NUMERICAL SOLUTION OF AN IDENTIFICATION PROBLEM IN ELECTROMYOGRAPHY

TOBIAS SPROLL* AND ANTON SCHIELA*

Abstract. In medical treatment, it can be necessary to identify specific motor units in muscles from measurements on the skin. In [24], we already derived and analyzed an optimization problem suitable for such an identification. This work presents an optimization algorithm that exploits the specific mathematical structure of this problem. Our algorithm is based on a Newton line-search approach in function space with inexact function evaluations due to adaptive numerical quadrature. We present a global convergence result for this method and describe in detail its algorithmic implementation. Finally, we illustrate its practical performance by a numerical study.

Key words. optimization in function spaces, inexact function evaluations, electromyography

AMS subject classifications. 49M41, 65N30, 65N21

1. Introduction. In the human body, various bioelectric sources arise, which we can measure indirectly as electric potentials on the skin. A fundamental question in medical research and diagnosis is: can we identify the bioelectric source from these surface measurements? Such identification of bioelectric activity is needed in many fields of medicine, e.g., in measuring brain activity (EEG) or cardiac activity (ECG). Correspondingly, a lot of work has been performed to develop tools for computational assistance, see, e.g., [11] and references therein for EEG. In general, refinements of Tychonov regularization techniques are applied to the spatial problem.
When measuring potentials caused by muscle activity, the corresponding technique is called surface electromyography (sEMG). Potential applications arise in research (Which motor unit is responsible for which movement?) or pre-operative planning (Where is the location of important nerves which should not be harmed in operations?). Similar techniques as described above have also been applied to electromyography measurements [18, 26, 27]. The presented techniques yield a smooth, distributed reconstruction of sources, which is appropriate in applications with smoothly distributed sources within the tissue. These approaches mainly solve spatial problems, not taking into account the spatio-temporal structure of the problem.
However, bioelectrical sources responsible for muscle activity have a special structure. A muscle consists of muscle fibers organized in bundles, so-called motor units. Muscle activity now means that the peripheral nervous system activates one or more motor units. This activation causes electrical signals, so-called action potentials, to propagate along the muscle fibers of the motor unit. We can describe the action potentials as concentrated line measures that move along some trajectory, as time progresses, cf. [19, 24]. Approaches that directly attempt to process the spatio-temporal information and exploit the high temporal resolution of surface EMG are less common. A notable exception is [20]. The authors consider a regularized least-squares approach for fitting the EMG signal by a linear combination of a moderate number of analytically predefined and prelocated waveforms. This enables fast computations in real time with modest accuracy.
In [24], we presented an alternative method using an adjoint approach. We already showed that the corresponding optimization problem has at least one solution. Furthermore, we derived first-order optimality conditions building the foundation of the numerical algorithm. Our current paper discusses the numerical solution of the problem presented in [24].
We organized this paper as follows. First, we shortly recall the identification problem and discuss the mathematical model, the optimization problem, and its first-order optimality conditions.

*Mathematical Institute, University of Bayreuth, 95447 Bayreuth, Germany, (tobias.sproll@uni-bayreuth.de, anton.schiela@uni-bayreuth.de).

Then we elaborate our algorithm to solve the problem numerically. We rely on well-known algorithmic techniques, specifically an augmented Lagrangian approach to treat the present equality constraints and an SQP line-search approach, cf., e.g., [22, 1]. However, due to the special structure of our problem, some algorithmic adjustments are necessary. In particular, we use an adaptive quadrature rule to evaluate the objective functional, cf. [24], which changes the evaluation method from step to step. Thus, we interpret the evaluation as an inexact evaluation of continuous quantities rather than an exact evaluation of discrete quantities. Inexact function evaluations are not uncommon in nonlinear optimization, and the topic is studied in different settings, cf., e.g., [13, 14, 3, 6, 7, 28]. Although [3] studies the global convergence aspect in a Trust-Region framework, we can adapt the presented technique and condition to our setting and show global convergence. The following section explains the implementation details, in particular discretization, adaptive quadrature and their role in our SQP-Newton algorithm. Finally, we discuss a numerical exammple.

2. The Identification Problem. In this section, we define the underlying identification problem. This is an inverse problem in the following sense: given a surface EMG measurement at n electrodes, we are interested in the motor unit responsible for the measurement. To identify this motor unit numerically, we need a mathematical model that simulates a surface EMG measurement for a given motor unit. We already discussed this model in detail in [24]. Thus, we only revisit the key aspects of the model and introduce the optimal control problem afterward. Furthermore, we recall the existence of solutions and the first-order optimality conditions.

2.1. Mathematical Setting. Throughout this paper, $\Omega \subset \mathbb{R}^3$ is a domain with a sufficiently smooth boundary representing some body part. With Ω_M, we denote the sub-domain representing the muscle tissue where the motor unit is located. For simplicity, we assume that the motor unit consists of only one muscle fiber. We use a regular curve $u \in H^2(-1,1,\Omega)$ to represent such a single-fibered motor unit. When a motor unit is activated, two electrical signals, so-called action potentials, propagate from the neuro-muscular junction towards both ends of the motor unit. For each time instant t, we model the action potentials by a line measure $\rho_i(t) \in \mathcal{M}(\bar{\Omega})$, where $\mathcal{M}(\bar{\Omega})$ is the Banach space of Radon measures on $\bar{\Omega}$ which is isomorphic to the dual $C(\bar{\Omega})^*$. These propagating signals generate an electrical potential $\Phi(t) \in W^{1,p'}(\Omega)$ which is, for $p > 3$ and $1/p + 1/p' = 1$, given as the solution of the following PDE

$$(A_p\Phi(t))(v) = \rho(t)(v) = \int_{\bar{\Omega}} v \,\mathrm{d}\rho_i(t) \quad \forall v \in W^{1,p}(\Omega) \tag{2.1}$$

with

$$A_p : W^{1,p'}(\Omega) \mapsto W^{1,p}(\Omega)^*$$
$$(A_p\Phi(t))(v) = a_p(\Phi(t), v) := \int_{\Omega} (\sigma(x)\nabla\Phi(x,t))\nabla v(x)\,\mathrm{d}x + \int_{\partial\Omega_S} \sigma_S\Phi(s)v(s)\,\mathrm{d}s. \tag{2.2}$$

Due to the continuous and dense embedding $W^{1,p}(\Omega) \hookrightarrow C(\bar{\Omega})$ we can use the corresponding adjoint embedding $C(\bar{\Omega}) \hookrightarrow W^{1,p}(\Omega)*$ to regard $\rho(t)$ as an element of $W^{1,p}(\Omega)^*$ such that (2.1) is well defined and has a unique solution, see [25, 21].
Using this approach, we could compute the potential $\Phi(t)$ in the whole domain, for example by finite elements for each $t \in [0,T]$. But since we are only interested in the potential at a certain number of electrodes placed on the skin, namely the values

$$y_k(t) = B_k(\Phi(t)) = \int_{D_k} \Phi(s,t)\,\mathrm{d}s, \quad k = 1 \dots n, t \in [0,T] \tag{2.3}$$

such an approach would be computationally wasteful. Therefore, we follow an adjoint approach (cf. [24]) that reduces the computational effort drastically, which we now briefly recall. First, we define the so-called impulse response functions $\omega_k \in W^{1,p}(\Omega)$ which are the solution of the adjoint problem:

$$(A_p^*\omega_k)(\phi) = B_k(\varphi) \quad \forall \phi \in W^{1,p'}(\Omega), \quad k = 1 \dots n, \tag{2.4}$$

then, we compute

$$y_k(t) = B_k(\Phi(t)) = (A_p^*\omega_k)(\Phi(t)) = (A_p\Phi(t))(\omega_k) = \rho(t)(\omega_k).$$

Hence, after pre-computation of the impulse response functions ω_k and using the definition of the moving charges ρ_i, cf. [24, Eq. 16], we can simulate for each curve u and each time instant t a surface EMG measurement at an electrode by evaluating the following line integral:

$$y_k(u,t) = \int_{\bar{\Omega}} \omega_k \, \mathrm{d}\rho_i(t) = \int_{-1}^{1} \omega_k(u(\tau))\nu\rho_l(\tau,t) \, \mathrm{d}\tau + \tilde{\omega}_k(u,t). \tag{2.5}$$

Here ρ_l is a line measure that describes the propagating part of the action potential i_m and

$$\tilde{\omega}_k(u,t) := I_m(\nu(t_0 - t)) - (\omega_k(u(-1)) + \omega_k(u(1))) \, I_m(\nu(t_1 - t))$$

represents stationary sources ensuring that the conservation of charge is fulfilled, see [24, Sec. 2.4] with I_m being the antiderivative of i_m. For a detailed motivation and definition of i_m we refer to [24] and the literature, cited there.

2.2. Identification via Optimization. Our task is to find a curve $u \in H^2(-1,1,\Omega)$, such that the simulated measurements $y_k(u,t)$ and the true measurements $y_{m,k}(t)$ fit well, i.e that the difference

$$z_k(u,t) := y_k(u,t) - y_{m,k}(t)$$

becomes small for all t. We can formulate this as a least-squares type tracking problem of the form

$$J_1(u) = \frac{1}{2}\int_0^T \|z(u,t)\|_n^2 \, \mathrm{d}t = \frac{1}{2}\int_0^T \|y(u,t) - y_m(t)\|_n^2 \, \mathrm{d}t, \tag{2.6}$$

where we collect the measurements and simulations in the vectors $y(t)$ and $y_m(t) \in \mathbb{R}^n$. Here, $\|\cdot\|_n$ is the standard Euclidean norm on $\mathbb{R}^n$ and consequently $\langle\cdot,\cdot\rangle_n$ is the standard Euclidean scalar product on $\mathbb{R}^n$.
We can make an a priori guess u_{ref} for the trajectory of the motor unit by inspecting the measurement. Furthermore, we enforce H^2-regularity of our solution curve. This guarantees that our identification problem is well posed (c.f. [24]) and reflects the physiological observation that motor units are smooth in healthy tissue. These two aspects justify the following regularization term:

$$\tilde{J}_2(u) := \int_{-1}^{1} \frac{\alpha_1}{2}\|u(\tau) - u_{ref}(\tau)\|_2^2 + \frac{\alpha_2}{2}\|\ddot{u}(\tau)\|_2^2 \, \mathrm{d}\tau,$$

