



Knowledge-based modeling of simulation behavior for Bayesian optimization

Felix Huber^{1,2} · Paul-Christian Bürkner^{3,4} · Dominik Göddeke^{1,3} · Miriam Schulte^{2,3}

Received: 17 June 2023 / Accepted: 27 November 2023
© The Author(s) 2024

Abstract

Numerical simulations consist of many components that affect the simulation accuracy and the required computational resources. However, finding an optimal combination of components and their parameters under constraints can be a difficult, time-consuming and often manual process. Classical adaptivity does not fully solve the problem, as it comes with significant implementation cost and is difficult to expand to multi-dimensional parameter spaces. Also, many existing data-based optimization approaches treat the optimization problem as a black-box, thus requiring a large amount of data. We present a constrained, model-based Bayesian optimization approach that avoids black-box models by leveraging existing knowledge about the simulation components and properties of the simulation behavior. The main focus of this paper is on the stochastic modeling ansatz for simulation error and run time as optimization objective and constraint, respectively. To account for data covering multiple orders of magnitude, our approach operates on a logarithmic scale. The models use a priori knowledge of the simulation components such as convergence orders and run time estimates. Together with suitable priors for the model parameters, the model is able to make accurate predictions of the simulation behavior. Reliably modeling the simulation behavior yields a fast optimization procedure because it enables the optimizer to quickly indicate promising parameter values. We test our approach experimentally using the multi-scale muscle simulation framework OpenDiHu and show that we successfully optimize the time step widths in a time splitting approach in terms of minimizing the overall error under run time constraints.

Keywords Bayesian optimization · Constrained optimization · Numerical simulation · Bayesian modeling · Multi-scale simulation

1 Introduction

To solve complex multi-X problems numerically, simulations are typically composed of many components, which in turn are parameterized in many ways. For instance, there are components that capture individual physics phenomena on their individual spatial and temporal scales, realized by individual software solutions, and there are components that capture their complex and challenging interplay across scales and aspects specific to the used hardware. Such components can include discretization parameters, solver accuracies, coupling schemes and hardware specific algorithms. In an ideal world, such a parameterization of the involved numerical schemes, their implementation, and their interplay would be determined automatically, to achieve an overall optimal balance between the quality of solution and time to solution. The quality can, e.g., be measured in terms of accuracy or precision of the simulation result, or, more generally speaking, in terms of gained knowledge. The optimal setting of parameters must satisfy various constraints, some given by

✉ Felix Huber
felix.huber@mathematik.uni-stuttgart.de

Paul-Christian Bürkner
paul.buerkner@gmail.com

Dominik Göddeke
dominik.goeddeke@mathematik.uni-stuttgart.de

Miriam Schulte
miriam.schulte@ipvs.uni-stuttgart.de

¹ Institute of Applied Analysis and Numerical Simulation, University of Stuttgart, Allmandring 5b, 70569 Stuttgart, Germany

² Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany

³ Stuttgart Center for Simulation Science, University of Stuttgart, Universitätsstraße 32, 70569 Stuttgart, Germany

⁴ Department of Statistics, TU Dortmund University, Vogelpothsweg 78, 44227 Dortmund, Germany

the simulated problem, some given by the interplay of components, some given by the implementation choices, and some induced by the simulation pipeline itself. Examples include constraints on the simulation time and error, time step width limitations for stability reasons, hardware constraints that effect suitable algorithms or parallelization methods. The state of the art in this scenario is based on employing expert knowledge and expert experience from the methods side and the domain side. However, typical users of simulations are not always equipped with this knowledge or experience.

In this paper, we use “simulations” as an encompassing term for finding numerical approximations to forward problems and deliberately restrict the discussion to exemplary multi-scale problems governed by PDEs. We assume an existing understanding of the underlying simulation components, and focus mostly on the interplay of the components. This is already a challenging problem, because the efficiency of one component often depends on choices for other components and simply trying every parameter combination is infeasible. Choosing good combinations by expert “intuition” requires years of experience—which is naturally not given for new challenging simulations.

1.1 Motivating example

A basic example for combinations of components and parameters is the balancing of multiple resolution parameters to minimize the numerical error without wasting resources by refining too much in one direction. Such problems arise for different time step widths in time splitting schemes for multi-physics or multi-scale simulations or when balancing spatial and temporal discretization widths. If the error induced by one of the discretization parameters is already orders of magnitude smaller than the error induced by the other one, further refinement of the first only increases the computational cost without impacting the overall error in a measurable manner. Even if stability bounds, such as the Courant–Friedrichs–Lewy (CFL) condition, limit the relation of the discretization widths in some settings, such bounds only ensure numerical stability and do not ensure spatial and temporal error balancing.

In all these scenarios, we typically have some “textbook” information at hand about asymptotic error behavior in the form of convergence orders, but we do not know the respective constants, which makes the problem of choosing several discretization widths non-trivial. More complex examples, which we envision to solve with similar methods as those presented in this paper, include considering all kinds of optimal combinations of further methodological components such as the choices of the discretization scheme, discretization width, linear solver and stopping criterion of the solver. These questions can also be addressed in terms of classical simulation

adaptivity, while in this paper, we aim to establish an alternative, minimally invasive, more data-driven approach.

1.2 General formulation

We can formalize the general problem we want to address in the framework of optimization comprising an objective function $f: \Omega \rightarrow \mathbb{R}$ and a constraint function $g: \Omega \rightarrow \mathbb{R}$ with an upper bound g_{\max} . Typically, the objective function is the simulation error and the constraint is the run time, or vice versa. The admissible set Ω parameterizes the simulation components and can be either continuous or discrete. The stopping criterion of a linear solver is an example for a continuous parameter, while a solver type is chosen from a discrete set. However, depending on the setting, continuous parameters can also be restricted to a discrete set if, for example, we only allow powers of $\frac{1}{2}$ as discretization width. The optimization functions f and g are unknown and have to be learned from function evaluations, each of which means that we have to execute a simulation in our setting. This also implies that the feasible region $\{x \in \Omega \mid g(x) \leq g_{\max}\}$ is not explicitly given and has to be learned together with the constraint function g . The final optimization problem then reads

$$\operatorname{argmin}_{x \in \Omega} f(x) \text{ s.t. } g(x) \leq g_{\max}.$$

1.3 Knowledge-infused modeling for f and g

Without knowing anything about f and g , the optimization problem is difficult to solve and severely ill-posed. However, typical simulation behavior of subcomponents is not a black-box function: The individual components used inside a simulation are often numerical methods for which qualitative behavior is known a priori at least asymptotically, e.g., in terms of convergence rates, error reduction or computational complexity. For example, an Euler scheme is first order in time, while a Crank–Nicolson scheme is second order in time. This knowledge is very valuable, as it gives a good idea of how the simulation is expected to behave over a large range of the optimization space. Thus, we suggest to explicitly exploit this knowledge during the optimization. To predict f and g quantitatively, we learn models for f and g from data and the a priori knowledge. As simulations are usually performed in the pre-asymptotic regime of the numerical methods and as measurements of, e.g., run time, are usually noisy, we additionally enhance the model by terms that allow for deviations from the ideal asymptotic behavior.

1.4 Challenges

The main challenges of solving the outlined optimization problem by infusing a priori knowledge about the qualitative behavior of f and g are:

Expensive evaluations The optimizer has to work with data that are potentially expensive to generate and are, thus, scarce: Each evaluation of f and g corresponds to executing at least one simulation run and additional computations to estimate the simulation error and run time. Such simulation runs and the computation of error estimators are expensive. Therefore, the number of such expensive evaluations has to be kept small to keep the optimization cost acceptable. This, however, requires function models that can extrapolate with sufficient accuracy from these evaluation points to usually further away regions where we might find the optimum. Overall, the cost of running a simulation for specific parameters is assumed to be the dominating cost factor during the optimization. Therefore, it is important to devise a data-efficient optimization that uses available data optimally during the optimization, i.e., an optimization that carefully chooses new evaluation points in combination with a model ansatz that allows for data extrapolation instead of only interpolation.

Scales covered by function values The optimization has to work with data spanning many orders of magnitude. Typical values from simulations such as error norms or run times can quickly converge to zero or increase to very large values, depending on the parameters of the simulation, e.g., in terms of step widths of discretizations. While this is in general not necessarily a problem for fitting such functions in a regression approach, it becomes an issue in our case due to the deviations from the expected ideal asymptotic behavior that we mentioned earlier. Their scale and also the scale of model uncertainties between data points depend on the scale of the predicted data values. This means, that we assume larger absolute deviations and uncertainties for large values and smaller deviations for smaller values. As the function values of f and g span over multiple orders of magnitude, this influences the modeling process, as the model has to be accurate on all scales.

Non-convexity and local minima Simulations often consist of many different interacting components which influence the simulation outcome and cannot be optimized individually. Interactions of components typically also cause both the objective function and the feasible region to be non-convex. In particular, the optimum can be in the interior of the feasible region and we might have to face several local minima. An example with an optimum inside the feasible region would be a simulation where a finer temporal discretization reduces the run time as it decreases the number of iterations of iterative

solvers because the smaller time step width makes the problem in each time step easier to solve. Several local optima might be caused by sweet spots where, e.g., errors cancel out for specific combinations of parameters.

1.5 Summary of contributions

We propose an optimization approach specifically suited for data from a simulation context. The approach infuses a priori knowledge in the models and combines these models with Bayesian optimization. We make the following specific contributions:

Bayesian optimization We suggest a Bayesian optimization approach: In each optimization step, known data are used to build a stochastic model of the optimization functions, which is then used to select new evaluation points. This allows the optimizer to work with scarce data and gives the flexibility to adjust the internal model.

