substack{j=1 \\ j\neq i}}^{7} \frac{1}{x_{ij}} \cdot [\sin(\theta_i^* - \theta_j^*) + \cos(\theta_i^* - \theta_j^*)(\triangle\theta_i - \triangle\theta_j)], \tag{4.4}$$

which is the same as:

$$M_i \triangle\ddot{\theta}_i + D_i \triangle\dot{\theta}_i = -D_i \frac{\sum_{j=1}^7 p_j}{\sum_{j=1}^7 D_j} + p_i - \sum_{\substack{j=1 \\ j\neq i}}^{7} \frac{1}{x_{ij}} \cdot \sin(\theta_i^* - \theta_j^*) - \sum_{\substack{j=1 \\ j\neq i}}^{7} \frac{1}{x_{ij}} \cdot \cos(\theta_i^* - \theta_j^*)(\triangle\theta_i - \triangle\theta_j). \tag{4.5}$$

Obviously we could eliminate the term $-D_i \frac{\sum_{j=1}^7 p_j}{\sum_{j=1}^7 D_j} + p_i - \sum_{\substack{j=1 \\ j\neq i}}^{7} \frac{1}{x_{ij}} \cdot \sin(\theta_i^* - \theta_j^*)$ because it is zero by definition of θ^*. So with equation (4.4) and (4.5) we look at the modelled system in dependence of $\triangle\theta$, which is the difference of the phase angle to its steady state. The resulting equation is

$$M_i \triangle\ddot{\theta}_i + D_i \triangle\dot{\theta}_i = -\sum_{\substack{j=1 \\ j\neq i}}^{7} \frac{1}{x_{ij}} \cdot \cos(\theta_i^* - \theta_j^*)(\triangle\theta_i - \triangle\theta_j). \tag{4.6}$$

Since this equation is a second order differential equation, we need to rewrite it as a system of 1st order which can then be discretized.

$$\begin{pmatrix} \operatorname{diag}(M_i) & 0 \\ 0 & I \end{pmatrix} \dot{\begin{pmatrix} \frac{d}{dt}\triangle\theta \\ \triangle\theta \end{pmatrix}} = \begin{pmatrix} \operatorname{diag}(-D_i) & L(\theta^*) \\ I & 0 \end{pmatrix} \begin{pmatrix} \frac{d}{dt}\triangle\theta \\ \triangle\theta \end{pmatrix}, \tag{4.7}$$

where $\triangle\theta = \operatorname{vec}(\triangle\theta_i)$ and $L(\theta^*) = (l_{ik})_{i,k=1,\ldots,7}$ is

$$\begin{aligned} l_{ik} &= -\sum_{\substack{j=1 \\ j\neq i}}^{7} \frac{1}{x_{ij}} \cdot \cos(\theta_i^* - \theta_j^*) \text{ if } i = k \\ l_{ik} &= \frac{1}{x_{ik}} \cdot \cos(\theta_i^* - \theta_k^*) \text{ if } i \neq k. \end{aligned}$$

Since we did not allow coefficients on the left side of our equation in (2.1a), we have to remove the matrix $\left(\begin{smallmatrix} \operatorname{diag}(M_i) & 0 \\ 0 & I \end{smallmatrix}\right)$. This results in the following equation:

$$\dot{\begin{pmatrix} \frac{d}{dt}\triangle\theta \\ \triangle\theta \end{pmatrix}} = \begin{pmatrix} \operatorname{diag}(-\frac{D_i}{M_i}) & (\operatorname{diag}(M_i))^{-1}L(\theta^*) \\ I & 0 \end{pmatrix} \begin{pmatrix} \frac{d}{dt}\triangle\theta \\ \triangle\theta \end{pmatrix}. \tag{4.8}$$

This equation is still a continuous-time model. State of the art PMUs have a sample rate of 40 msec. By converting the model with that step size and assuming the system matrices B and W as identity matrices we have a proper description of the system for the developed algorithms. The only thing that still has to be defined is the matrix C. This matrix needs to specify that the first seven states of the system can not be observed by a sensor since the PMUs can only measure the phase angle, not the phase velocity. Consequently, C is defined as $C := (0_7 I_7) \in \mathbb{R}^{7\times 14}$.

Starting with that system, the branch and bound algorithm chose the sensors 1, 3, and 5 as displayed in Figure 4.3.

The algorithms with substitution (both for discrete-time and for continuous-time systems) and the reverse greedy algorithm found the same optimal subset as the branch and bound algorithm. The H_2-norm of the system with PMUs at node 1, 3, and node 5 is 27.1. The greedy algorithm did not determine this subset but proposed to place the PMUs at the nodes 2, 3, and 5, which would result in a H_2-norm of 27.4. This means, the greedy algorithm results in a performance gap of about 1.1%.