where $\|\cdot\|_2$ is the standard Euclidean norm on $\mathbb{R}^3$ (consequently $\langle\cdot,\cdot\rangle_2$ is the standard Euclidean scalar product on $\mathbb{R}^3$). Next, we define the constraint function

$$\begin{aligned} G : H^2(-1,1,\mathbb{R}^3) &\mapsto H^1(-1,1,\mathbb{R}) \\ [G(u)](\tau) &:= \|\dot{u}(\tau)\|_2^2 - \nu^2, \end{aligned}$$

and demand that $[G(u)](\tau) = 0$ for almost every $\tau \in [-1,1]$. This shall ensure that the signal passes the motor unit with constant speed $\nu > 0$, as assumed in [24]. We also request that the solution is located in the muscle tissue $\overline{\Omega}_M$, as condition, which is, however, satisfied automatically at the optimal solution in practical problems. These two constraints lead to the following admissible set

$$U_{ad} := \{v \in H^2(-1,1,\mathbb{R}^3) \,|\, v(\tau) \in \overline{\Omega}_M,\, [G(v)](\tau) = 0,\ \text{for a. e. } \tau \in [-1,1]\},$$

which is well-defined, since $H^2(-1,1,\mathbb{R}^3) \hookrightarrow C^1(-1,1,\mathbb{R}^3)$.
We can then formulate the optimization problem as follows

$$\min_{u \in U_{ad}} J(u) := J_1(u) + \tilde{J}_2(u) \tag{2.7}$$

As shown in [24], problem (2.7) has at least one solution, and we get the following first-order optimality conditions:

$$\begin{aligned} 0 =& \int_0^T \langle z(u,t), z'(u,t)\delta u\rangle_n \,\mathrm{d}t + \langle \lambda, \langle \dot{u}, \delta\dot{u}\rangle_2)\rangle_{1,2} \\ &+ \int_{-1}^1 \alpha_1 \langle u(\tau) - u_{ref}(\tau), \delta u(\tau)\rangle_2 + \alpha_2 \langle \ddot{u}(\tau), \delta\ddot{u}(\tau)\rangle_2 d\tau \quad \forall \delta u \in H^2(-1,1,\mathbb{R}^3) \\ 0 =& \|\dot{u}(\tau)\|_2^2 - \nu^2 \quad \text{for a.e. } \tau \in [-1,1], \qquad \lambda \in H^1(-1,1,\mathbb{R}) \end{aligned}$$

with

$$z'(u,t)v = y'(u,t)v = \int_{-1}^1 \hat{\rho}_l(\tau,t)\nu\omega'(u(\tau))(v(\tau))\,\mathrm{d}\tau + \tilde{w}'(u,t)v.$$

In contrast to [24], we replace $\tilde{\lambda}(\langle \dot{u}, \delta\dot{u}\rangle_2)$ with $\langle \lambda, \langle \dot{u}, \delta\dot{u}\rangle_2\rangle_{1,2}$. This is possible since we know from the Riesz-Fréchet representation theorem, see [2, Theorem 5.5], that for $\tilde{\lambda} \in H^1(-1,1,\mathbb{R})^*$ there exists a $\lambda \in H^1(-1,1,\mathbb{R})$ such that

$$\tilde{\lambda}(v) = \langle \lambda, v\rangle_{1,2} \quad \forall v \in H^1(-1,1,\mathbb{R}).$$

We denote with $\langle\cdot,\cdot\rangle_{k,p}$ the standard scalar product on $W^{k,p}$ and consequently $\|\cdot\|_{k,p}$ is the standard norm on $W^{k,p}$. We also define the following dual norms for linear and bilinear forms on $H^2(-1,1,\Omega)$:

$$\|\ell\|_{2,2^*} := \sup_{\|v\|_{2,2}=1} |\ell(v)|, \qquad \|b\|_{2,2^*} := \sup_{\|v\|_{2,2}=1, \|w\|_{2,2}=1} |b(v,w)|.$$

For brevity of notation, we drop the temporal variables t and τ if they are unnecessary. The following section discusses how we can solve the optimal control problem numerically.

3. Algorithmic Framework. The optimization problem (2.7) is posed on an infinite-dimensional function space, and we will derive a numerical algorithm for its solution. We rely on well-known algorithmic concepts, but there are a couple of non-standard features of (2.7) requiring special treatment.
We will treat the constraint $G(u) = 0$ by an augmented Lagrangian approach, and the resulting unconstrained subproblems are solved by a Newton line-search method. Computing the difference between the simulated and actual measurements $z(u,t)$ requires for each t the evaluation of the integral (2.5), which cannot be performed exactly. Instead, adaptive numerical quadrature is used, resulting in an inexact evaluation of $z(u,t)$ and its derivatives and, thus, also of $J(u)$ and its derivatives via (2.6). Therefore, we will derive criteria for inexact function evaluations, which keep the computational effort low, but still guarantee global convergence.

3.1. Augmented Lagrangian Method. We use an augmented Lagrangian method to transform the constrained problem (2.7) into an unconstrained problem. We proceed in a standard way with the exception that the employed norms reflect the functional analytic structure of the problem. Therefore, we add a penalty term to $\tilde{J}_2$ and define

$$J_2(u,\xi) := \tilde{J}_2(u) + \langle \lambda + \frac{\mu}{2} G(u), G(u) \rangle_{1,2},$$

where $\xi := (\lambda, \mu) \in H^1(-1,1,\mathbb{R}) \times \mathbb{R}^+$ combines the Lagrangian multiplier λ and the penalty parameter μ. We can then define for any given ξ the unconstrained problem

$$\min_{u \in H^2(-1,1,\mathbb{R}^3)} \mathcal{L}(u,\xi) := J_1(u) + J_2(u,\xi). \tag{3.1}$$

The function $\mathcal{L}$ is called the augmented Lagrangian function of problem (2.7). It is known that for increasing μ and with a suitable update strategy for λ the solution of (3.1) converges to the solution of the constrained problem (2.7), see [1, Proposition 4.2.2]. We can achieve this with the following algorithm:

Algorithm 1: Augmented Lagrangian Algorithm

Input: $\tilde{u}_0$, $\mu_0 = 1$, $0 < c_1 < 1$, $1 < c_2$, and $\lambda_0 \equiv 0$;
do
  $\tilde{u}_{j+1} \leftarrow$ Solve (3.1) with an SQP-Newton method;
  if $\|G(\tilde{u}_{j+1})\|_{1,2} < c_1 \|G(\tilde{u}_j)\|_{1,2}$ **then**
    update λ_{j+1} and set $\mu_{j+1} = \mu_j$;
  else
    set $\mu_{j+1} = c_2 \mu_j$ and $\lambda_{j+1} = \lambda_j$;
  end
while $\|\mathcal{L}'(\tilde{u}_{j+1}, \xi_j)\|_{2,2^*} > \varepsilon_1$ *and* $\|G(\tilde{u}_{j+1})\|_{1,2} > \varepsilon_2$;
Output: $\tilde{u}_j$ and λ_j

Concerning the outer loop of the augmented Lagrangian method, it remains to define an update strategy for the Lagrangian multiplier λ_j. To define an update that generalizes the well-known least squares Lagrange multiplier update to our functional analytic setting, we take a look at the following quadratic optimization problem

$$\min_{w \in H^1(-1,1,\mathbb{R}^3)} \frac{1}{2} \langle w, w \rangle_{1,2} + J'(\tilde{u}_j) w \quad \text{s.t. } G'(\tilde{u}_j) w = 0,$$

which yields, with $\lambda \in H^1(-1,1,\mathbb{R}^3)$, the following first-order optimality conditions

$$\begin{aligned} 0 &= \langle w, v \rangle_{1,2} + J'(\tilde{u}_j) v + \langle \lambda, G'(\tilde{u}_j) v \rangle_{1,2} \quad \forall v \in H^1(-1,1,\mathbb{R}^3) \\ 0 &= G'(\tilde{u}_j) w. \end{aligned} \tag{3.2}$$

Testing (3.2) with $v \in \ker G'(\tilde{u}_j)^\perp := \{v \in H^1(-1,1,\mathbb{R}^3) : \langle v, w\rangle_{1,2} = 0 \quad \forall w \in \ker G'(\tilde{u}_j)\}$ transforms the first equation of (3.2) into

$$0 = J'(\tilde{u}_j)v + \langle \lambda, G'(\tilde{u}_j)v\rangle_{1,2} \quad \forall v \in \ker G'(\tilde{u}_j)^\perp.$$

We observe that this Lagrangian multiplier removes slopes that are H^1-orthogonal to $\ker G'(\tilde{u}_j)^\perp$, and at an optimal solution $\tilde{u}_j = u^\star$, we see that $\bar{\lambda}$, computed via (3.2) is equal to the original Lagrangian multiplier $\lambda^\star$. Defining $\nabla_{H_1} J(\tilde{u}_j)$ by $\langle \nabla_{H_1} J(\tilde{u}_j), v\rangle_{1,2} = J'(\tilde{u}_j)v$ for all v we observe that $\bar{\lambda}$ is indeed the solution of the minimization problem

$$\min_{\lambda \in H^1} \|\nabla_{H_1} J(\tilde{u}_j) - G'(\tilde{u}_j)^* \lambda\|_{1,2}^2,$$

where $G'(\tilde{u}_j)^*$ is the Hilbert space adjoint of $G'(\tilde{u}_j)$ in H^1.