Knowledge infusion To build accurate models of the optimization functions, we infuse available a priori knowledge about the objective function f and the constraint g into the modeling process. This allows the model to make extrapolated predictions further away from known data points and reduces the number of function evaluations necessary to find a good approximation of the optimum.

Logarithmic scaling To deal with data ranging across multiple orders of magnitude, we model the optimization function on a logarithmic scale. This also allows modeling uncertainties and deviations from the a priori infused asymptotic behavior as additive terms on the logarithmic scale, which corresponds to a multiplicative factor on the original scale. Modeling on a logarithmic scale in addition ensures that function values in our model are always positive for quantities such as simulation errors or run times.

Prior choice We show how a weakly informative prior can be applied for the qualitative a priori knowledge on the logarithmic scale. This matches the modeling approach on the logarithmic scale and significantly reduces spurious trends in the posterior distribution.

2 Related work

The need to optimize simulation components is not new [25]. However, often the parameters are optimized manually, heuristically or set to values taken from literature and only adjusted when problems arise. Automatically optimized parameters are typically present in the context of adaptivity. Adaptivity adjusts parameters during the runtime of a simulation, frequently using local criteria. Examples include

adaptive time step sizes [20] while ensuring stability based on CFL conditions, adaptive stopping criteria for linear solvers [31] or inexact non-linear Newton solvers [11] based on residuals, or spatial grid adaptivity based on local error estimators, using hierarchical surpluses [9] or a posteriori estimates [4, 12].

Performance modeling techniques that focus on hardware aspects, such as the roofline model [10, 43], complement these mathematical approaches and help to select well-performing algorithms for the used hardware.

Adaptive methods often restrict adaptivity to selected parameters that can easily be changed during the simulation [11, 20, 31], and hence neglect the interplay between different simulation components. Usually, they do not allow to optimize a global objective such as a global error norm under global constraints such as limited overall runtime. Also, it is sometimes easier to optimize parameters on subproblems [8] instead of the full simulation setup. This means that many parameters remain unchanged during the simulation and reaching an overall optimum can not be expected. While adaptive methods can change solver properties dynamically, they usually come at a relatively high implementation cost and are not always feasible for all parameters [4, 12].

In contrast to dynamical adaptivity, external parameter optimization can improve existing simulations without modifying the underlying algorithms. This noninvasive approach is particularly useful when the parameters remain constant throughout the simulations. To solve a global parameter optimization problem externally, many different methods can be used: When evaluations of the objective and constraint functions are relatively cheap, many evaluations of these functions are feasible and direct grid-based or random search methods can be used [5, 6]. In our setting, each function evaluation is expensive because each evaluation corresponds to a simulation run. Therefore, the optimization methods have to be more efficient in terms of knowledge gained from an evaluation, but can spend more time in selecting suitable data points. Model-based optimization methods address this issue: They build an internal model of the unknown objective and constraint functions that is used as an easy to evaluate surrogate in the optimization process. Commonly used models include random forests [13, 23], geometric models [32] and Gaussian processes [38]. Such models can then be used in a Bayesian setting to simultaneously approach the optimum and improve the models by selecting new data points according to different selection strategies [14, 21, 24, 33, 39, 42]. However, in many of these settings little is known about the functions to be optimized which are therefore commonly treated as black-box functions with few prior assumptions [44]. This is in contrast to our setting, where knowledge about the simulation can be used:

The simulation behavior is usually not a black-box and stronger a priori knowledge is available. For example, a pri-

ori knowledge in the form of global trends can be included by parameterizing the mean function of a Gaussian process with additional basis functions [34–36] which also yields a Gaussian process posterior. However, this approach does not ensure that the model is, e.g., positive and it is difficult to model deviations relative to the global trend. Approaches as in [1] and [2] make use of a known composite structure in the objective function by only modeling the unknown components as Gaussian processes and computing the known parts explicitly. This approach does not directly transfer to our setting as it does not model a global trend. Other, more flexible properties that can be expressed as linear constraints for a Gaussian process such as monotony or convexity are discussed in [26, 27].

Outside the simulation parameter optimization context, optimization problems with unknown constraints are often solved by modeling the objective and the constraint explicitly inside a Bayesian optimization as in [16, 17], or combined with classical constrained optimization algorithms as in [19].

3 Knowledge-infused modeling of objectives and constraints

To deal with scarce and expensive data, we use a Bayesian optimization approach for active learning. In this section, we explain all components of this approach with a special focus on the model ansatz for the stochastic approximation of the objective and constraint functions f and g including the choice of a suitable prior for the respective model coefficients.

3.1 Bayesian optimization overview

The Bayesian optimization approach approximates the optimal parameter setting for our numerical methods in an iterative process. In each iteration, it simultaneously improves the models for the objective and the constraint while trying to get closer to the optimum. Each iteration consists of the following steps:

Based on N known data points $\{(x_i, f(x_i), g(x_i))\}_{i=1}^N$, we build a *model* of the objective and constraint functions using the knowledge-based model described later in Sect. 3.2. More specifically, we approximate the posterior distribution for these models.

Then, an *acquisition function* is evaluated based on the model and its predictions. The acquisition function is used to approach the optimum by selecting a new promising evaluation point. For this, the acquisition function has to be maximized, either by using an optimization method (such as in [3]) or by evaluating all possible evaluation points if the parameter space is discrete and small enough. The acquisition function steers the optimization process by choosing

points that are closer to the optimum or that reduce the model uncertainty.

Last, the optimization functions f and g are *evaluated* at the new evaluation point. This is the cost-dominating step in our setting, as it requires at least one new simulation run. In our setting, this step also includes additional computations such as estimating the simulation error or measuring the run time. The simulation result yields a new data point $(x_{i+1}, f(x_{i+1}), g(x_{i+1}))$ for the next optimization step.

The optimization can be stopped after a fixed number of iterations or when the acquisition function becomes small, which indicates that little further improvement is expected. For a more detailed introduction into Bayesian optimization, we refer to [15, 17, 37].

Note that, while many different acquisition functions exist, they all depend on a good model for the optimization functions in order to give good results. Therefore, we consider the model ansatz as a natural starting point to add a priori knowledge into the optimization in order to improve the performance and to use sparse and expensive data efficiently.

In the following, we discuss how models for numerical methods can be build that exploit the available knowledge.

3.2 Model ansatz

Our model for both the objective and the constraint consists of two parts: A global model μ that is based on the a priori knowledge and a local model s for deviations from the global model. We combine both model components to approximate the optimization functions on a logarithmic scale. We model the objective $f : \Omega \rightarrow \mathbb{R}$ in a stochastic setting with a random variable as

$$\ln \hat{f}(x) = \mu(x) + s(x). \tag{1}$$

We now give an overview on each sub-model for f . The modeling of the constraint g is done analogously.

As global model $\mu(x)$ we use the logarithm of a linear combination of q basis functions φ_j representing the a priori knowledge, e.g., on asymptotic convergence rates of a numerical method,

$$\mu(x) = \ln \left(\sum_{j=1}^q \beta_j \varphi_j(x) \right)$$

with coefficients $\beta = (\beta_1, \dots, \beta_q) \in B \subseteq \mathbb{R}^q$ which are restricted to a set B such that the linear combination is positive. The global model is meant to give a good representation of global trends in the optimization functions on a logarithmic scale, which enables the model to make predictions further away from known data points. Hence, promising parameter regions can be identified for further investigation in the optimization process and less promising regions can be ruled

out based on only few data points. As an example, the error of a second order time stepping scheme with step width h as parameter could use the basis function h^2 in the global model with $x = h$ to represent the asymptotic convergence order for $h \rightarrow 0$. Similarly, the basis function h^{-1} might represent the expected run time scaling in the global model for the run time constraint in this example. The coefficients $\beta = (\beta_1, \dots, \beta_q) \in \mathbb{R}^q$ are random variables inferred in our Bayesian approach as discussed in Sect. 3.3.

The local model $s : \Omega \rightarrow \mathbb{R}$ accounts for deviations from the global model and is modeled as a zero-mean Gaussian process with covariance kernel function $k : \Omega \times \Omega \rightarrow \mathbb{R}$:

$$s \sim \mathcal{GP}(0, k)$$

This allows corrections to μ due to non-modeled higher order terms, pre-asymptotic behavior or local effects such as error cancellation of different simulation components. In the current model, we do not include measurement noise such that our measured data are exact function evaluations of f and g . However, measurement noise can be included by an extra additive term in the model, representing pointwise deviations. In our experiments we consider an optimization problem, where we model pointwise deviations.

3.2.1 Logarithmic scaling

Modeling on a logarithmic scale has several advantages: First, it allows the model to represent values accurately over the whole parameter range. This is essential when modeling simulation behavior, which often covers a large range of scales as seen in our example from Sect. 4, where we consider the numerical error and run time depending on discretization resolutions. Second, the additive correction term from the local model accounts for corrections and noise relative to the global model. Applying the corrections on the logarithmic scale implies, that deviations from the local model scale proportionally to the magnitude of the modeled function f . Therefore, further adjustments to the Gaussian process to account for the local order of magnitude of data values are often not necessary and using a stationary kernel for the Gaussian process is sufficient to capture scale-dependent corrections and noise. Third, the model output, and thus the predictive distribution, on the non-logarithmic original scale is always positive. Therefore, the model is well-suited for simulation quantities such as simulation errors or run times that are known to be positive.