The algorithm with CCL did not find a subset of sensors that contains just a few sensors. Over 50 parameterizations were tried for this algorithm. But with the chosen parameters, the algorithm with CCL either chose all seven sensors or did not converge and hence stopped with no result.

The reason why the algorithm with CCL did not find appropriate solutions could be that the discrete-time system is very ill-conditioned. The dynamics matrix has a lot of eigenvalues with

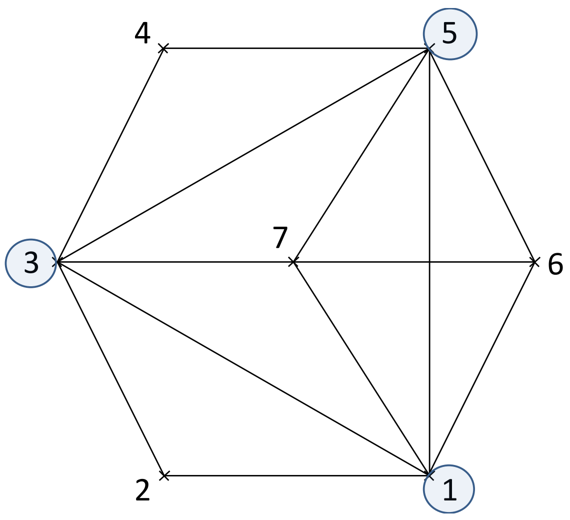

Figure 4.3: Symbolic illustration of the grid with the optimal subset of two sensors

absolut values close to one because of converting the continuous-time system with a very small sample rate. This means, the system is very close to instability without an observer. To compute the H_2 norm of the observer error system, this system has to be stable. Consequently a solution with a small subset of sensors is very close to the constraints, which could be the problem for the algorithm with CCL.

4.5 Concluding Comparison of the Algorithms

Algo.	**Performance**	**Execution-time**	**Correlation**	**Parameterization**
Algo. with substitution	$+$	$+$	$+$	$+-$
Algo. with CCL	$+$	0	$+$	$--$
Branch and bound algo.	$++$	$--$	$++$	$++$
Greedy algorithm	$+$	$+-$	$--$	$++$
Reverse greedy algo.	$+$	$+-$	$--$	$++$

Table 4.2: Comparison of the different algorithms

In Table 4.2 we see an overview of the advantages of the different algorithms. One fact that stands out is that the algorithms with substitution are the only algorithms that have no severe lack.

The algorithms with substitution select the right sensors for different sizes of the subset and especially the algorithm for continuous-time systems has a short execution time. The algorithms search for the optimal subset with respect to correlation between certain sensors since they can remove several sensors at a time. The parameterization of the algorithms is simple when we search for a sensor subset with a bounded observer error or a trade-off of the number of sensors to the observer error. In contrast to the branch and bound or the (reverse) greedy algorithm, it is not possible to exactly tell the algorithms how many sensors they should select.

The algorithm with cone complementarity linearization has the same benefits as the algorithm with substitution, except for the good execution-time and the easy paramterization. The algorithm with CCL has a second loop inside of the iterative reweighted ℓ_1-minimization which slows the algorithm down. The great disadvantage of this algorithm is that it is very hard to parameterize the algorithm correctly. In the application example no appropriate parameterization was found although over 50 different sets of parameters were tried. Nevertheless, the algorithm only lead to useless results.

The branch and bound algorithm generates the optimal solution and is easy to parameterize. In contrast to the other algorithms, the branch and bound algorithm has a non-polynomial complexity and therefore an execution-time that is not usable in praxis.

The greedy algorithm is an intuitive and fast approach which generates usually a good solution. However, the shortsighted approach cannot consider the importance of certain sensors in small subsets. Another application example was generated where the greedy algorithm first removes the sensor at the highest branched node, although this sensor would belong to the optimal subset at the end. It seems that highly branched nodes are easy to observe with a lot active sensors so the sensor at this node is removed at the beginning of the algorithm. However, with few active sensors, the highly branched sensors are important to observe the states at other nodes. It is possible that this property is more important when considering larger networks and did not affect the results in Section 4.3. However, in our examples the greedy algorithm shows a good performance and generates an overview of the H_2-performance for different sizes of subsets.

The reverse greedy algorithm is also a fast algorithm with a good solution to our examples. In contrast to our examples, more complex grids with instable parts can not be solved by the reverse greedy algorithm because it needs an observer with only one sensor that has a stable observer error system. If such a sensor could not be found, the algorithm has no starting point and ends with no result. As the other greedy approach, the reverse greedy algorithm can only perform shortsighted choices and cannot consider correlation of the sensors. Therefore, it is possible that a very poor solution is provided.