3.2. Inexact Function Evaluation by Adaptive Quadrature. While the outer loop of the augmented Lagrangian method is fairly standard, our particular problem structure requires some algorithmic adjustments for the inner SQP-Newton method solving (3.1).
In this algorithm, we must compute the Lagrangian function $\mathcal{L}$ and its derivatives. To do this numerically, we replace the integrals in J_1 and J_2 with quadrature rules. Thereby, we note that after FE discretization, the integrands in J_2 are piecewise polynomials of some fixed order. Thus, we can evaluate these integrals exactly with a piecewise quadrature rule of sufficient order.
It remains to compute the functional J_1, given by (2.6). We observe that the integrand of (2.6) requires the evaluation of $z(u,t)$ which, in turn, can only be computed by an evaluation of an integral, namely (2.5). This nested integral structure is the most challenging and time-consuming part of the computation.
For the outer integral in (2.6), we can use any piecewise quadrature rule as long as the segmentation of the time interval is fine enough. The inner integral in (2.5) is more difficult to compute. As shown in [24], the action potential is a propagating signal, with only small support on the motor unit u. Furthermore, the action potential shows large oscillations in this area. These oscillations make a standard piecewise quadrature rule on uniform intervals inefficient. Thus, an adaptive quadrature rule is appropriate to compute $z(u,t)$ up to a certain accuracy. Due to this adaptive procedure, which changes from step to step, the computed result cannot be interpreted as an exact evaluation of a discrete quantity, but as an inexact evaluation of a continuous quantity. Thus, a deliberate adjustment of accuracy requirements is necessary to avoid interference of the introduced errors with the global convergence of our algorithm. Inexact function evaluations are not uncommon in nonlinear optimization, and the topic is studied in different settings, cf., e.g., [13, 14, 3, 6, 7, 28]. Our approach resembles [3], where a Trust-Region algorithm is shown to converge globally under a condition, similar to (3.4), below. Using a sufficient error bound, cf. Lemma 3.1 below, we can ensure that our algorithm satisfies this condition, which we then use to modify a classical global convergence result for backtracking line search algorithms, cf., e.g., [22, Theorem 3.2].
To indicate that J_1 and thus also $\mathcal{L}$ are computed via a quadrature rule, we use an index Q and write $J_{1,Q}$ respective $\mathcal{L}_Q$. We don't use an index for J_2 since we can compute it exactly. By $\mathcal{L}_Q$ we denote the version of the Lagrangian functional with $J_{1,Q}$ evaluated inexactly and all other addends evaluated exactly. The following Lemma connects the accuracy of the computation of z with the relative error in the computation of differences of $\mathcal{L}$:

LEMMA 3.1. *Let $0 < \varsigma < 1$ and assume that the following error bound for $z(\cdot,t)$ holds pointwise*

for every t and two trajectories u, v:

$$\begin{aligned}&|\langle z(v,t) - z_Q(v,t), z(v,t) + z_Q(v,t)\rangle_n - \langle z(u,t) - z_Q(u,t), z(u,t) + z_Q(u,t)\rangle_n| \\ \leq &\frac{2\varsigma}{T}|\mathcal{L}_Q(v,\xi) - \mathcal{L}_Q(u,\xi)|.\end{aligned} \tag{3.3}$$

Then the Lagrangian function can be evaluated with the following accuracy estimate:

$$|\left(\mathcal{L}(v,\xi) - \mathcal{L}(u,\xi)\right) - \left(\mathcal{L}_Q(v,\xi) - \mathcal{L}_Q(u,\xi)\right)| \leq \varsigma|\mathcal{L}_Q(v,\xi) - \mathcal{L}_Q(u,\xi)|. \tag{3.4}$$

Proof. As mentioned above, we can compute J_2 exactly by choosing a quadrature rule of a sufficiently high order. Therefore:

$$\begin{aligned}&|\left(\mathcal{L}(v,\xi) - \mathcal{L}(u,\xi)\right) - \left(\mathcal{L}_Q(v,\xi) - \mathcal{L}_Q(u,\xi)\right)| = |\left(J_1(v) - J_1(u)\right) - \left(J_{1,Q}(v) - J_{1,Q}(u)\right)| \\ \leq &\frac{1}{2}\int_0^T \left|\left(\|z(v,t)\|_n^2 - \|z_Q(v,t)\|_n^2\right) - \left(\|z(u,t)\|_n^2 - \|z_Q(u,t)\|_n^2\right)\right| \,\mathrm{d}t \\ = &\frac{1}{2}\int_0^T |\langle z(v,t) - z_Q(v,t), z(v,t) + z_Q(v,t)\rangle_n - \langle z(u,t) - z_Q(u,t), z(u,t) + z_Q(u,t)\rangle_n| \,\mathrm{d}t \\ \overset{(3.3)}{\leq} &\varsigma|\mathcal{L}_Q(v,\xi) - \mathcal{L}_Q(u,\xi)|.\end{aligned}$$

∎

3.3. Newton Line-Search with Inexact Function Evaluations. Using a Newton method, we attempt to find a stationary point of (3.1). We will employ a Hessian modification strategy to ensure that only directions of sufficient descent are computed. Further, we consider that we evaluate all required quantities inexactly by adaptive quadrature.
To compute an update step δu_k we minimize the following quadratic model of $\mathcal{L}_Q(u_k, \xi)$:

$$\min_{\delta u_k \in H^2(-1,1,\Omega)} \mathcal{L}_Q'(u_k,\xi)\delta u_k + \frac{1}{2}H_Q(u_k;\Lambda)(\delta u_k, \delta u_k)$$

where H_Q is a modified hessian with respect to some $\Lambda \geq 0$:

$$H_Q(u_k;\Lambda)(v,v) := \mathcal{L}_Q''(u_k,\xi)(v,v) + \Lambda\langle v, v\rangle_{2,2}.$$

This is equivalent to solving the linear equation:

$$0 = \mathcal{L}_Q'(u_k,\xi)v + H_Q(u_k,\Lambda)(\delta u_k, v) \quad \forall v \in H^2(-1,1,\Omega). \tag{3.5}$$

We will choose $\Lambda \geq 0$ such that for some $\gamma > 0$:

$$\gamma\|v\|_{2,2}^2 \leq H_Q(v;\Lambda)(v,v), \tag{3.6}$$

implying

$$\mathcal{L}_Q'(u_k,\xi)(\delta u_k) = -H_Q(u_k,\Lambda)(\delta u_k, \delta u_k) \leq -\gamma\|\delta u_k\|_{2,2}^2. \tag{3.7}$$

We choose Λ by a standard strategy: starting with $\Lambda = 0$ we compute δu_k and test if (3.7) holds. If not, we use some increasing sequence of $\Lambda > 0$ until (3.7) is satisfied. Since $\|\mathcal{L}_Q''(u_k,\xi)\|_{2,2^*}$ is bounded, this loop will terminate at the latest if $\Lambda \geq \gamma + \|\mathcal{L}_Q''(u_k,\xi)\|_{2,2^*}$ and we obtain

$$\|H_Q(u_k,\Lambda)\|_{2,2^*} \leq \Lambda + \|\mathcal{L}_Q''(u_k,\xi)\|_{2,2^*}.$$

As usual, the iterate u_k is then updated via $u_{k+1} := u_k + \beta_k u_k$, where the damping factor β_k is computed by a standard back-tracking line search, which terminates if an Armijo condition of the form

$$\mathcal{L}_{Q_\beta}(u_k + \beta\delta u_k, \xi) \leq \mathcal{L}_{Q_\beta}(u_k, \xi) + \eta \mathcal{L}'_{Q_\beta}(u_k, \xi)(\beta\delta u_k) \quad \beta \in]0, 1] \tag{3.8}$$

is fulfilled. Note that (3.8) is evaluated by a possibly different quadrature rule Q_β than the quadrature rule Q, used to compute δu_k. Nevertheless, under certain conditions, we can show that the back-tracking line search terminates with some β that is above a certain lower bound:

LEMMA 3.2. *Let $\eta \in]0, 1[$, $\sigma > 0$ and assume that δu_k fulfills (3.7) and*

$$\mathcal{L}'_{Q_\beta}(u_k, \xi)\delta u_k \leq \sigma \mathcal{L}'_Q(u_k, \xi)\delta u_k \tag{3.9}$$

for all quadrature rules employed during the line-search back-tracking. Additionally, we assume that $\mathcal{L}'_{Q_\beta}(u_k, \xi)$ satisfies the following Lipschitz condition:

$$\left| \left[\mathcal{L}'_{Q_\beta}(u_k, \xi) - \mathcal{L}'_{Q_\beta}(u_k + \beta\delta u_k, \xi) \right] (\delta u_k) \right| \leq \beta L \|\delta u_k\|_{2,2}^2.$$

Then there exists a $\bar{\beta}(\eta, L, \gamma, \sigma)$ such that for all $\beta < \bar{\beta}$ the Armijo condition (3.8) is fulfilled.

Proof. Using the Lipschitz condition for $\mathcal{L}'_{Q_\beta}(\cdot, \xi)$, we can compute

$$\begin{aligned}
&\mathcal{L}_{Q_\beta}(u_k + \beta\delta u_k, \xi) - \mathcal{L}_{Q_\beta}(u_k, \xi) - \mathcal{L}'_{Q_\beta}(u_k)(\beta\delta u_k) \\
&= \int_0^1 \left[\mathcal{L}'_{Q_\beta}(u_k + t\beta\delta u_k, \xi) - \mathcal{L}'_{Q_\beta}(u_k, \xi) \right] (\beta\delta u_k) \, \mathrm{d}t \leq \int_0^1 tL\beta^2 \|\delta u_k\|_{2,2}^2 \, \mathrm{d}t \leq \frac{L\beta^2}{2} \|\delta u_k\|_{2,2}^2.
\end{aligned}$$

Together with (3.7) and (3.9) this implies

$$\begin{aligned}
&\mathcal{L}_{Q_\beta}(u_k + \beta\delta u_k, \xi) - \mathcal{L}_{Q_\beta}(u_k, \xi) \leq \mathcal{L}'_{Q_\beta}(u_k)(\beta\delta u_k, \xi) + \frac{L\beta^2}{2} \|\delta u_k\|_{2,2}^2 \\
\leq &\eta \mathcal{L}'_{Q_\beta}(u_k)(\beta\delta u_k, \xi) + (1 - \eta)\sigma \mathcal{L}'_Q(u_k)(\beta\delta u_k, \xi) + \frac{L\beta^2}{2} \|\delta u_k\|_{2,2}^2 \\
\leq &\eta\beta \mathcal{L}'_{Q_\beta}(u_k)(\delta u_k, \xi) + \beta \left(\frac{L\beta}{2} - (1 - \eta)\sigma\gamma \right) \|\delta u_k\|_{2,2}^2.
\end{aligned}$$

The last term is negative if we choose $0 < \beta < \frac{2(1-\eta)\sigma\gamma}{L}$, and thus the result follows. ∎

Thus, a line-search algorithm (which we will elaborate on in Alg. 5, below) can terminate with two different outcomes: either (3.8) holds eventually, or (3.9) is violated at some point. In the latter case, the step δu_k should be rejected and recomputed with tighter tolerance, because the predicted decrease $\mathcal{L}'_Q(u_k, \xi)\delta u_k$ may have been too optimistic for this direction.