Intuitively, these advantages are also the reason why convergence results or run time scaling plots are usually shown and discussed on a logarithmic scale: It is often much easier for analysts to derive conclusions on the polynomial degree of trends from such a representation.

3.2.2 Global model

In the global model, the parameters β_j of the linear combination are unknown and have to be learned by the Bayesian model. Because the modeled quantity is known to be positive, the coefficients β_j of the global model are restricted to a set

$$B \subseteq \left\{ \boldsymbol{\beta} \in \mathbb{R}^q \mid \forall_{x \in \Omega} \sum_{j=1}^q \beta_j \varphi_j(x) > 0 \right\}.$$

This ensures that the global model is positive for all evaluation points and, thus, the logarithm in the model is well-defined. Knowing that the coefficients represent individual error components or run time scaling, choosing $\beta_j > 0$ for $j = 1, \dots, q$ is often sufficient.

Using a linear combination of basis functions in the global model is sufficient for our experiments in Sect. 5. Generally, there is much flexibility when choosing a suitable global model and other parameterized functions can be used to describe the global trends.

3.2.3 Local model

The local model consists of a zero-mean Gaussian process with kernel function k . As kernel we suggest to use standard radial basis functions, such as a Gaussian or Matérn kernels [34], to model corrections to be locally correlated in the parameter space. This means that the correlation of two points depends on their distance relative to the length scale of the kernel. For parameters such as discretization widths, that are typically varied on a logarithmic scale, e.g., as powers of $\frac{1}{2}$, we suggest using a kernel that operates on this scale as well. For this purpose, we transform the kernel k to a kernel on the logarithmic scale \tilde{k} by transforming its inputs: $\tilde{k}(x, x') := k(\ln x, \ln x')$.

3.3 Posterior and predictive distribution

A drawback of modelling on the logarithmic scale is that the model becomes nonlinear in the unknown coefficients β_j and, therefore, the model prediction and its stochastic quantities such as mean value and standard deviation cannot be computed analytically. In the following, we shortly outline the calculation of the posterior distribution and the model prediction.

Our Bayesian approach computes the posterior distribution $p(\boldsymbol{\beta} \mid \mathbf{y}, \mathbf{x})$ of the model coefficients $\boldsymbol{\beta} = (\beta_j)_{j=1}^q$ based on the evaluated data $\mathbf{y} = (\ln f(x_i))_{i=1}^N$ at $\mathbf{x} = (x_i)_{i=1}^N$ and the Bayesian theorem:

$$p(\boldsymbol{\beta} \mid \mathbf{y}, \mathbf{x}) = \frac{p(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{x}) p(\boldsymbol{\beta})}{p(\mathbf{y} \mid \mathbf{x})}$$

The posterior is computed using the prior $p(\boldsymbol{\beta})$ for the coefficients and the likelihood $p(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{x})$ of observing the data given the coefficients $\boldsymbol{\beta}$. The term $p(\mathbf{y} \mid \mathbf{x})$ does not depend on the coefficients $\boldsymbol{\beta}$ and is therefore a constant scaling factor. In Sect. 3.4, we go more into detail on how the prior distribution for $\boldsymbol{\beta}$ is chosen.

The likelihood $p(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{x})$ of the data is implied by the local model. Given fixed values for $\boldsymbol{\beta}$, the measured values are evaluations of a realization of a Gaussian process with mean μ and kernel k

$$\ln \hat{f} \sim \mathcal{GP}(\mu, k) \quad (2)$$

because the observations are the sum of the global and the local model as given in (1), i.e., $\ln \hat{f}(x) = \mu(x) + s(x)$. For a finite number of observations \mathbf{y} , the function values are distributed according to a normal distribution

$$\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \Sigma)$$

with mean $\mathbf{m} = (\mu(x_i))_{i=1}^N$ and the covariance matrix $\Sigma = (k(x_i, x_j))_{i,j=1}^N$. In other words, the likelihood $p(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{x})$ is the density of the normal distribution $\mathcal{N}(\mathbf{m}, \Sigma)$, where the global model serves as a parameterized mean function and the local model defines the covariance.

To predict the simulation behavior at unknown evaluation points, the predictive distribution

$$p(\ln \hat{f}(x) \mid \mathbf{y}, \mathbf{x}) = \int_{\mathbb{R}^q} p(\ln \hat{f}(x) \mid \boldsymbol{\beta}, \mathbf{y}, \mathbf{x}) p(\boldsymbol{\beta} \mid \mathbf{y}, \mathbf{x}) d\boldsymbol{\beta} \quad (3)$$

is computed using the posterior distribution $p(\boldsymbol{\beta} \mid \mathbf{y}, \mathbf{x})$ of the coefficients and the likelihood $p(\ln \hat{f}(x) \mid \boldsymbol{\beta}, \mathbf{y}, \mathbf{x})$ as given in (2).

The global model allows the model to extrapolate reliably, while the local model ensures that locally correlated deviations from the trend are representable in the model. The predictive distribution also allows quantifying the model prediction and uncertainties for the predicted simulation behavior by computing expected values, standard deviations or quantiles of the predicted values.

Due to the nonlinearity of our model caused by the logarithmic scaling, the predictive distribution (3) cannot be computed analytically. Therefore, we approximate the posterior distribution of the coefficients β_j with Markov chain Monte Carlo sampling based on the prior distribution $p(\boldsymbol{\beta})$ and the likelihood $p(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{x})$. In particular, we use Hamiltonian Monte Carlo sampling with Stan [40]. The resulting samples are then used to approximate the posterior predictive

distribution (3) as

$$p(\ln \hat{f}(x) | \mathbf{y}, \mathbf{x}) \approx \frac{1}{|S|} \sum_{\beta \in S} p(\ln \hat{f}(x) | \beta, \mathbf{y}, \mathbf{x}),$$

where S denotes the set of posterior samples, and $|S|$ the number of samples. Importantly, sampling from the posterior and the predictive distribution is comparably cheap as we do not run new simulations.

For a more detailed introduction to Bayesian modeling, we refer to [18].

3.4 Prior for the global model coefficients

By introducing a parameterized global model μ , we have to specify a prior for the coefficients β_j in our Bayesian setting. When the order of magnitude of these coefficients is unknown, we want to apply a weakly informative prior that admits a large range of values.

However, in our case applying a weakly informative prior directly to the coefficients β_j leads to unintended results: The logarithm in the global model implies a prior density for the global model that is skewed to larger function values on a logarithmic scale. Therefore, the prior has a spurious bias towards large values for $\mu(x)$, which hinders the extrapolation properties of the global model. We illustrate this effect experimentally in Sect. 5.1.2.

Therefore, it is important on which scale we want the prior to be weakly informative. As we model the unknown functions on a logarithmic scale, we suggest applying a weakly informative prior also on the logarithmic scale, i.e., to the logarithm of the coefficients $\ln \beta_j$.

Applying the prior to β_j on the logarithmic scale leads to the desired property that the predictive distribution $p(\ln \hat{f}(x) | \mathbf{y}, \mathbf{x})$, as given in (3), becomes nearly invariant to scaling the data $\mathbf{f} = (f(x_i))_{i=1}^N$ and the coefficients β_j by a factor $b > 0$ simultaneously.

For this, we first show that the likelihood $p(\mathbf{y} | \beta)$ is invariant to this scaling. With given coefficients β , the likelihood of the observed data \mathbf{y} is a normal distribution with the global model as mean and the covariance given by the Gaussian process in the local model as discussed in Sect. 3.3. Using $\mathbf{y} = \ln \mathbf{f}$ and the definition of the global model μ for given coefficients β , we can write the likelihood for the scaled data and coefficients as

$$\begin{aligned} p(\ln(b\mathbf{f}) | b\beta, \mathbf{x}) &= \mathcal{N}(\ln(b\mathbf{f}) | \mu(\mathbf{x} | b\beta), \Sigma) \\ &= \mathcal{N}(\ln \mathbf{f} + \ln b | \mu(\mathbf{x} | \beta) + \ln b, \Sigma) \\ &= \mathcal{N}(\ln \mathbf{f} | \mu(\mathbf{x} | \beta), \Sigma) \\ &= p(\ln \mathbf{f} | \beta, \mathbf{x}) \end{aligned}$$

where $\mathcal{N}(\mathbf{y} | \mathbf{m}, \Sigma)$ denotes the probability density of the normal distribution $\mathcal{N}(\mathbf{m}, \Sigma)$ evaluated at \mathbf{y} , and $\mu(\mathbf{x} | \beta)$ is the global model for given coefficients β evaluated at the data points \mathbf{x} .