When solving a sensor placement problem in praxis, the algorithm with substitution and the (reverse) greedy algorithm should be chosen. The (reverse) greedy algorithm can provide a fast overview of the performance of the desired number of sensors and a first subset. After that, the algorithm with substitution can be used to improve the chosen subset or confirm its performance.

5 Conclusion and Future Directions

In this thesis, we considered a H_2-optimal sensor placement for linear dynamical systems for an infinite time horizon. The model of the system can either be a discrete-time or a continuous-time model. Furthermore, the influence of errors on certain states of the system can be parameterized as well as the subset of all possible sensors.

The problem of sensor placement arises in various fields of application, for example at observing power grids as described in chapter 4. The aim is here to reduce costs for expensive sensors and still have the best possible observer system for a given number of sensors.

The original problem of sensor placement is a non-convex problem, which can either be approached by combinatorial algorithms like the branch and bound algorithm or with greedy algorithms in different variations, as described in Chapter 4. In contrast to these methods, the main algorithms in this thesis use another approach. The presented algorithms optimize the structure, i.e. the sensor placement, and the observer gain matrix in parallel. Due to non-convexities, which arise when counting the used sensors and when calculating the H_2-norm with an unknown observer, we had to reformulate the optimization problem. The discrete ℓ_0-norm was relaxed with an iterative reweighted ℓ_1-norm. The H_2-norm was written with a LMI characterization, which simplified replacing the non-convex bilinear part either with a substitution or with a cone complementarity linearization. The proposed approaches were then developed into algorithms and implemented to efficiently and accurately solve the sensor placement problem.

The mentioned algorithms were tested in Chapter 4. The first test was a comparison of the execution time depending on the system size. The second test checked on the H_2-performance of the chosen sensor subsets of the different algorithms, while comparing the ability to parameterize the algorithms so that different numbers of sensors were selected. Additionally, a practical example was solved. Here, the algorithms had to determine an optimal sensor subset. The performance of the chosen subsets were compared. The example had slightly other requirements for the algorithms since the problem was more ill-conditioned than the randomly generated examples. This revealed further difficulties when parameterizing the algorithm with cone complementarity linearization.

The conclusion of all the tests is that the algorithm with substitution is one of the best of the presented algorithms, since it has no severe lacks, e.g. execution time or parameterization, and a satisfying performance. Similar to the greedy algorithm, the algorithm with substitution for continuous-time systems has a very short execution time despite of its not advanced implementation. The greedy approach and the reverse greedy approach both lead to good solutions at the considered examples and are the fastest algorithms. The shortsighted procedure does not influence the performance at the described examples.

Future research concerning this topic could expand this approach into the following directions. The assumption that the system matrix A is completely known may not be fulfilled. It is more realistic to assume that one knows A with certain inaccuracies. Another possibility is that a non-linear system can be estimated in a defined sector and that the algorithms use bounds of this sector to derive an optimal subset of sensors.

Another direction of future research could be the expansion of the model to actuator placement which is the dual problem to the one described here. Moreover, for actuator placement usually the H_∞-norm is used. Consequently, the approach would have to be adapted to the LMI characterization of this norm.

6 Appendix - Mathematical Definitions and Notation

6.1 ℓ_0-Vector-Norm

The ℓ_0-norm for vectors has more than one commonly used definition like $||x||_0 := \lim_{p\to 0} ||x_i||_p^p$ for $p \in \mathbb{R}^+$ and $x \in \mathbb{R}^n$. Precisely said, the ℓ_0-norm refers to the number of non-zero elements in a vector. Therefore it is straightforward to define also:

$$||x||_0 := \sum_{i=1}^{n} |\text{sign}(x_i)|$$

with

$$\text{sign}(x_i) := \begin{cases} -1 & \text{for } x_i < 0 \\ 0 & \text{for } x_i = 0 \\ 1 & \text{for } x_i > 0 \end{cases}$$

Since the positive scalability (i.e. $|\alpha|\ ||x||_0 = ||\alpha x||_0$ for $\alpha \in \mathbb{R}^+$) is not satisfied in general, the ℓ_0-norm is in fact not a real norm, however it is often referred to as the 0-norm.

The ℓ_0-norm can be relaxed using a ℓ_1-norm, which is defined as follows:

$$||x||_1 := \sum_{i=1}^{n} |x_i|.$$

6.2 H_2-System-Norm

Define $G(s) = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} := C(sI - A)^{-1}B$ as the state-space realization of the transfer matrix, $Q_C := \int_0^\infty e^{At}BB^Te^{A^Tt}dt$ as the controllability Gramian and $Q_O := \int_0^\infty e^{A^Tt}CC^Te^{At}dt$ as the observability Gramian for continuous-time systems.