We can now connect these parts to the following conceptual algorithm:

Algorithm 2: SQP-Newton

Input: $u_0 = \tilde{u}_j$, $tol_0 = 10^{-5}$, $\varepsilon_1 = 10^{-6}$;
do
  $\Lambda = 0$;
  do
    $\delta u_k \leftarrow$ solve (3.5) with Alg. 4 (see below) and tolerance tol_k;
    increase Λ ; `// only if` (3.6) `is violated`
  while (3.6) *is violated*;
  $\beta_k \leftarrow$ apply line search Alg. 5 (see below), until (3.8) holds or (3.9) is violated;
  if (3.9) *is violated* **then**
    tighten tol_k and set $u_{k+1} = u_k$;
  else
    $u_{k+1} = u_k + \beta_k \delta u_k$;
  end
while $\|\mathcal{L}'(u_{k+1}, \xi)\|_{2,2^*} \geq 0.9\varepsilon_1$;
Output: $\tilde{u}_{j+1} = u_k$;

To formulate a global convergence result that takes into account inexactness by adaptive quadrature, let us introduce δv_k as the solution of the following equation using the exact derivative:

$$0 = \mathcal{L}'(u_k, \xi)v + H_Q(u_k, \Lambda)(\delta v_k, v) \quad \forall v \in H^2(-1, 1, \Omega), \tag{3.10}$$

satisfying the inequality

$$\|\mathcal{L}'(u_k, \xi)\|_{2,2^*} \leq \|H_Q(u_k, \Lambda)\|_{2,2^*} \|\delta v_k\|_{2,2}. \tag{3.11}$$

Additionally we assume that δu_k satisfies

$$\|\delta v_k\|_{2,2} \leq \Theta \|\delta u_k\|_{2,2}. \tag{3.12}$$

This can be fulfilled uniformly if the quadrature rule is kept sufficiently accurate. The main result of this section is then

THEOREM 3.3. *Let $0 < \gamma \leq \Gamma$, $\varsigma, \eta \in]0, 1[$, $\sigma, \Theta > 0$ be fixed constants. Assume that numerical quadrature rules are used, such that the inequalities (3.4), (3.7), (3.9), and (3.12) are satisfied in each step, and such that $\|H_Q(u_k, \Lambda)\|_{2,2^*} \leq \Gamma$. Then algorithm 2 converges globally, i.e.,*

$$\lim_{k \to \infty} \|\mathcal{L}'(u_k, \xi)\|_{2,2^*} = 0.$$

Proof. Lemma 3.2 implies that there exists $\bar{\beta} > 0$ such that $\bar{\beta} < \beta_k \ \forall k$. Thus, we can compute:

$$\begin{aligned}
&\mathcal{L}(u_{k+1}, \xi) - \mathcal{L}(u_k, \xi) \\
&\quad = \mathcal{L}(u_{k+1}, \xi) - \mathcal{L}(u_k, \xi) - \big(\mathcal{L}_{Q_{\beta_k}}(u_{k+1}, \xi) - \mathcal{L}_{Q_{\beta_k}}(u_k, \xi)\big) + \big(\mathcal{L}_{Q_{\beta_k}}(u_{k+1}, \xi) - \mathcal{L}_{Q_{\beta_k}}(u_k, \xi)\big) \\
&\quad \overset{(3.4)}{\leq} (1 - \varsigma)\big(\mathcal{L}_{Q_{\beta_k}}(u_{k+1}, \xi) - \mathcal{L}_{Q_{\beta_k}}(u_k, \xi)\big) \overset{(3.8)}{\leq} (1 - \varsigma)\eta\beta_k \mathcal{L}'_{Q_{\beta_k}}(u_k, \xi)\delta u_k \\
&\quad \overset{(3.9)}{\leq} (1 - \varsigma)\eta\sigma\beta_k \mathcal{L}'_Q(u_k, \xi)\delta u_k \overset{(3.7)}{\leq} -(1 - \varsigma)\eta\sigma\bar{\beta}\gamma\|\delta u_k\|_{2,2}^2 \leq 0.
\end{aligned}$$

Since G is continuous, see [24, Lemma 3.1], and $u_k(\tau) \in U_{ad}$ for all k and all τ, it follows that $\|G(u_k)\|_{1,2}$ is bounded and thus $\mathcal{L}(u_k, \xi)$ is bounded from below, more precisely

$$\mathcal{L}(u_k,\xi) = \underbrace{J(u_k)}_{\geq 0} + \underbrace{\langle \lambda, G(u_k)\rangle_{1,2}}_{\geq -\|\lambda\|_{1,2}\|G(u_k)\|_{1,2}} + \underbrace{\frac{\mu}{2}\langle G(u_k), G(u_k)\rangle_{1,2}}_{\geq 0} \geq -\|\lambda\|_{1,2}\|G(u_k)\|_{1,2} =: \underline{\mathcal{L}}.$$

Therefore, $\sum_{k=0}^{\infty} \mathcal{L}(u_{k+1},\xi) - \mathcal{L}(u_k,\xi) \geq \underline{\mathcal{L}} - \mathcal{L}(u_0,\xi)$ implying $|\mathcal{L}(u_{k+1},\xi) - \mathcal{L}(u_k,\xi)| \to 0$. Thus, we can conclude that $\|\delta u_k\|_{2,2^*} \to 0$. Finally (3.11) and (3.12) imply:

$$0 \leq \|\mathcal{L}'(u_k,\xi)\|_{2,2^*} \leq \Gamma\|\delta v_k\|_{2,2} \leq \Gamma\Theta\|\delta u_k\|_{2,2} \to 0. \qquad \square$$

Local Convergence of the SQP-Newton method. It is well known that the local convergence rate of an SQP method depends on accurate search directions, e.g., the better the preconditioner approximates the Hessian of the objective function, the faster the SQP method converges locally. For our problem, the search direction additionally depends on the accuracy of the adaptive quadrature. To achieve fast linear convergence we require for some $\epsilon > 0$ that

$$\|\delta u_k - \delta v_k\|_H \leq \epsilon \|\delta v_k\|_H \tag{3.13}$$

is satisfied for all k. Here, $\|\cdot\|_H$ is the following energy norm:

$$\|v\|_H := H(u_k,\Lambda)(v,v) \quad \forall v \in H^2(-t_A, t_A, \Omega_M).$$

Unfortunately, we can not include this requirement directly in the adaptive quadrature algorithm. But the following result gives us an applicable condition.

LEMMA 3.4. *Assume that the error bound*

$$\left|\langle z(u_k,t), z'(u_k,t)(\delta u_k - \delta v_k)\rangle_n - \langle z_Q(u_k,t), z_Q'(u_k,t)(\delta u_k - \delta v_k)\rangle_n\right| \leq \frac{\epsilon^2}{T}\|\delta v_k\|_H^2 \tag{3.14}$$

is satisfied for all $t \in [0,T]$. Then $\|\delta u_k - \delta v_k\|_H \leq \epsilon\|\delta v_k\|_H$.

Proof. With $\delta e_k := \delta u_k - \delta v_k$, (3.5), and (3.10) we compute

$$\begin{aligned}\|\delta e_k\|_H^2 = H(u_k;\Lambda)(\delta e_k, \delta e_k) &= \left|\left[J_1'(u_k) - J_{1,Q}'(u_k)\right](\delta e_k)\right| \\ \leq \int_0^T \left|\langle z(u_k,t), z'(u_k,t)(\delta e_k)\rangle_n - \langle z_Q(u_k,t), z_Q'(u_k,t)(\delta e_k)\rangle_n\right| \,\mathrm{d}t &\leq \epsilon^2\|\delta v_k\|_H^2.\end{aligned}$$

Taking the square root on both sides provides the result. $\square$

By our choice of Λ we may assume that the energy norm $\|\cdot\|_H$ and the H^2-norm are equivalent, i.e. there exist $0 < \gamma \leq \Gamma$ such that $\gamma\|v\|_{2,2} \leq \|v\|_H \leq \Gamma\|v\|_{2,2}$. Then (3.13) implies (3.12), so that it is sufficient to test for (3.13). The details of our step computation algorithm will be elaborated in Alg 4 below.

4. A Practical Optimization Algorithm. Up to now, we discussed our algorithmic approach in a somewhat idealized setting, leaving away the details of implementation.
In this section, we discuss these details and turn the above-described algorithmic concept into a practical algorithm. We thereby split the discussion into three parts. First, we discuss the discretization and the evaluation of the required quantities, second adaptive quadrature, and finally, we discuss the implementation of our inexact SQP method with line search.

4.1. Discretization of the problem. To solve the problem numerically, we must discretize and evaluate the required quantities. This includes the discretization of the motor unit u and the computation of the impulse response functions ω_k.

Discretization of Signal Trajectories. To discretize $u \in H^2(-1,1,\Omega_M)$ in a conformal way, we choose a segmentation $\mathcal{T}_{FE}$ of the interval $[-1,1]$ and employ cubic Hermite Finite Elements on $\mathcal{T}_{FE}$ see [12, Section 8.6.2]. We thus obtain the following ansatz space:

$$V_h := \{u \in C^1(-1,1,\Omega) : u|_I \in P_3(I)\ \forall I \in \mathcal{T}_{FE}\} \subset H^2(-1,1,\Omega_M).$$

Discretization of Impulse Response Functions. Computing $J_{1,Q}$ via (2.5) requires evaluating the impulse response functions ω_k and their derivatives along the trajectory. These evaluations are well defined in the continuous setting since $\omega_k \in C^\infty(\Omega_M) \cap W^{1,p}(\Omega)$, see [24, Lemma 2.2]. The impulse response functions do not change during the optimization, so we can compute them a-priory, using finite elements on Ω. To this end, we use standard Lagrange elements on a triangulation $\mathcal{K}$ of Ω. On $\mathcal{K}$, we use continuous piecewise polynomial ansatz functions to discretize $W^{1,p}(\Omega)$ and $W^{1,p'}(\Omega)$ by

$$W_h := \{w_h \in C(\Omega,\mathbb{R}) : w_h|_K \in P_m(K)\ \forall K \in \mathcal{K}\}.$$

For the computation of our hession approximation $H_Q(u,\Lambda)$, we need the second spatial derivatives of the $\omega_{k,h}$, which cannot be represented by linear finite elements. Therefore, we must use at least polynomials of order $m = 2$ or, more accurately, $m = 3$ as ansatz functions.
We compute approximations $\omega_{k,h}$ for each ω_k with a Galerkin method, applied to the adjoint problem (2.4). This leads to the discrete problem

$$\text{find } \omega_{k,h} \in W_h \text{ s.t.} \quad (A_p^*\omega_{k,h})(\phi_h) = B(\phi_h) \quad \forall \phi_h \in W_h$$

with A_p and B defined in (2.2) and (2.3). After finite element discretization, we end with a sparse linear system of equations, which we solve with a standard preconditioned conjugate gradient method. Since grid hierarchies are usually not available in practical problems, we use a standard incomplete Cholesky decomposition, see [17], as a preconditioner.