Having shown that the likelihood is scaling invariant, the remaining term affecting the shape of the predictive distribution is the prior $p(\beta)$. Therefore, we want to use a prior, that results in an approximately scaling invariant predictive distribution. This can be achieved by a weakly informative prior on the logarithmic coefficients. To show this, we write the predictive distribution in terms of $\tilde{\beta} := \ln \beta$ and perform a variable substitution to $\tilde{\beta}_b := \ln(b\beta) = \ln \beta + \ln b$ to rescale the coefficients:

$$\begin{aligned} p(\ln(\hat{f}(x)) | \ln \mathbf{f}, \mathbf{x}) &= \int_{\mathbb{R}^q} p(\ln(\hat{f}(x)) | e^{\tilde{\beta}}, \ln \mathbf{f}, \mathbf{x}) \\ &\quad \cdot \frac{p(\ln \mathbf{f} | e^{\tilde{\beta}}, \mathbf{x}) p(\tilde{\beta})}{p(\ln \mathbf{f} | \mathbf{x})} d\tilde{\beta} \\ &= \int_{\mathbb{R}^q} p(\ln(\hat{f}(x)) | b^{-1}e^{\tilde{\beta}_b}, \ln \mathbf{f}, \mathbf{x}) \\ &\quad \cdot \frac{p(\ln \mathbf{f} | b^{-1}e^{\tilde{\beta}_b}, \mathbf{x}) p(\tilde{\beta}_b - \ln b)}{p(\ln \mathbf{f} | \mathbf{x})} d\tilde{\beta}_b. \end{aligned}$$

This substitution usually leads to a Jacobi determinant, that alters the prior distribution for the new variables. However, because the scaling invariance corresponds to a translation on the logarithmic scale, the Jacobi determinant becomes 1 for the logarithmic parameterization and does not perturb the prior. Using the scaling invariance of the likelihood, the terms can be rewritten as

$$\begin{aligned} &= \int_{\mathbb{R}^q} p(\ln(\hat{f}(x)) | e^{\tilde{\beta}_b}, \ln \mathbf{f}, \mathbf{x}) \\ &\quad \cdot \frac{p(\ln(b\mathbf{f}) | e^{\tilde{\beta}_b}, \mathbf{x}) p(\tilde{\beta}_b - \ln b)}{p(\ln \mathbf{f} | \mathbf{x})} d\tilde{\beta}_b. \end{aligned}$$

If the prior density is rather flat over a large range of the parameter space, the prior density $p(\tilde{\beta}_b - \ln b)$ is similar to $p(\tilde{\beta}_b)$. Applying a similar argument to the marginal likelihood that serves as a normalization factor, shows the approximate scaling invariance:

$$\begin{aligned} &\approx \int_{\mathbb{R}^q} p(\ln(b\hat{f}(x)) | e^{\tilde{\beta}_b}, \ln \mathbf{f}, \mathbf{x}) \\ &\quad \cdot \frac{p(\ln(b\mathbf{f}) | e^{\tilde{\beta}_b}, \mathbf{x}) p(\tilde{\beta}_b)}{p(\ln(b\mathbf{f}) | \mathbf{x})} d\tilde{\beta}_b \\ &= p(\ln(b\hat{f}(x)) | \ln(b\mathbf{f}), \mathbf{x}). \end{aligned}$$

Therefore, we use a weakly informative prior on the parameters $\ln \beta_j$ on the logarithmic scale to achieve an approxi-

mately scaling invariant prediction. An example for such a prior is a normal distribution with a large variance. A similar argument is made in [7] to get an approximately invariant posterior distribution from a data-translated likelihood by choosing a suitable scale for the prior.

3.5 Acquisition function

New evaluation points for the optimization are chosen by maximizing the acquisition function. The acquisition function uses the predictive distribution to compute how promising possible evaluation points are for the optimization. As the focus of this work lies on the general modeling aspects for the optimization, we do not discuss the broad spectrum of acquisition functions that can be used. In our experiments, we use a constrained version of the expected improvement acquisition function [16, 17]

$$a_{EI}(x) = \mathbb{E}(\max(\ln f_N^* - \ln \hat{f}(x), 0) \mid \mathbf{x}, \mathbf{y}) \cdot \mathbb{P}(\hat{g}(x) \leq g_{\max} \mid \mathbf{x}, \mathbf{y}) \quad (4)$$

where

$$f_N^* = \min_{\substack{i=1, \dots, N \\ g(x_i) \leq g_{\max}}} f(x_i)$$

is the smallest feasible value evaluated so far. The acquisition function consists of two terms that are computed over the predictive distributions (3) of $\hat{f}(x)$ and $\hat{g}(x)$, accounting for the minimization goal of the objective and the constraint in the optimization problem. The first term is the unconstrained expected improvement function, that favors evaluation points, that improve the objective function over f_N^* , ignoring the feasibility of such a point. To make the acquisition function aware of the constraint, the second term $\mathbb{P}(\hat{g}(x) \leq g_{\max} \mid \mathbf{x}, \mathbf{y})$ is the probability that an evaluation point is feasible, to ensure that the optimizer stays close to the feasible region. Evaluations which violate the constraint $\hat{g}(x) \leq g_{\max}$ with a high probability are penalized with a low probability. Analogously to our modeling approach, we apply the acquisition function on the logarithmic scale, to measure the improvement over the current minimal value f_N^* on a logarithmic scale.

4 Test application

To evaluate our modeling approach on an exemplary test application, we use the multi-physics multi-scale biomechanics simulation framework OpenDiHu¹ [29, 30]. Typical

simulations of muscle activation starting with the signal coming from the spinal cord and leading to the actual contraction or release of a skeletal muscle comprise the following steps: On the subcellular level, the signal triggers a range of bio-electrical-chemical reactions, resulting in the generation of a transmembrane voltage. On the meso-scale, this transmembrane voltage leads to a traveling activation signal on the individual muscle fibers. A muscle consists of many thousands of muscle fibers and the activation signals control the contraction of each muscle fiber. On the macro-scale, i.e., the scale of the whole muscle, this contraction is modeled by classical 3D mechanics. Obviously, the three components each have their own temporal and spatial scales. The components are coupled through homogenization and are implemented in separate software subcomponents. Since a single muscle can comprise up to several hundred thousand individual fibers [28, 30], the whole simulation is only feasible in a massively parallel environment.

For simplicity reasons, we limit ourselves to a setup with only two scales in this paper: We simulate the generation of an electrical activation signal and its traveling along a single muscle fiber. While this is only a subset of the full simulation and essentially removes parallelism from the picture, it already demonstrates many typical challenges that arise in simulations when multiple numerical components are combined: The simulation consists of two coupled sub-models, each having an individual time discretization. Because numerical errors in the models influence each other, an optimal discretization for each model can not be computed a priori, and classical adaptivity on each scale cannot exploit its full potential.

4.1 Model equations and discretization

In detail, the simulation of a muscle fiber consists of two coupled models that build a reaction–diffusion system:

$$\partial_t u = \frac{\sigma}{AC} \partial_x^2 u - \frac{1}{C} I_{\text{ion}}(u, \mathbf{z}) \quad (5)$$

$$\partial_t \mathbf{z} = \mathbf{Z}(u, \mathbf{z}) \quad (6)$$

The bidomain Eq. (5) models the homogenized transmembrane voltage u along a one-dimensional muscle fiber. The subcellular model (6) describes how cells along the muscle fiber react to the voltage, which results in a transmembrane current I_{ion} . The surface-to-volume ratio A , the electrical capacitance C of the membrane and the conductivity of the fiber σ are assumed to be constant for each fiber and are listed in Table 1. The subcellular model is described by the Hodgkin–Huxley model [22] and has additional state variables \mathbf{z} which describe, for example, ion concentrations in the cells. The system (5) and (6) is discretized in space with a one-dimensional equidistant grid and linear Lagrange finite

¹ <https://github.com/maierbn/opendihu>.

Table 1 Physical parameters for the bidomain equation (5) in the OpenDiHu simulation. The length of the simulated fiber is l_{fiber}

σ [mS cm ⁻¹]	A [cm ⁻¹]	C [μF cm ⁻²]	l_{fiber} [cm]
3.828	500	0.58	11.9

elements for the diffusion equation. The reaction equations (6) are solved at each grid point.

In the time dimension, the system of equations is decoupled by a second order Strang operator splitting [41], which allows to use an individual time stepping scheme for each subproblem. Each splitting step of width h_s consists of three steps that are performed on each interval $]t_n, t_n + h_s]$:

1. First, we solve the reaction equations

$$\begin{aligned} \partial_t u &= -\frac{1}{C} I_{\text{ion}}(u, \mathbf{z}) \\ \partial_t \mathbf{z} &= Z(u, \mathbf{z}) \end{aligned} \tag{7}$$

numerically on the interval $]t_n, t_n + \frac{h_s}{2}]$ with \mathbf{z} and u from time t_n as initial values.

2. Second, we solve the diffusion equation

$$\partial_t u = \frac{\sigma}{AC} \partial_x^2 u \tag{8}$$

numerically on the interval $]t_n, t_n + h_s]$ using the values of u at time $t_n + \frac{h_s}{2}$ from the previous step. The state \mathbf{z} of the subcellular model remains unchanged.

3. Third, we solve the reaction equations (7) again, this time on the interval $]t_n + \frac{h_s}{2}, t_n + h_s]$ with \mathbf{z} and u from time $t_{n+1} = t_n + h_s$ from the previous step as initial values.

Because the linear diffusion equation (8) is a stiff problem, it is solved with an implicit Euler method or a Crank–Nicolson method with step width h_1 . The nonlinear reaction term (7) is a system of nonlinear ordinary differential equations and solved with Heun’s method with step width $\frac{h_0}{2}$ in phase 1 and 3 of each splitting step. The time step width h_s of the Strang splitting is set to the maximum of h_0 and h_1 to avoid the splitting error to become larger than necessary.

4.2 Optimization problem

The optimization goal is to find time step widths h_0 and h_1 for the subproblems, such that we minimize the simulation error within an upper limit for the simulation run time. This means, we solve

$$\operatorname{argmin}_{(h_0, h_1) \in \Omega} f(h_0, h_1) \quad \text{s.t. } g(h_0, h_1) \leq g_{\text{max}} \tag{9}$$

for a given run time limit g_{max} . We choose

$$\Omega = \{2^{-i} \times 4 \times 10^{-3} \text{ ms} \mid i = 0, \dots, 14\}^2$$

as the discrete admissible set for the time step widths h_0 and h_1 to ensure that the time step width of each solver divides the splitting time step width $h_s = \max(h_0, h_1)$.