The H_2-norm of a system $G(s)$ is defined as (according to Scherer and Weiland [2000]):

$$||G(s)||_2 := \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} \text{trace}(G(j\omega)G(j\omega)^*)d\omega}. \tag{6.1}$$

This system norm is used to specify the performance of the systems in this thesis, because it is the typical norm for observers of dynamical systems, e.g. the well-established Kalman filter (see Herrnberger [2010]). A typical interpretation of the norm is to see $||G(s)||_2$ as the total output energy of an impulse response of the system(as in Scherer and Weiland [2000]).

The relation of the H_2-norm to the Gramians for a system $G(s) = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix}$ is

$$||G||_2^2 = \text{trace}(CQ_C C^T) = \text{trace}(B^T Q_O B).$$

The definition of the H_2-norm for discrete-time systems is very similar. Define $G(z) = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} :=$ $C(zI - A)^{-1}B$ as the state-space realization of the transfer matrix for a discrete-time system. Consequently, the H_2-norm for discrete-time systems is defined as:

$$||G(z)||_2 := \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi} \text{trace}(G(e^{j\omega})G(e^{j\omega})^*)d\omega}. \tag{6.2}$$

6.3 Matrices and LMIs

Next, we will give some notational specifications of this thesis. Given a matrix $L \in \mathbb{R}^{n\times n}$ we will denote L^{-1} and L^T as the inverse and the transpose of the matrix. L_{*j} with $j \in 1, 2, \ldots, m$ defines the jth column of the matrix L.

A positive or negative definite (semidefinite) matrix L is written as $L \succ 0$ ($L \succeq 0$) or $L \prec 0$ ($L \preceq 0$) respectively, while $>, <, \leq, \geq$ denote an elementwise comparison.

Bibliography

F. Bach, S.D. Ahipasaoglu, and A. d'Aspremont. Convex relaxations for subset selection. *ArXiv preprint ArXiv:1006.3601*, 2010.

E.J. Candes, M.B. Wakin, and S.P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

L. El Ghaoui, F. Oustry, and M. AitRami. A cone complementarity linearization algorithm for static output-feedback and related problems. *Automatic Control, IEEE Transactions on*, 42 (8):1171–1176, 1997.

M. Fazel. *Matrix rank minimization with applications.* PhD thesis, Stanford University, 2002.

M. Herrnberger. Moderne Methoden der Regelungstechnik III. Lecture at the Institute of Automatic Control in SS 2010, Technichal University Munich, Germany, October 2010.

S. Joshi and S. Boyd. Sensor selection via convex optimization. *Signal Processing, IEEE Transactions on*, 57(2):451–462, 2009.

P. Kundur, N.J. Balu, and M.G. Lauby. *Power system stability and control*, volume 4. McGraw-hill New York, 1994.

J. Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289. IEEE, 2004.

Y. Mo, R. Ambrosino, and B. Sinopoli. Sensor selection strategies for state estimation in energy constrained wireless sensor networks. *Automatica*, pages 1330–1338, 2011.

P.M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *Computers, IEEE Transactions on*, 100(9):917–922, 1977.

J. Rieber. Lecture robust contol. Lecture at the Institute for Systems Theory and Automatic Control in WS 2006/2007, University of Stuttgart, Germany, October 2006.

C. Scherer and S. Weiland. Linear matrix inequalities in control. *Lecture Notes, Dutch Institute for Systems and Control, Delft, The Netherlands*, 2000.

S. Schuler, U. Münz, and F Allgöwer. Decentralized state feedback control for interconnected process systems. In *AdChem*, 2012.

R. Smith. Robust control & convex optimization. Lecture at the department of electrical & computer engineering in SS 2010, University of California, Santa Barbara, USA, June 2010.

J.F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

L. Vandenberghe and S. Boyd. Applications of semidefinite programming. *Applied Numerical Mathematics*, 29(3):283–299, 1999.

Danksagung

An dieser Stelle möchte ich mich recht herzlich bei Herrn Dr. Ulrich Münz für die intensive Betreuung bedanken, die teilweise viel Zeit in Anspruch nahm. Ohne die zahlreichen fachlichen Diskussionen und interessanten Anregungen wäre diese Arbeit so nicht zustande gekommen.

Außerdem bedanke ich mich auch bei der Siemens AG für die Ermöglichung der Arbeit und die finanzielle Unterstützung währenddessen.

Herrn Prof. Dr. Tobias Damm danke ich für die sehr angenehme Kooperation und die sehr hilfsbereite Beantwortung bei allen fachlichen oder organisatorischen Fragestellungen.

Schließlich bedanke ich mich noch bei Herrn Prof. Dr. Lars Grüne für die Bereitschaft, die Arbeit des Zweitgutachters zu übernehmen.

Erklärung

Hiermit erkläre ich, dass ich diese Bachelorarbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt habe und dass ich diese Arbeit nicht bereits zur Erlangung eines akademischen Grades eingereicht habe.

Bayreuth, 17. August 2012