4.2. Adaptive Quadrature Algorithm. The crucial and most involved step of computing $\mathcal{L}(u_k,\xi)$ at an iterate u_k is the computation of the difference between simulated and actual measurement $z(u_k,t)$. Here we will use adaptive quadrature of the integral (2.5), based on Gauß-Kronrod quadrature rules [15]. In section 3.3, we have defined the two error bounds (3.3) and (3.14) to ensure that the SQP-Newton converges globally and locally with a reasonable rate. These two bounds need to be met by computation of $z(u_k,t)$ via adaptive quadrature, which has to be performed for each electrode and for each time instance where the simulated signal has to be evaluated. Thus, most of the computing time is spent during the quadrature algorithm. Our algorithm 3 is based on standard ideas, and similar versions can be found, e.g., in [4, 23].
The two error bounds (3.3) and (3.14) require slightly different versions of our adaptive quadrature algorithm. However, their general structure stays the same. Thus, we will first explain the general structure for some global error bound $\mathbf{E}$ that has to satisfy a given global tolerance *tol* and then discuss the specific details when we talk about the computation of the step and the line search.
Adaptive quadrature uses a partition of the domain of integration, which is gradually refined with the help of error indicators. Consider a given partition $\mathcal{T}$ of the integration interval $[-1,1]$, which consists of $|\mathcal{T}|$ sub-intervals. We start with computing the contribution to z on each sub-interval $I \in \mathcal{T}$, once with the Gauß rule (z_{G_I}) and once with the corresponding Kronrod rule (z_{K_I}). Using these quantities, we then compute $z_K(u,t) = \sum_{I\in\mathcal{T}} z_{K_I}(u,t)$, $z_G(u,t) = \sum_{I\in\mathcal{T}} z_{G_I}(u,t)$,

Algorithm 3: Adaptive Quadrature

```
Input: u_k, T_0, tol ;
Variable Input: u_{k+1} or δe_k ;                    // depends on the choice of E
Set T_R = T_0 ;
while |T_j| < maxIntervals do
    compute z_{K_I}, z_{G_I}, and E_I for all I ∈ T_R ;
    compute z_K, z_G and E ;          // possibilities for E, see (4.3) and (4.2)
    if E < tol then
        break ;
    else
        choose T_R ⊂ T_j such that (4.1) is satisfied ;
        T_{j+1} ← refine all intervals in T_R ;
    end
end
Output: z_K, z_G, and T_j ;
```

and a global error estimate $\mathbf{E}$ (to be specified below). Simultaneously, we can evaluate local error estimates $\mathbf{E}_I$, which are the portions of the global error on the interval I. If $\mathbf{E} > tol$ is violated, we choose $\mathcal{T}_R \subset \mathcal{T}$ such that $\mathcal{T}_R$ is the smallest subset with

$$\sum_{I \in \mathcal{T}_R} \mathbf{E}_I \geq \chi \sum_{I \in \mathcal{T}} \mathbf{E}_I. \tag{4.1}$$

Here, χ is a parameter that shall ensure that $\mathcal{T}_R$ is not too small, i.e., such that the adaptive quadrature does not need too many iterations. To find the smallest subset $\mathcal{T}_R$, we sort the local error estimates $\mathbf{E}_I$ by value and choose the intervals with the highest magnitude. We choose $\chi = 0.75$. However, frequently the errors are concentrated in only a few intervals. We choose a safety parameter ε_S to avoid the unnecessary refinement of too many intervals with very small contributions, i.e., an interval I is not refined if $\mathbf{E}_I < \varepsilon_S$. One possible choice is $\varepsilon_S = tol/|\mathcal{T}|$, which corresponds to the magnitude of an average error distribution.
Next, we refine all intervals in $\mathcal{T}_R$, and on this new partition, we reevaluate the integral. Since function evaluations are, in general, expensive, we store the local values $z_{G_I}(u,t)$, $z_{K_I}(u,t)$, and $\mathbf{E}_I$ and only recompute the local values on the refined intervals.

Evaluation of $\omega_{k,h}$. During our quadrature routine, we have to evaluate the FE approximations $\omega_{k,h}$ of the impulse response functions at all quadrature points. For a given quadrature point $x_i = u(\tau_i)$, the evaluation of $\omega_{k,h}(u(\tau_i))$ is

$$\omega_{k,h}(u(\tau_i)) = \sum_{\operatorname{supp}(p_l) \subset K_i} p_l(x_i) c_l.$$

Here, the c_l are the coefficients of FE function $\omega_{k,h}$, p_l are the Lagrange basis functions, and K_i is the tetrahedron for which $x_i \in K_i$. In general, it is easy to decide, if x_i is contained in a given tetrahedron, but identifying K_i for given x_i is numerically expensive without additional information, because a full search through all tetrahedra of the grid may be necessary to find K_i To find K_i efficiently, we exploit that the quadrature points are ordered along the trajectory of u. Thus, we can use a neighborhood search: if $K_{i-1} \ni x_{i-1}$ is known, and x_i is the next quadrature point, we test all neighbors of K_{i-1} if they contain x_i. If that fails, we compute the distance between the center of the neighbors and the quadrature point x_i. We then choose the neighbor with the lowest distance to the quadrature point and test its neighbors. We repeat

this procedure until we find the tetrahedron that contains x_i or a maximal number of tetrahedra tested. In the latter outcome, or if an initial inclusion $x_{i-1} \in K_{i-1}$ is not known, we fall back to a full search (or hierarchic search if possible) over the whole grid.

Computation of $\mathcal{L}(u, \xi)$. The last quantity to compute is the Lagrange functional $\mathcal{L}(u, \xi)$ and its derivatives. If we inspect the terms in $\mathcal{L}$, we notice that J_2 and the regularization term of H are only compositions of parts of the H^2 scalar product. The only exception is J_1, why we treat the computation of J_1 separately.
Since u is, after FE discretization, a polynomial, we can compute J_2 and the regularization term $\langle v, v \rangle_{1,2}$ and their derivatives exactly, using a Gauss rule of sufficiently high order on each interval of the segmentation $\mathcal{T}_{FE}$.
To evaluate J_1, we compute the outer integral over time which defines the tracking problem by the midpoint rule. The measured values $y_m(t)$ are, in general, also given at discrete equidistant time points t_i. Thus, this is naturally the best method if we choose the quadrature points at t_i. This determines the times steps t_i for which we compute $z(u, t_i)$ with the above-described adaptive quadrature algorithm, starting with $\mathcal{T}_{FE}$ as initial segmentation. We use the resulting $\mathcal{T}_{\delta v}$ to compute z'_G, z'_K, and z''_G.

4.3. Details of the inexact SQP-method. In section 3.3, we proved that the SQP-Newton method converges globally if the error bound (3.4) for the numerical quadrature is fulfilled and if in addition (3.7), (3.9), and (3.12) hold. Additionally, we require the error bound (3.13) to ensure that the SQP-Newton method converges locally at a reasonable speed. In this section, we devise algorithmic measures to enforce these error bounds by customizing the numerical quadrature.
The exact evaluation of (3.4) would require the computation of the exact values of the integrals, which are not available. Thus, we have to replace the exact quantities with estimates. A standard method to derive such estimates is to replace the integrals with quadrature rules of higher order. In the following, we will use, as already mentioned, the Gauß-Kronrod quadrature formula. These quadrature rules consist of a Gauß quadrature rule, denoted by an index G, and an extended quadrature rule of higher order (called Kronrod extension), denoted by the index K. With their help, we define the quantities:

$$
\begin{aligned}
E(u, v, t) &:= |\langle (z_K - z_G)(u, t), (z_K + z_G)(u, t) \rangle_n - \langle (z_K - z_G)(v, t), (z_K + z_G)(v, t) \rangle_n| \\
\mathcal{E}(u, v, t) &:= |\langle z_K(u, t), z'_K(u, t)v \rangle_n - \langle z_G(u, t), z'_G(u, t)v \rangle_n| \\
tol_{E,k} &:= \frac{2\varsigma}{T} |\mathcal{L}_K(u_{\text{try}}, \xi) - \mathcal{L}_K(u_k, \xi)| \\
tol_{\mathcal{E},k} &:= \frac{\epsilon^2}{T} \|\delta v_k\|_H^2
\end{aligned}
$$

where u_{try} is a trial candidate for the next iterate u_{k+1}. The error bounds (3.3) and (3.14) are then replaced by their computable counterparts:

$$E(u_k, u_{\text{try}}, t) \leq tol_{E,k} \quad \forall t \in [0, T] \tag{4.2}$$

$$\mathcal{E}(u_k, \delta e_k, t) \leq tol_{\mathcal{E},k} \quad \forall t \in [0, T]. \tag{4.3}$$

Here $\delta e_k := \delta u_k - \delta v_k$. To compute the directions δu_k and δv_k, we apply a Galerkin method to (3.5) and (3.10), which lead to the discrete problems:

$$H_G(u_k, \Lambda_k)(\delta u_k, w) + \mathcal{L}'_G(u_k, \xi_j)w = 0 \quad \forall w \in V_h. \tag{4.4}$$

$$H_G(u_k, \Lambda_k)(\delta v_k, w) + \mathcal{L}'_K(u_k, \xi_j)w = 0 \quad \forall w \in V_h. \tag{4.5}$$

Observe that (4.5) differs from (3.10), since the computationally unavailable quantity $\mathcal{L}'(u_k, \xi_j)$ is replaced by its computable approximation $\mathcal{L}'_K(u_k, \xi_j)$.
We incorporate these error bounds in two parts of our algorithm. First, when computing the direction δu_k where we use (4.3) together with the adaptive quadrature to ensure that the algorithm converges quickly, locally, i.e., that (3.13) is satisfied. And secondly, we use the error bound (4.2) to enforce global convergences, cf. Theorem 3.3, when applying the line search.