In space, we discretize the muscle fiber with 1190 linear Lagrange elements. To ensure that only the discretization errors dominate the final result, we solve the linear system that arises from the diffusion equation (8) with a conjugate gradient solver with a tight relative residual tolerance of 10^{-14} .

For each evaluation, we simulate 4×10^{-2} ms of physical time starting from an initial activation wave profile. As our example application is part of a larger simulation example in OpenDiHu that is coupled to the transmembrane voltage, we use the L^2 -error of the u variable at the final time 4×10^{-2} ms for the objective function. The error is estimated by computing the L^2 -difference against a very fine reference simulation with a time discretization of $h_0 = h_1 = 2^{-17} \times 4 \times 10^{-3}$ ms.

4.3 Error and run time modeling

To build individual models for the error f and the run time g in the optimization problem (9), we use the modeling approach from Sect. 3:

$$\begin{aligned} \ln \hat{f}(h_0, h_1) &= \mu_f(h_0, h_1) + s_f(h_0, h_1) \\ \ln \hat{g}(h_0, h_1) &= \mu_g(h_0, h_1) + s_g(h_0, h_1) \end{aligned}$$

It combines global model components μ_f and μ_g with local deviation models s_f and s_g on a logarithmic scale.

To build the global models μ_f and μ_g , we use a priori knowledge directly stemming from a priori asymptotic convergence results and known asymptotic run time scaling. The convergence order of the simulation error depends on the specific solver combination. The Strang splitting, Heun’s method and the Crank–Nicolson method are second order accurate, whereas the implicit Euler method is first order accurate. This leads to an ansatz of multivariate polynomials in the two step widths. The model

$$\mu_f^{\text{IE}}(h_0, h_1) = \ln(c_c + c_0 h_0^2 + c_1 h_1 + c_s \max(h_0, h_1)^2)$$

constitutes the global model for the error when using the implicit Euler scheme for the diffusion problem (8) and

$$\mu_f^{\text{CN}}(h_0, h_1) = \ln(c_c + c_0 h_0^2 + c_1 h_1^2 + c_s \max(h_0, h_1)^2)$$

is applied if we use the second order accurate Crank–Nicolson scheme instead. In both cases, the term c_c represents the fact that the simulation error will not become arbitrarily

small by reducing the time step width alone, because other error sources, e.g., induced by floating point arithmetic and inaccuracies in the linear solver, also affect the outcome. c_s is the splitting error. For each time stepping scheme, a smaller time step width results in more time steps to be computed while the cost of the splitting itself is negligible. Therefore, we use

$$\mu_g(h_0, h_1) = \ln(d_c + d_0 h_0^{-1} + d_1 h_1^{-1})$$

for the global model of the run time constraint in both settings. The coefficient d_c represents constant run time parts such as set up and initialization times. For all global models, we restrict the coefficients to be positive.

The local model is based on a Gaussian process with the kernel

$$k(x, x') = \alpha^2 \exp\left(-\frac{\|x - x'\|^2}{2\rho^2}\right) + \sigma^2 \delta_{x, x'},$$

which consists of two terms. The first term models local correlation using a Gaussian kernel with variance α^2 and length scale ρ . With the second term with the Kronecker delta δ , we add additional pointwise noise with standard deviation σ to account for pointwise, uncorrelated deviations from the global model that cannot be explained by the Gaussian kernel. Such pointwise deviations are caused by numerical inaccuracies and fluctuations caused by the many layers of complex computations in the simulation that affect the simulation error and run time and make the optimization functions non-smooth. Not taking these effects into account can cause instabilities in the modeling process.

4.4 Simplified one-dimensional analytical test case

To validate our modelling approach in the next section, we construct a simpler one-dimensional example function, where the ground truth is known, but with structural similarity to the OpenDiHu example. More specifically, we assume that the simple one-dimensional function has a global trend consisting of a constant and a quadratic term. The constant corresponds to underlying offsets, the quadratic term to the convergence order in the real example. A local deviation from this trend is represented by a squared exponential function such that the objective function reads:

$$f(h) = 0.1 + 1.5h^2(1 - 0.8e^{-(2.2-h)^2}) \quad (10)$$

The coefficients are chosen to be on different scales so that the effect of the prior is better visible. The constants in the squared exponential functions are chosen such that the local deviation is clearly visible in the experimental results.

The a priori knowledge for the model consists of the basis functions $\varphi_1(h) = 1$ and $\varphi_2(h) = h^2$ and, thus, the global model reads

$$\mu_f(h) = \ln(c_1 + c_2 \cdot h^2)$$

with $c_1, c_2 > 0$. The local model uses the same Gaussian kernel as the test application in Sect. 4.3.

5 Experiments

To evaluate our modeling approach, we split our experiments in two parts. We start with validating the model in Sect. 5.1 by showing how the log scale, the global model and the prior affect the model predictions. In Sect. 5.2, we then show how the model can be used inside a Bayesian optimization approach.

5.1 Model validation

To examine the effect of the model and the prior, we employ two examples: First, the simple one-dimensional example function from Sect. 4.4, and second, the OpenDiHu example we presented in Sect. 4.

In the following, we use the simple one-dimensional example function to show the usefulness of the a priori knowledge in the global model and the logarithmic scale in Sect. 5.1.1. In Sect. 5.1.2 we show the effect of the prior on the model quality, followed by an evaluation of the overall model quality for OpenDiHu in Sect. 5.1.3.

In both setups, the structure of the deviations from the global trend is assumed to be known for simplicity reasons. The variables α , ρ , and σ in the local model are selected a priori and are not subject to the Bayesian inference.

5.1.1 Influence of the global model and the logarithmic scale

In this first experiment, we show the effect of including a global model representing our a priori knowledge on the global trend and of modeling on a logarithmic scale. In particular, we show that our model is able to represent the global trend, that the predicted values are positive, and that the magnitude of the model uncertainties is relative to the magnitude of the global trend. For our experiments, we use the simple one-dimensional function defined in (10) and five given evaluation points for the parameters h chosen equidistantly on the logarithmic scale between $\ln(0.25)$ and $\ln(4)$. The respective known data expose the quadratic trend clearly, while the constant term is more difficult to deduce from the evaluated data.

Table 2 Priors and parameters used in the model validation experiments with the one-dimensional problem (10) in Sect. 5.1.1 and Sect. 5.1.2

Model	Global model prior on c_1 and c_2	Local model			HMC samples
		ϱ	α^2	σ^2	
Gaussian		0.25	0.1	0	Analytic
Linear	$c_1, c_2 \sim \text{Flat}$	0.25	0.1	0	Analytic
Logarithmic	$\ln c_1, \ln c_2 \sim \text{Student-t}(4, 0, 7)$	0.25	0.5	0	8000
Logarithmic	$c_1, c_2 \sim \text{Student-t}(4, 0, 30)$	0.25	0.5–8	0	8000
Logarithmic	$\ln c_1, \ln c_2 \sim \text{Student-t}(4, 0, 7)$	0.25	0.5–8	0	8000

We compare the model results of three different modeling approaches using the parameters as listed in Table 2:

Gaussian process The Gaussian process model acts on the original, linear scale and completely omits a global model component:

$$\hat{f}(h) = s(h)$$

The model uses the radial basis function from Sect. 4.3 as kernel. The predictive distribution can be computed analytically for this model.

Linear The linear model extends the Gaussian process model by using additional basis functions to expand the ansatz space of the Gaussian process regression as in [34], i.e., it is also on a linear scale, but includes a global model:

$$\hat{f}(h) = c_1 + c_2 \cdot h^2 + s(h)$$

Also for this model, the predictive distribution can be computed analytically.

Logarithmic The logarithmic model represents the complete model on the logarithmic scale as introduced above in Sect. 4.4. Due to the nonlinearity induced by the logarithmic scaling, the prediction of this model can not be computed analytically and requires sampling.

To evaluate the quality of the model predictions, we compare the model predictions as the predicted mean and the model uncertainties as the 50% confidence interval between the 0.25 and 0.75 quantiles of the predictive distribution for all three models in Fig. 1.

The model consisting only of a Gaussian process is unable to represent the global trend because the mean quickly decays to zero further away from the evaluated data points. Further, the predicted values are allowed to be negative which contradicts the knowledge that errors and runtimes have positive values. This also causes the lower quantiles to quickly decay to zero and even reach into negative values.

Adding the prior knowledge linearly as additional basis functions allows the linear model to capture the global trend reasonably well. However, the confidence interval becomes relatively large for small predicted values, because the model is additive on a linear scale, which implies that the model

fits larger values more accurately in terms of relative error than it does fit smaller values. Further, the posterior distributions still permit negative values, which are not possible for quantities such as errors and run times that are known to be always positive. Alternative approaches that operate on a linear scale, for example scaling the kernel functions relative to the global model, at least also suffer from the latter drawback and increase the complexity as they require determining a suitable kernel scaling.

Both of these problems are resolved when the model is built on a logarithmic scale: The posterior distribution only permits positive values and the posterior uncertainty scales nicely relative to the mean of the prediction. In contrast to the previous models, the model uncertainties for the logarithmic model cannot reach negative values and stay positive and follow the trend. Note that the combination of a global and a local model is also able to approximate the local minimum as the local model approximates the local deviation from the global trend.

5.1.2 Effect of the prior on the model quality

In our second experiment, we assess how the scale on which we apply a weakly informative prior to the coefficients affects the extrapolation quality of our model. We use the same setup with the one-dimensional function (10) and compare the model predictions for models with two different priors on c_1 and c_2 for our logarithmic model from Sect. 4.4: First, we use a wide Student-t distribution as weakly informative prior directly for the coefficients:

$$c_1, c_2 \sim \text{Student-t}(4, 0, 30)$$

The arguments define the degrees of freedoms, the mean and the standard deviation of the distribution. Second, we use a weakly uninformative prior for the logarithms of the model coefficients:

$$\ln c_1, \ln c_2 \sim \text{Student-t}(4, 0, 7)$$

Due to the different scales, the distribution on the logarithmic scale uses a smaller variance.

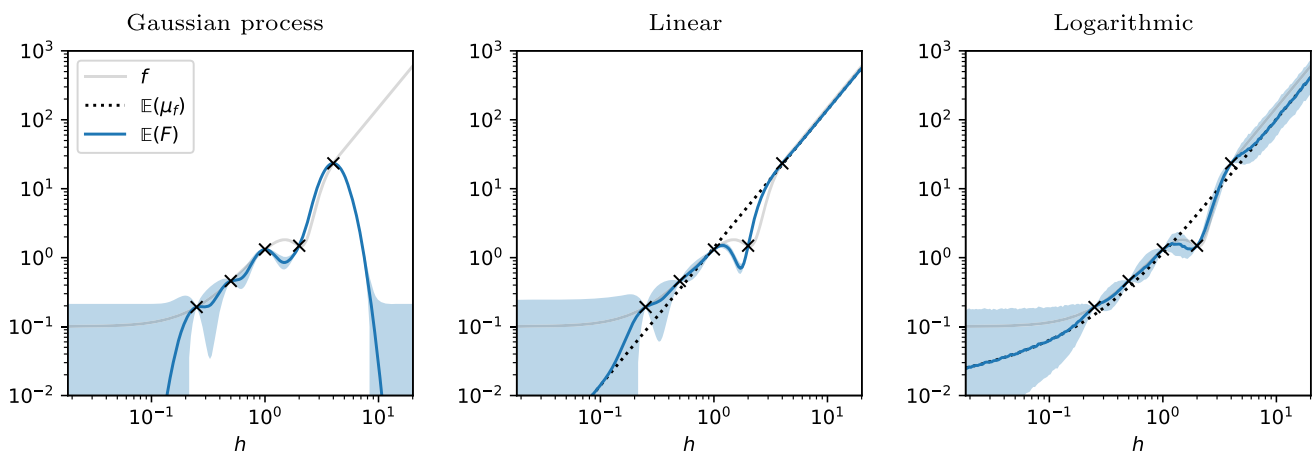


Fig. 1 Model validation: Verification of the usefulness of the global model and the logarithmic scale: Model prediction means $\mathbb{E}(\hat{f} | \mathbf{x}, \mathbf{y})$ (blue lines) and model uncertainties as the 50% confidence interval of the predictive distribution (blue shaded areas) for the 1d example (10) using a Gaussian process model, a linear model, and our logarithmic

model. The unknown function f is shown in gray with the 5 evaluation points marked by black crosses. The mean of the global model $\mathbb{E}(\mu_f | \mathbf{x}, \mathbf{y})$ is shown as dotted line. The Gaussian process model does not have a global model. (Color figure online)

In addition to the two prior choices, we vary the variance α^2 of the Gaussian process in the local model, which corresponds to assuming different degrees of deviations from the global model. Table 2 summarizes the used model parameters. For the resulting combinations of priors and local model variances, we compare the extrapolation quality of the resulting global model predictions.

Figure 2 shows how the prior affects the model prediction.

For small variances in the local model the scale on which the weakly informative prior on the global model coefficients is applied has only a small effect on the model and the overall model is constrained by the local model. However, when we assume more deviations from the global model, i.e., for larger variances, the influence of the data points decreases and the effect of the prior becomes more pronounced:

When using a weakly informative prior for the coefficients c_1 and c_2 , i.e. on the linear scale, the mean of the global model moves towards too large values for larger local model variances. Hence, the global model does not represent the trend of the data and the model is unable to predict meaningful values outside the range of known data points. This is caused by the nonlinearity in the global model as discussed above in Sect. 3.4: The weakly informative prior on c_1 and c_2 becomes more informative on the logarithmic scale as it implies a preference for larger values. This effect is visualized in Fig. 3, where the probability density of a random variable is transformed to the logarithmic scale.

Using a weakly informative prior for $\ln c_1$ and $\ln c_2$ avoids this problem, as it is directly applied on the correct scale. As a result, the global model stays closer to the data points also for larger prior variances in the Gaussian process. This also

leads to more meaningful predictions further away from the data.

5.1.3 Model quality for OpenDiHu

We now show that our modeling approach is also able to represent data from the two-dimensional OpenDiHu example from Sect. 4.2. For this we compute a maximum likelihood estimation (MLE) for the model parameters using data for all time step combinations in Ω and a simulated time of 4×10^{-2} ms. The MLE optimizes the coefficients β_j for the basis functions, and the prior variance α^2 , the length scale ϱ and the additional noise σ of the kernel. The parameters are optimized with a restarted L-BFGS method using the `optimize` capability of Stan starting from randomized initial values. The simulation error is approximated as the L^2 difference to a finer discretization as described in Sect. 4.2. To measure the run time, we run the simulation 4 times and take the minimum value. This is done to reduce the influence of fluctuations caused by caching effects because the simulation loads muscle geometry data from the hard drive and compiles the subcellular model at runtime. While simulating 4×10^{-2} ms reduces the computational cost, the time interval is not representative of a typical simulation, that simulates multiple milliseconds to seconds. As the simulation behavior does not change significantly over the simulated time, we scale the run time to a more realistic simulated time of 4 ms to reduce the influence of the constant overhead from starting the simulation and loading required input data. This is done by measuring the run time for the entire simulation and the time stepping schemes separately. The time spent in the time stepping schemes is then scaled by a factor of 100.

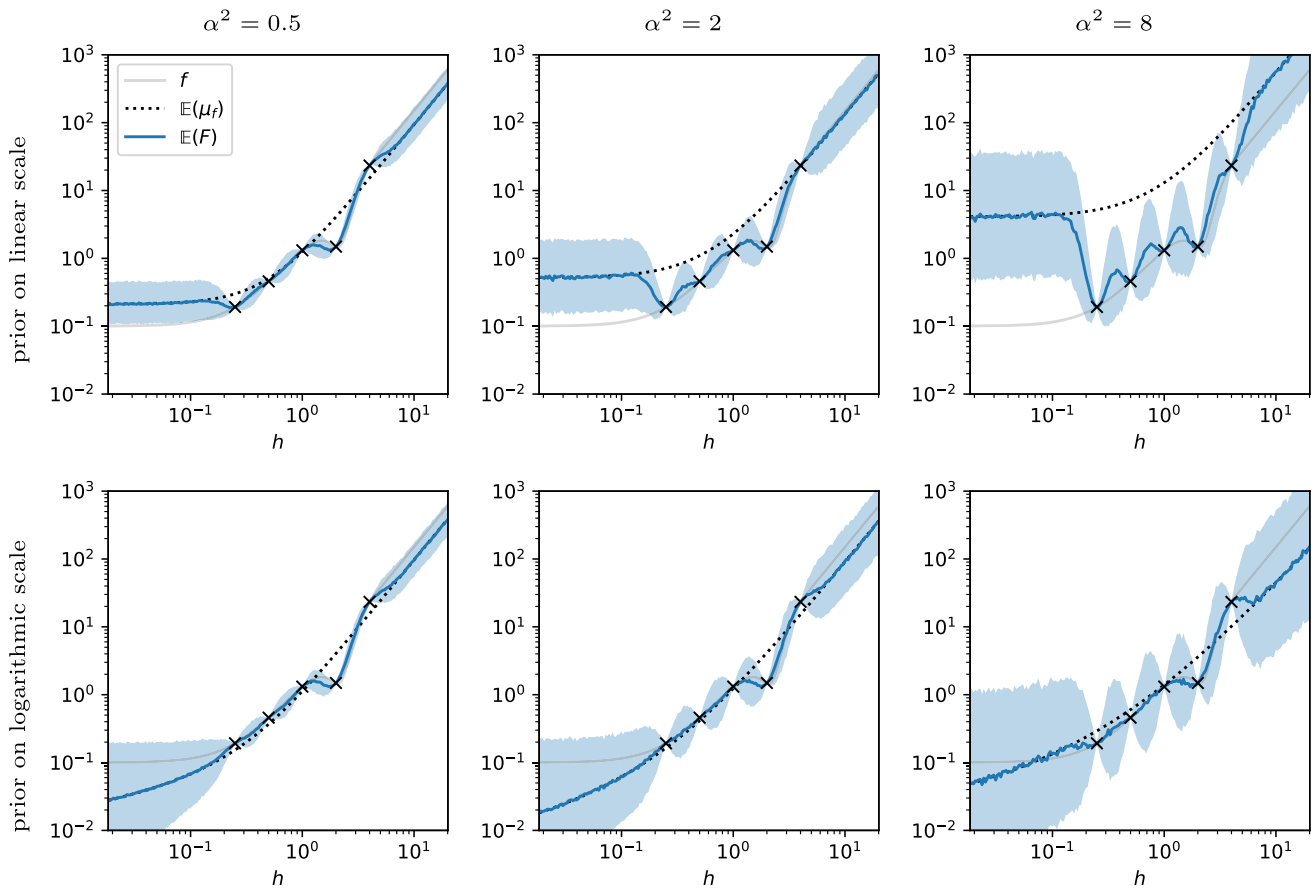


Fig. 2 Assessment of the effect of the prior on the model quality: Model prediction means $\mathbb{E}(\hat{f} \mid \mathbf{x}, \mathbf{y})$ (blue lines) and model uncertainties as the 50% confidence interval of the predictive distribution (blue shaded areas) for the 1d example (10) for different prior variances α^2 of the Gaussian process using a weakly informative prior for the coefficients

on either the linear scale (c_1, c_2) or logarithmic scale ($\ln c_1, \ln c_2$). The unknown function f is shown in gray with the 5 evaluation points x_i marked by black crosses. The mean of the global model $\mathbb{E}(\mu_f \mid \mathbf{x}, \mathbf{y})$ is shown as dotted line. (Color figure online)