Computation of Search Directions. When computing the directions of descent δu_k and δv_k, defined by (4.4) and (4.5), respectively, we need to ensure that they satisfy condition (3.13) and thus also (3.12).
We use the following correction loop, as specified in Algorithm 4. First, we compute $\mathcal{L}'_G(u_k, \xi_j)$, $\mathcal{L}'_K(u_k, \xi_j)$ and $H_G(u_k, \Lambda)$ as described above, cf. Section 4.2. The J_1 part of $\mathcal{L}'$ is computed adaptively up to the required accuracy. Second, we solve (4.4) and (4.5), if necessary with regularization, as described below (3.7). Having computed δu_k and δv_k, we test if they satisfy (3.13). If this is not the case, we tighten the given error bound and restart the computation. Otherwise, Algorithm 4 terminates successfully and returns δu_k and δv_k. Since δv_k is more accurate than δu_k, we will use δv_k as input for the following line-search procedure.

Algorithm 4: Computation of SQP-Newton direction

Input: u_k, tol_{k-1} and $\mathcal{T}_0$; `// in general` $\mathcal{T}_0 = \mathcal{T}_{FE}$
compute J'_2 and J''_2 on $\mathcal{T}_{FE}$;
for *i=1,2,...* **do**
  if $i == 1$ **then**
    $z'_G, z'_K, z''_G, \mathcal{T}_i \leftarrow$ Alg. 3 (Input: u_k, $\mathcal{T}_0$, $tol_{E,k-1}$, (4.6)) ;
  else
    $z'_G, z'_K, z''_G, \mathcal{T}_i \leftarrow$ Alg. 3 (Input: u_k, δe_k, $\mathcal{T}_{i-1}$, $tol_{\mathcal{E},k}$, (4.3));
  end
  compute $J'_{1,G}, J'_{1,K}, J''_{1,G}, \mathcal{L}'_G$ and $\mathcal{L}'_K$;
  do
    choose Λ and compute H_G ; `// choice of` Λ`, see Sec. 3.3`
    $\delta u_k, \delta v_k \leftarrow$ solve (4.4) resp. (4.5) ;
  while (3.7) *is not satisfied*;
  if $\|\delta e_k\|_H \leq \epsilon \|\delta v_k\|_H$ **then**
    terminate "accuracy requirement (4.3) fulfilled";
  else
    update $tol_{\mathcal{E},k}$;
  end
end
Output: δv_k and $\mathcal{T}_{\delta v_k} = \mathcal{T}_k$;

Let us now explain some features of Algorithm 4 in more detail. As described in Section 4.2, we can split the computation of $\mathcal{L}'_G(u_k, \xi_j)$, $\mathcal{L}'_K(u_k, \xi_j)$ and $H_G(u_k, \Lambda)$ into two parts. Since $J'_2(u_k, \xi_j)$ and $J''_2(u_k, \xi_j)$ can be evaluated exactly, these computations can be performed prior to the loop. Second, we compute $J'_{1,G}(u_k)$, $J'_{1,K}(u_k)$, and $J''_{1,G}(u_k)$ requiring the adaptive computation of $z'_G(u_k, t)$, $z'_K(u_k, t)$, and $z''_G(u_k, t)$ as described in the previous section.
When computing $z(u_k, t)$ adaptively with Algorithm 3, we use, in general, the error criterion (4.3). This, however, is only possible for $i \geq 2$, since (4.3) requires an approximation of the difference $\delta u_k - \delta v_k$ which is only available after the first pass of the loop. Thus, in the first pass of the loop, we have to replace (4.3) by an alternative criterion. A reasonable choice would

be using (4.2) used in the line search, see below. But this error bound depends on u_{k+1} and $tol_{E,k}$ which are unavailable. Thus, we drop the quantities requiring u_{k+1} and replace $tol_{E,k}$ with $tol_{E,k-1}$ resp. 10^{-5} if $k = 0$. These modifications lead to the following error bound:

$$|\langle (z_K - z_G)(u_k, t), (z_K + z_G)(u_k, t) \rangle_n| \leq tol_{E,k-1}. \tag{4.6}$$

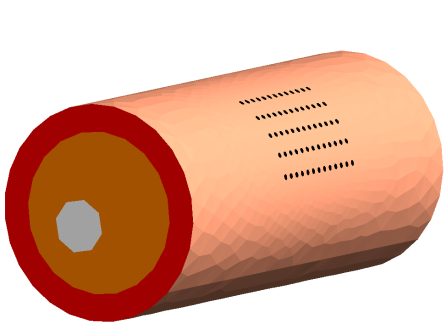
Beginning with the second pass of the loop, we can use δu_k and δv_k from the previous pass to apply the error bound (3.13) when adaptively computing $z(u_k, t)$.
When solving (4.4) and (4.5), we first choose $\Lambda = 0$ and compute $H_G(u_k, \Lambda)$. Then we solve the resulting systems of linear equations with the help of a Cholesky decomposition and test if (3.7) is satisfied. If this test fails, we increase Λ, i.e., set $\Lambda = 0.1$ and multiply it with 2 after each failed test. Next, we update $H_G(u_k, \Lambda)$ and recompute δu_k and δv_k until they fulfill (3.7).

Inexact Line Search. When computing the damping factor β, we must ensure that the Lagrangian functional satisfies (3.4). Therefore, it is necessary to evaluate $z(u_k, t)$ and $z(u_{\text{try}}, t)$ adaptively with the error bound (4.2). This error bound depends on $tol_{E,k}$, which is not a priori known. Thus, we cannot use the error bound directly. Instead, we add a feedback loop to a standard backtracking line search algorithm and use $tol_{E,k-1}$ as the initial tolerance. As initial segmentation for the adaptive quadrature algorithm, we use the final segmentation $\mathcal{T}_{\delta v_k}$ of the step computation. This leads to the modified backtracking algorithm 5, which we describe now.

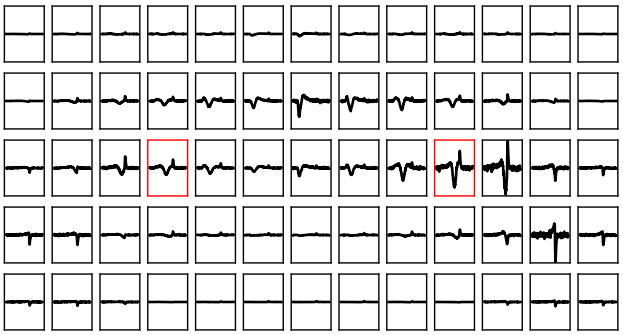
Algorithm 5: Computation of the damping factor β

Input: u_k, δv_k, $J_2(u_k, \xi)$, $J_2'(u_k, \xi)\delta v_k$, $\mathcal{T}_{\delta v_k}$, $tol_{E,k} = tol_{E,k-1}$;
Parameter: $\varsigma = 0.9$, $\eta = 10^{-3}$, $\sigma = 0.01$, $\beta = 2$;
do
  do
    $\beta \leftarrow 0.5\beta$ and $u_{\text{try}} \leftarrow u_k + \beta \delta v_k$;
    compute $J_2(u_{\text{try}}, \xi)$ on $\mathcal{T}_{FE}$;
    $z_G(u_k, t), z_K(u_k, t), z_G(u_{\text{try}}, t), z_K(u_{\text{try}}, t), \mathcal{T}_\beta \leftarrow$ Alg. 3
      (Input: u_k, u_{try}, $\mathcal{T}_{\delta v_k}$, $tol_{E,k}$, and error bound (4.2));
    compute $z_K'(u_k, t)\delta v_k$ on $\mathcal{T}_\beta$;
    compute $J_{1,G}(u_k)$, $J_{1,K}(u_k)$, $J_{1,K}'(u_k)\delta v_k$, $J_{1,G}(u_{\text{try}})$, and $J_{1,K}(u_{\text{try}})$;
    compute $\mathcal{L}_G(u_k, \xi)$, $\mathcal{L}_K(u_k, \xi)$, $\mathcal{L}_K'(u_k, \xi)\delta v_k$, $\mathcal{L}_G(u_{\text{try}}, \xi)$ and $\mathcal{L}_K(u_{\text{try}}, \xi)$;
  while $\mathcal{L}_K(u_{\text{try}}, \xi) - \mathcal{L}_K(u_k, \xi) > \beta\eta \mathcal{L}_K'(u_k, \xi)\delta v_k$;
  $tol_{E,k} = \frac{2\varsigma}{T} |\mathcal{L}_K(u_{\text{try}}) - \mathcal{L}_K(u_k)|$;
  if $\mathcal{L}_{Q_\beta}'(u_k, \xi)\delta v_k \geq \sigma \mathcal{L}_{Q_{\delta v_k}}'(u_k, \xi)\delta v_k$ **then**
    terminate "(3.9) violated";
  end
while $|(\mathcal{L}_K(u_{\text{try}}, \xi) - \mathcal{L}_K(u_k, \xi)) - (\mathcal{L}_G(u_{\text{try}}, \xi) - \mathcal{L}_G(u_k, \xi))| \geq tol_{E,k}$;
terminate "(3.8) and estimated (3.4) fulfilled";
Output: β;

We start with $\beta = 1$ and set $u_{\text{try}} = u_k + \beta \delta v_k$. As before, we can compute $J_2(u_{\text{try}}, \xi)$ without resorting to adaptive quadrature. Furthermore, we can reuse $J_2(u_k, \xi)$ and $J_2'(u_k, \xi)$ computed during the computation of δu_k. To obtain $J_{1,G}(u_{\text{try}})$, $J_{1,K}(u_{\text{try}})$, $J_{1,G}(u_k)$, and $J_{1,K}(u_k)$, we compute $z(u_k, t)$ and $z(u_{\text{try}}, t)$ with the adaptive quadrature algorithm 3. Thereby, we must ensure that both quantities must satisfy the error bound (4.2). Thus, we slightly modify the adaptive quadrature algorithm such that it computes $z(u_k, t)$ and $z(u_{\text{try}}, t)$ simultaneously with the same Gauss-Kronrod rule. The modifications read as follow: Since $z(u_k, t)$ and $z(u_{\text{try}}, t)$ are independent, we can compute the local quantities $z_{G,I}(u_k, t)$, $z_{K,I}(u_k, t)$, $z_{G,I}(u_{\text{try}}, t)$, $z_{K,I}(u_{\text{try}}, t)$,



(a) Cylindrical domain with 65 electrodes



(b) Simulated measurement for a reference motor unit at 65 electrodes

Figure 1: Geometric setting and simulated measurements

and $E_I(u_k, u_{try}, t)$ separately while iterating over the segmentation $\mathcal{T}_j$ (starting with $\mathcal{T}_{\delta v_k}$). After computing the global values as usual, we test if (4.2) is satisfied. If this is not the case, we choose a refinement partition $\mathcal{T}_R$ such that (4.1) is satisfied. As usual, we refine $\mathcal{T}_j$ and recompute $z_{G,I}(u_k, t)$, $z_{K,I}(u_k, t)$, $z_{G,I}(u_{\text{try}}, t)$, and $z_{K,I}(u_{\text{try}}, t)$ on the refined intervals.
As before we can compute $z'_K(u_k, t)\delta v_k$ on the final segmentation $\mathcal{T}_\beta$ and then all other quantities needed for evaluation of the Armijo condition. If the latter is violated, we multiply β by 0.5 and repeat the process until the Armijo condition is met. When we have found an acceptable β, we update $tol_{E,k}$ and test if (3.4) is still satisfied. If this is not the case, we recompute the tolerance-dependent quantities and continue with the line search. This final test ensures that (3.4) is satisfied at termination. It seldom fails since we already used the adaptive quadrature to control this error.