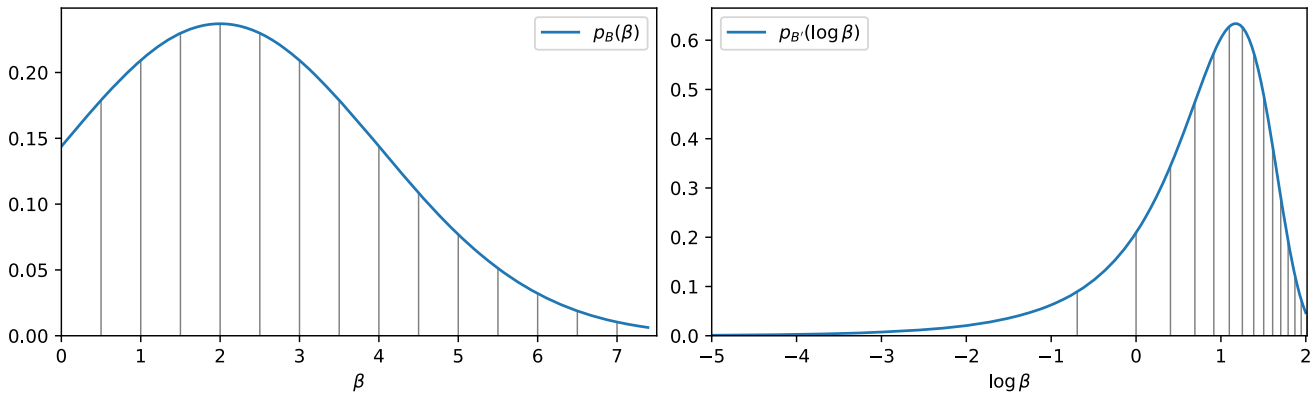


Fig. 3 Visualization of a prior distribution transformed to the logarithmic scale. Left: Probability density function p_B for a random variable B that is distributed following a truncated normal distribution with mean 2,

standard deviation 2 and a lower bound of 0. Right: Transformed probability density $p_{B'}$ for $B' = \ln B$. The transformed density is clearly skewed towards larger values

The estimated parameters are then used to visualize the individual model components. In particular, it is of interest, how well the global model captures the trends in the data, and how the local model makes adjustments to the model to match the data points.

Figure 4 shows the true data (first column) from the Crank–Nicolson and implicit Euler setup together with the model components resulting from the MLE parameters.

The results show, that the global model (second column) is able to capture the optimization function well and captures most features of the global behavior of the data. The locally correlated component of the local model (third column) makes smaller local adjustments to the model. The remaining pointwise noise (last column), caused by the σ term in the kernel, is relatively small.

The data for the Crank–Nicolson setup also show that the error decreases when both time step widths are of the same size, which is visible along the diagonal $h_0 = h_1$. This effect is caused by error cancellation between the splitting error and the integration errors of the subproblems. The error cancellation cannot be represented by the global model with only positive coefficients c_0 and c_1 and has therefore be represented in the local model. The largest adjustments from the local model are therefore concentrated along the diagonal where cancellation effects have stronger influence.

The error cancellation could be included in the global model by either allowing negative values for c_0 and c_1 , or by adding additional basis functions, that model cancellation effects in a more general way. As such strong error cancellation is a rather special property of our example problem and often not known a priori, we continue with the model with positive coefficients in the optimization experiments. In the next experiment we see that this model is suitable in an optimization loop.

5.2 Optimization results

Having seen that the model is able to represent the data, we now show how the model works in conjunction with a Bayesian optimization where data are scarce and expensive to obtain. For this we perform an optimization of the example application from Sect. 4 to minimize the simulation error under runtime constraints.

For the sake of simplicity, we assume that the general structure of the local model and the noise is known, by using the values of the MLE results from Sect. 5.1.3, rounded to two significant digits, as fixed values for the variance, length scale and noise of the kernel in the local model. Table 3 lists the used values and the priors used for the coefficients in the global model. We use the expected improvement acquisition function from Sect. 3.5 and perform up to 15 optimization steps. The optimization ends earlier if the acquisition function becomes smaller than 10^{-5} .

In a first experiment we use data from a setup with Heun’s method for the reaction term and Crank–Nicolson as implicit solver for the diffusion term. To reduce the computational cost, the data for the simulation error and run time are measured for a simulated time of 4×10^{-2} ms and scaled to a more realistic simulated time of 4 ms as described in Sect. 5.1.3.

Figure 5a shows optimization steps starting with four initial data points on coarse discretizations. We can see that the global trend is captured well from only few data points, even though the global model is unable to predict the details of the error cancellation along the diagonal. Nonetheless, the optimizer directly explores points close to the global optimum and finds the minimum within 4 steps. In this example, the optimizer is able to find the minimum so quickly, because the optimization functions can be approximated well by the global model alone and thus the deviations from the trend are small. This allows accurate extrapolation using only the global model.

In the next experiment we therefore use data, in which the global model alone is not able to represent the optimization functions. For this, we use data from the setup with the implicit Euler method for the diffusion term. Again, we scale the run time of the time stepping schemes to a simulated time of 4 ms. To increase the noise level, we decompose the full data into the global, local and noise components by computing a MLE. The local models of the error and run time are then scaled by a factor of 10 and the components are recombined to create new optimization functions which are then used in the optimization. Figure 5b shows, that the unknown functions are more complex and cannot be described by the global model alone. Therefore, it takes the optimizer more steps to approximate the optimization functions accurately. Even though the models are able to extrapolate further away from the data, it can be clearly seen, that it takes multiple steps until the model for the constraint function describes the constraint boundary well. The details in the constraint function can only be described by the local model once enough data close to the constrained boundary is available. This shows, that our model can also model more complex functions by combining the global and local model. We also see, that the optimizer evaluates more points further away from the constrained boundary. This is caused by the larger kernel magnitude α , meaning the model assumes more uncertainty in the local model, which leads to a wider predictive distribution.

6 Discussion

We have shown that Bayesian optimization in combination with a knowledge-based model ansatz for simulation error and run time can successfully be used to find optimal numer-