5. Numerical Example. In the final chapter of this paper, we study a numerical example to observe the behavior of our algorithm. Additionally, we discuss the influence of the quality of impulse response functions on the convergence. Our algorithm was implemented in C++, using the DUNE library, see [5], for mesh-related operations and Eigen for the linear algebra, see [8]. We used the FE toolbox Kaskade 7, see [10], to precompute the impulse response functions.

5.1. General Setup. For our numerical test, we use a cylindrical domain representing a part of a limb, e.g., the middle part of an upper arm, consisting of three layers. The inner layer represents a bone (white), the middle layer represents muscle tissue (orange), and the outer layer is fat tissue (red). Furthermore, we consider a grid of $5 \times 13 = 65$ electrodes (black circles) on the skin (beige). The cylinder has a length of $18cm$ and a diameter of $8cm$. The bone has a diameter of $2cm$, the muscle tissue has a diameter of $7cm$, and the fat layer has a depth of $0.5cm$. As shown in Figure 1a, we shifted the bone and placed the electrodes on the opposite boundary. We used gmsh, see [9], to create a FE conforming mesh from the geometry.
Concerning the triangulation, there are two domains of interest where we want the mesh to be sufficiently fine. The first region is the domain where the electrodes are placed. Due to the circular shape of the electrodes and a possible curved skin, the mesh must be sufficiently fine in this region. The second region is that part of the muscle tissue, where the motor unit is roughly known to be located. This area was chosen as a cylinder with a radius of $0.5cm$ around the axis $(x,\ 0.,\ 0.029)$. We refined the mesh inside this cylinder two resp. three times resulting

in two different meshes containing 82581 resp. 181084 tetrahedra, which we will use later for comparison purposes.
For the approximation of a motor unit an FE discretization of $[-1, 1]$ that consists of 19 subintervals is sufficient. Therefore, the matrix representation of $H(u, \Lambda)$ is only a 120×120 matrix, such that we can use a direct solver, i.e., a Cholesky decomposition, to solve (3.5). The matrix for computing the update of the Lagrange multiplier is similarly small, but, in contrast to $H(u, \Lambda)$, it is sparse.
We created synthetic measurement data by simulating measurements for a given reference motor unit. Analytically our reference motor unit is given through

$$u(t) = \begin{pmatrix} \cos(\pi/90)\sin(0.38t) - 0.02\sin(\pi/90) \\ 0.1\cos(0.38t) \\ \sin(\pi/90)\sin(0.38t) + 0.02\cos(\pi/90) \end{pmatrix} \quad \text{for } t \in [-1, 1].$$

Figure 2 shows two views of the reference motor unit (cyan). The left picture visualizes the positions relative to the electrodes, and the right picture shows the change of depth, i.e., the distance change away from the electrodes. This setting combines two critical difficulties, which are usually encountered separately in practice: the change of depth (e.g., in facial muscles) or a curved motor unit (e.g., in the biceps). It is thus slightly more complicated than most practical use cases.
To simulate the measurements, we divided the time interval $[0, 2]ms$ into 200 equidistantly distributed measure points t_i and computed for each measure-point $y(u, t_i)$ with (2.5). We used the adaptive quadrature algorithm 3 with a standard relative error criterion and tolerance 10^{-9} to compute the integral. To create a realistic measurement, we finally added white noise with a data-to-noise ratio of 5%. Figure 1b shows the 65 simulated measurements in a grid. The measurements contained in the red boxes are depicted in detail in Figure 3 as reference measurements in the discussion of the example.
Looking at the grid of simulated measurements in Figure 1b we can already guess roughly the trajectory of the motor unit visually. This yields an initial guess for our algorithm, which we have chosen as a straight line in the region of the reference trajectory. In Figure 2, the initial trajectory is visualized in red. We then used the presented algorithm to identify the reference motor unit from the simulated measurement.

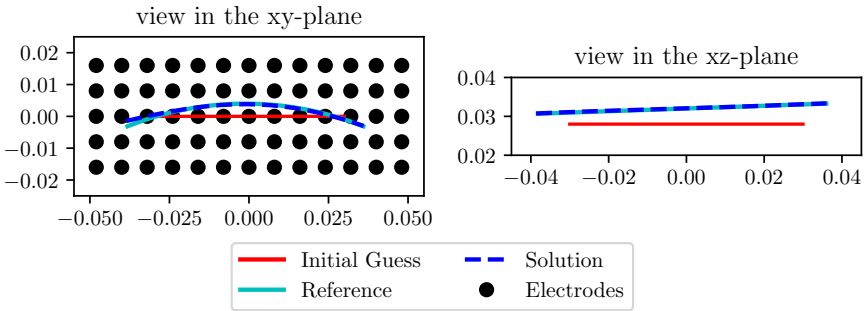


Figure 2: Comparison of the computed solution with the reference trajectory

5.2. Numerical Results. We start with a qualitative discussion of the identification, using impulse response functions computed with cubic ansatz polynomials on a grid that was refined

three times in the area of the motor unit. Figure 2 compares the reference motor unit, shown in cyan, with the computed solution, shown in blue. The right picture shows that the changing depth is identified very well. As seen in the left picture, the curved trajectory is also identified well with a slight deviation at the left side where the motor unit is further away from the electrodes. Figure 3 compares the measurement produced by the reference trajectory with the measurement generated from the computed solution for two selected electrodes. We can see that both measurements fit well and mainly differ in the added noise. This observation underlines that the algorithm is suitable to identify a motor unit from a given sEMG measurement.

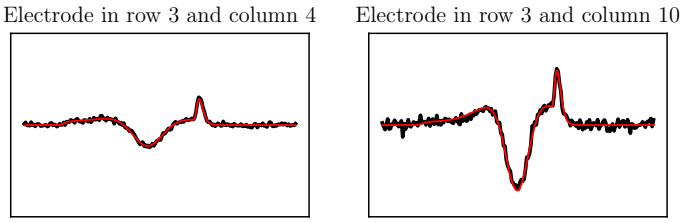


Figure 3: Comparison of simulated (black) and identified measurement (red).

Performance study. Figure 4 shows that the algorithm needs 33 Newton steps distributed over 7 augmented Lagrangian steps. The first augmented Lagrangian step requires the most Newton steps, since it has the least accurate initial guess, and thus globalization is active. After this first iteration, all other augmented steps require 2 to 5 Newton iterations, but we do not see superlinear convergence. We attribute for this observation to the non-smoothness of the discretized impulse response functions $\omega_{k,h}$. Therefore, small jumps in the first derivative, occur at the boundary between two adjacent tetrahedra. We will later observe the influence of the impulse response functions in more detail.

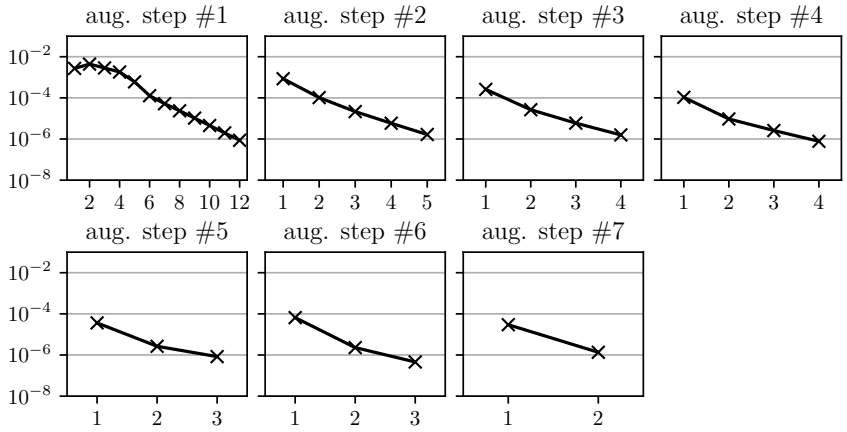


Figure 4: Energy norm of $\|\delta v_k\|_H$ during all augmented Lagrangian steps

Figure 5 visualizes the properties of the augmented Lagrangian method. We see that $\|G(u)\|_{2,2}$ converges linearly to zero. To achieve this, the algorithm alternates between increasing the

penalty parameter μ and updating the Lagrange multiplier λ.

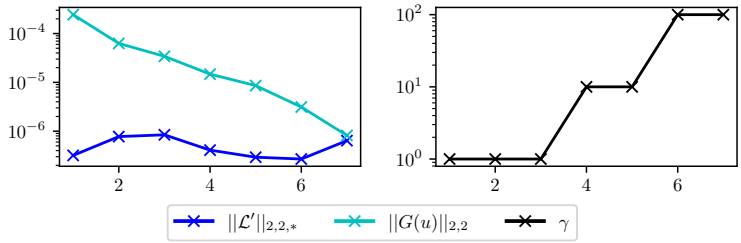


Figure 5: Key data of the augmented Lagrangian algorithm

Lastly, Figure 6 compares the tolerance, given to the adaptive quadrature, with the size of the resulting segmentation of the integration domain. For the sake of brevity we visualize the average and maximal number of points among all used segmentations in each step and concentrate our discussion on the first augmented Lagrangian loop. Since both tolerances depend on differences that get very small during the algorithm, the adaptive quadrature algorithm refines the segmentation more for the last steps of the algorithm. Next, we notice that the average number of grid points is much lower than the maximal. This observation indicates that the adaptive quadrature needs only a fine segmentation for a few measure points and underlines the theoretical considerations in Section 3.2 and justifies using adaptive quadrature.

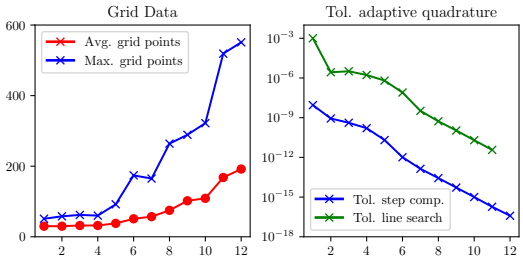


Figure 6: Comparison of the average and maximal size of adaptive quadrature grids.

Influence of Impulse Response Functions. Next, we discuss the influence of the discretization of the impulse response functions $\omega_{k,h}$ by finite elements. Discretization introduces small discontinuities of the derivatives $\omega'_{k,h}$ at the facets of the triangulation, which may affect the performance of our algorithm. To assess the situation, we solve the above problem using two different grid resolutions and, on each grid, finite elements of degrees two and three.
In our numerical experience the inner Newton method behaves, after the first augmented Lagrangian step, similarly for all impulse response functions and only differs in the first augmented Lagrangian step, as shown in Figure 7. We see that the inner SQP-Newton method requires more iterations using the coarse grid (2 refinements) and polynomials of order 2. After the first Newton iteration, the trajectory is already identified quite well and the remaining augmented Lagrangian steps mainly optimize the parameterization, which needs 2 to 7 Newton steps per augmented Lagrangian step.
Size and order of the finite element space also influence the computational times for each step. Here, we notice that increasing the polynomial degree from two to three approximately doubles

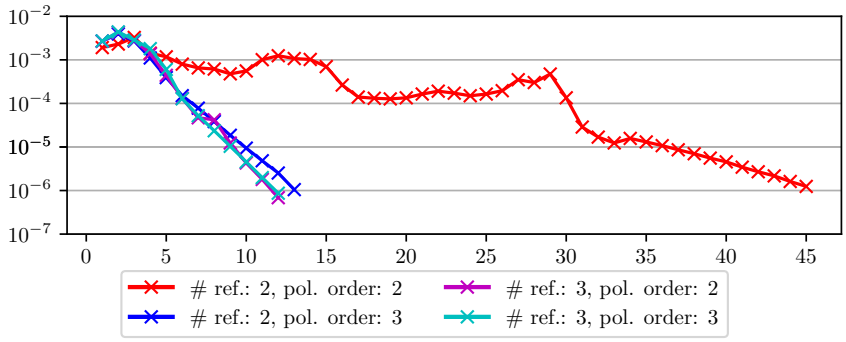


Figure 7: Comparison of the energy norms of δv_k for different impulse response functions

the computation time. Since changing the polynomial degree from two to three, increases the degrees of freedom per tetrahedron from 10 to 20, we could expect this behavior. Refining the grid increases the computation times only abount 50%.

ref./ degree	Newton steps	avg. points in $[-1,1]$	avg. time simulation [s]	avg. time Newton step [s]	total time [s]	time to compute $\omega_{k,h}$ [s]
2/2	69	88.7	1.33	4.47	299	158.72
2/3	35	84.1	2.54	8.85	285	1135.22
3/2	40	134	1.81	6.54	246	557.04
3/3	33	78.4	2.69	8.15	249	4664.88

Table 1: Comparing results for different impulse response functions $\omega_{k,h}$

Including the times needed to compute the impulse response functions in our discussion, we observe that increasing the polynomial degree increases the computation time by a factor of 7, whereas refining the domain increases the computation time only by a factor of 3.5. Thus, if the optimization problem is solved only once, the computation of the impulse response function is possibly too time-consuming when using polynomials of degree 3. But if the optimization problem is solved multiple times, e.g., when identifying different motor units in one muscle, then polynomials of higher degrees are the better choice. In addition, higher-degree finite elements may yield more accurate identification results.

6. Conclusion. We conclude that our optimization algorithm is capable of identifying motor units from sEMG measurements in an efficient way, at least in the context of synthetically generated measurements. This was made possible by exploiting the specific problem structure and the use of adaptive quadrature for efficient computation.
Still, there are a few open questions. As the example has shown, the quality of the impulse response functions has a non-negligible influence on the convergence of the SQP-Newton method. Thus, their influence should be examined in more detail and the computation of higher order functions should be accellerated, e.g., by using a hierarchical basis in the polynomial degree with a suitable preconditioner (cf. e.g. [16]). Finally, testing the algorithm with real measurements should be the objective of further studies.

[1] Dimitri P. Bertsekas, *Nonlinear programming*, 2. ed., 3. print ed., Athena Scientific, Belmont, Mass., 2008.

[2] Haim Brezis, *Functional Analysis, Sobolev Spaces and Partial Differential Equations*, Universitext, Springer Science+Business Media LLC, New York, NY, 2010.

[3] Richard G. Carter, *Numerical optimization in hilbert space using inexact function and gradient evaluations*, Technical report 89-45, ICASE, Langley (1989).

[4] Elise de Doncker, *An adaptive extrapolation algorithm for automatic integration*, ACM SIGNUM Newsletter **13** (1978), no. 2, 12–18.

[5] Andreas Dedner, Bernd Flemisch, and Robert Klöfkorn, *Advances in DUNE: Proceedings of the DUNE User Meeting, held in October 6th-8th 2010 in Stuttgart, Germany*, Springer, Berlin, 2012.

[6] Ron S. Dembo and Trond Steihaug, *Truncated-newton algorithms for large-scale unconstrained optimization*, Mathematical programming **26** (1983), no. 2, 190–212.

[7] Peter Deuflhard, *Global inexact newton methods for very large scale nonlinear problems*, IMPACT of Computing in Science and Engineering **3** (1991), no. 4, 366–393.

[8] Gaël Guennebaud, Benoît Jacob, et al., *Eigen v3*, http://eigen.tuxfamily.org, 2010.

[9] Christophe Geuzaine and Jean-François Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, International Journal for Numerical Methods in Engineering **79** (2009), no. 11, 1309–1331.

[10] Sebastian Götschel, Martin Weiser, and Anton Schiela, *Solving Optimal Control Problems with the Kaskade7 Finite Element Toolbox*, Advances in Dune, Springer, Berlin, 2012, pp. 101–112.

[11] Roberta Grech, Tracey Cassar, Joseph Muscat, Kenneth P. Camilleri, Simon G. Fabri, Michalis Zervakis, Petros Xanthopoulos, Vangelis Sakkalis, and Bart Vanrumste, *Review on solving the inverse problem in EEG source analysis*, Journal of neuroengineering and rehabilitation **5** (2008), 25.

[12] Wolfgang Hackbusch, *Elliptic differential equations*, vol. 18, Springer Berlin Heidelberg, Berlin, Heidelberg, 2017.

[13] Matthias Heinkenschloss and Luis N. Vicente, *Analysis of inexact trust-region sqp algorithms*, SIAM Journal on Optimization **12** (2002), no. 2, 283–302.

[14] D. P. Kouri, M. Heinkenschloss, D. Ridzal, and B. G. van Bloemen Waanders, *Inexact objective function evaluations in a trust-region algorithm for pde-constrained optimization under uncertainty*, SIAM Journal on Scientific Computing **36** (2014), no. 6, A3011–A3029.

[15] Aleksandr S. Kronrod, *Nodes and weights of quadrature formulas*, Consultants Bureau Enterprises, 1965.

[16] Sabine Le Borne, *Hierarchical preconditioners for high-order FEM*, Domain decomposition methods in science and engineering XXII, Lect. Notes Comput. Sci. Eng., vol. 104, Springer, Cham, 2016, pp. 559–566.

[17] C J Lin and J J More, *Incomplete Cholesky factorizations with limited memory*, SIAM Journal on Scientific Computing **21** (1999).

[18] Yang Liu, Yong Ning, Sheng Li, Ping Zhou, William Z. Rymer, and Yingchun Zhang, *Three-Dimensional Innervation Zone Imaging from Multi-Channel Surface EMG Recordings*, International Journal of Neural Systems **25** (2015), no. 6, 1550024.

[19] M. M. Lowery, *EMG Modeling and Simulation: 8*, Surface Electromyography : Physiology, Engineering, and Applications, John Wiley & Sons, Ltd, 2016, pp. 210–246.

[20] Luca Mesin, *Real time identification of active regions in muscles from high density surface electromyogram*, Computers in Biology and Medicine **56** (2015), 37–50.

[21] Christian Meyer, Lucia Panizzi, and Anton Schiela, *Uniqueness Criteria for the Adjoint Equation in State-Constrained Elliptic Optimal Control*, Numerical Functional Analysis and Optimization **32** (2011), no. 9, 983–1007.

[22] Jorge Nocedal and Stephen J. Wright, *Numerical optimization*, Springer series in operations research, Springer, New York, NY, 1999.

[23] R. Piessens, *Quadpack: A subroutine package for automatic integration*, Springer Series in Computational Mathematics Ser, vol. v.1, Springer Berlin / Heidelberg, Berlin, Heidelberg, 1983.

[24] Tobias Sproll and Anton Schiela, *An adjoint approach to identification in electromyography: Modeling and first order optimality conditions*, Inverse Problems (2021).

[25] Guido Stampacchia, *Le problème de Dirichlet pour les équations elliptiques du second ordre à coefficients discontinus*, Annales de l'institut Fourier **15** (1965), no. 1, 189–257.

[26] Kees van den Doel, Uri M. Ascher, and Dinesh K. Pai, *Computed myography: three-dimensional reconstruction of motor functions from surface EMG data*, Inverse Problems **24** (2008), no. 6, 065010.

[27] ———, *Source localization in electromyography using the inverse potential problem*, Inverse Problems **27** (2011), no. 2, 025008.

[28] J. Carsten Ziems and Stefan Ulbrich, *Adaptive multilevel inexact sqp methods for pde-constrained optimization*, SIAM Journal on Optimization **21** (2011), no. 1, 1–40.