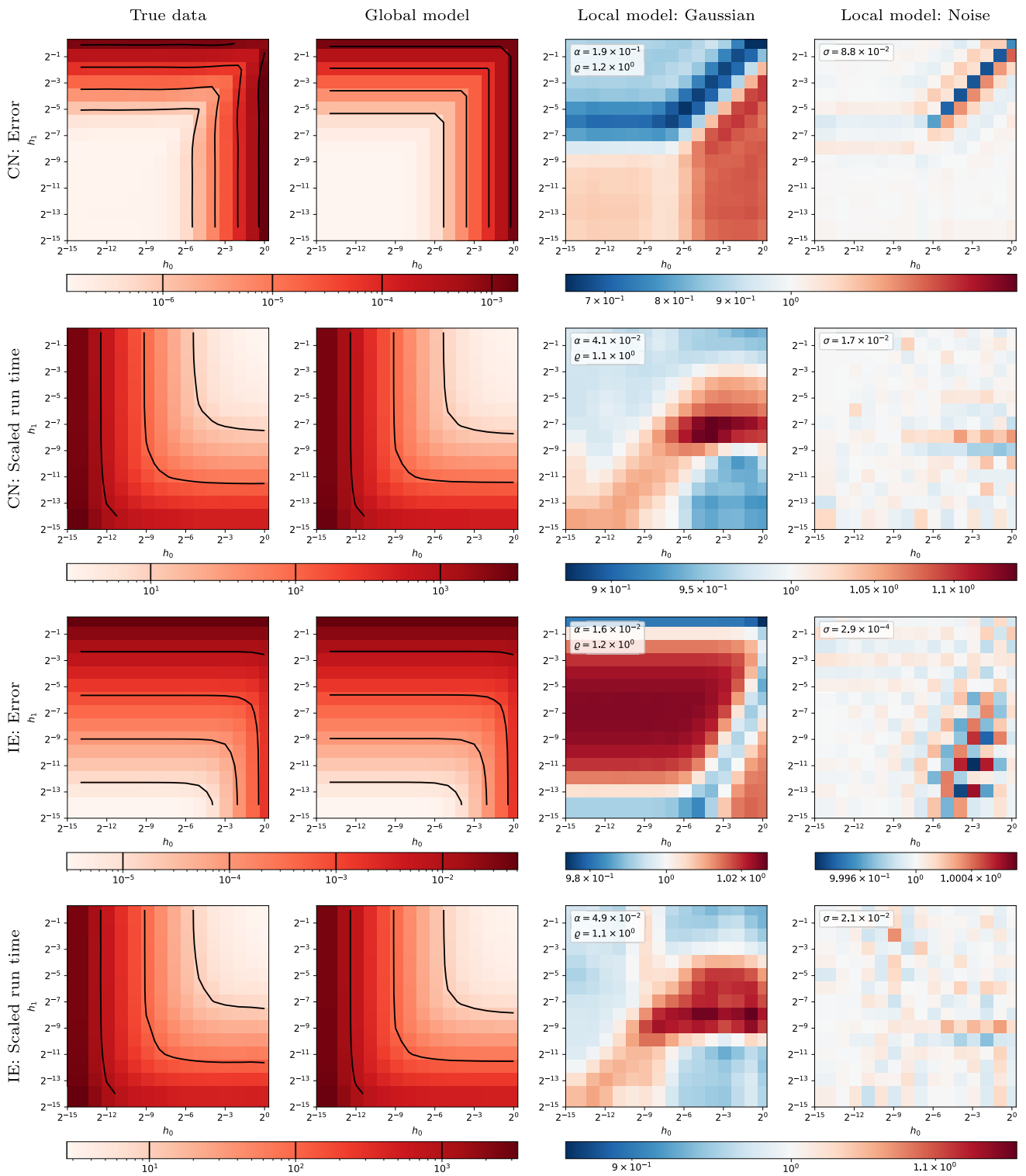


Fig. 4 Model validation on experimental data: Results of the maximum likelihood estimation of the model for the OpenDiHu example (9) for the setup with the Crank–Nicolson (CN) and implicit Euler (IE) method as solver for the diffusion term. The data for f and g (first column) is

well-represented by the respective global model $\exp(\mu_f)$ and $\exp(\mu_g)$ (second column) while the local model $\exp(s_f)$ and $\exp(s_g)$ with a correlated component (third column) and pointwise noise (last column) is able to represent local deviations

Table 3 Priors and model parameters used in the optimization experiments in Sect. 5.2 for the setup with the Crank–Nicolson (CN) and implicit Euler (IE) method as solver for the diffusion term. All models use 24,000 Monte Carlo samples

	logarithmic model	global model prior for coefficients	local model		
			ϱ	α^2	σ^2
CN error		$\ln c_j \sim \text{Student-t}(4, 0, 7)$	1.2	3.6×10^{-2}	7.7×10^{-3}
CN run time		$\ln d_j \sim \text{Student-t}(4, 0, 7)$	1.1	1.7×10^{-1}	2.9×10^{-4}
IE error		$\ln c_j \sim \text{Student-t}(4, 0, 7)$	1.2	2.6×10^0	8.4×10^{-8}
IE run time		$\ln d_j \sim \text{Student-t}(4, 0, 7)$	1.1	2.4×10^1	4.4×10^{-4}

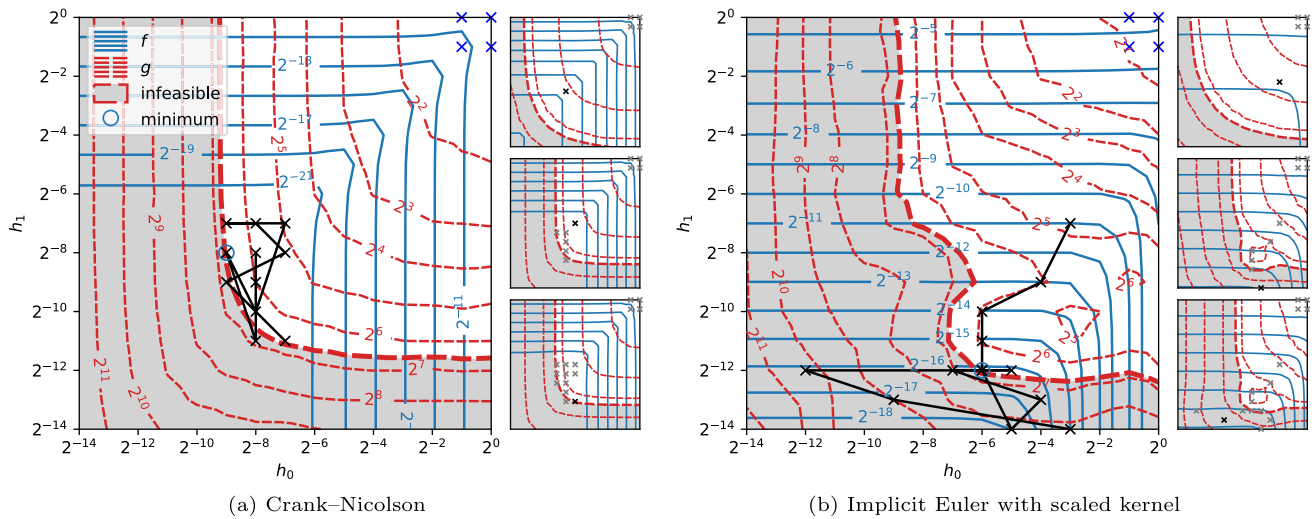


Fig. 5 Optimization steps for the OpenDiHu example using the expected improvement acquisition function (4). The four initial points on coarse discretizations are marked by blue crosses. Big plots: Objective f (blue) and constraint function g (dashed red, gray denoting the infeasible set). The true minimum is marked with a blue circle. Consecutive evaluation points are connected by straight lines to visualize

the evaluation order of the optimizer. Smaller plots: Expected values $\mathbb{E}(\hat{f} \mid \mathbf{x}, \mathbf{y})$ and $\mathbb{E}(\hat{g} \mid \mathbf{x}, \mathbf{y})$ of the models of optimization functions in the first, fifth and tenth step for the respective Crank–Nicolson and implicit Euler setup. The next evaluation point is marked in black. (Color figure online)

ical parameter combinations minimizing the global error under run time constraints. In contrast to classical adaptivity, our approach is non-invasive, i.e., requires no changes in the simulation code itself, requires only very few simulation runs to generate the required data, and is generalizable to other objective functions and constraints as well as to more than two parameters. Two key ingredients make it well-suited to model simulation behavior: A knowledge-based global model for global trends, and a local model that accounts for local deviations from the global model. By including prior knowledge such as convergence orders or run time scaling, the model can be used to make meaningful predictions further away from the data, which reduces the number of steps, and thus simulation runs, needed to find a good parameter combination. Modeling the data on a logarithmic scale ensures accurate predictions for values spanning multiple orders of magnitude, which is common for simulation errors and run times. Using a weakly informative prior for the coefficients of the global model on a logarithmic scale avoids an artificial bias in the model predictions.

In our numerical experiments, we have computed the simulation errors by comparing against a finer discretization

for simplicity reasons, which can be computationally expensive for many applications. In general, and in particular for more complex problems, the evaluation of the optimization functions can be non-trivial. While simulation-internal a posteriori error estimators can be more efficient to evaluate the simulation error they are not always available or come with additional implementation cost. Data such as run time measurements usually include measurement noise, which means that the optimization has the additional trade-off between precise measurements and the cost of running the simulation multiple times.

For the simulation components in the OpenDiHu application, good a priori knowledge is available. For more complex problems, this might be more difficult, but for most solver and discretization schemes, at least the asymptotic behavior is generally known, and can be used as a starting point. The quality of the model and the efficiency of the optimization depend on the prior knowledge that is available. Motivated by our example application, we used a linear combination of basis functions in the global model. However, an interesting extension of our approach is to investigate other forms of global trends: As we use Monte Carlo to sample from the

posterior distribution, the linear combination can easily be replaced by other parameterized functions and corresponding priors for the coefficients, if they describe the global trends more accurately. For instance, for components where the asymptotic convergence rate is not known a priori, such as linear and nonlinear solvers, one could model the convergence rate itself with an additional coefficient in the global model and a suitable prior.

In addition, the performance of the optimization depends also on the kernel parameters of the Gaussian process in the local model. Too large or too small variances and kernel widths that do not match the data, lead to a reduced quality of the data acquisition and consequently to excessive or insufficient exploration during the optimization and reduced model quality. If the values of the kernel parameters are not known, they can either be estimated in each step using the evaluated data, or they can be included into the Bayesian optimization to treat all parameters in a unified Bayesian setting without altering our method significantly.

An obvious next step in our research is to take a closer look at the acquisition function, in particular in terms of including the cost of data point evaluations which vary drastically with the chosen resolution. Further, the role of objective and constraint does not matter for our modeling approach. However, exchanging objective and constraint, i.e., optimizing run time under accuracy constraints, leads to a different relation between feasible parameters, and parameters that are cheap to evaluate. The feasible region, that is all simulation runs that result in errors below the given bound, would contain potentially very expensive data evaluations such that it is favorable to use an acquisition function that approaches the optimum from the cheaper infeasible region.

In general, our approach does not exclude adaptivity inside the simulations. Typical adaptive algorithms rely on parameters such as threshold values that control the accuracy of the model. Controlling these with our presented approach would allow an interesting combination of adaptive methods and external optimization for the remaining parameters that influence the simulation behavior. A simple example is the optimization of resolutions in time and space in an external Bayesian setting and an internal adaptive setting of stopping criteria for linear and non-linear solvers based on error estimators. The internal adaptivity would be part of the simulation implementation and, as such, be mirrored in the data used in the external Bayesian optimization.

An interesting additional benefit of using a model-based approach, is that after the optimization, the resulting model can be used for additional insight in the simulation behavior. For example, the model can be used to perform a rough sensitivity analysis of the found optimizer: For instance, one could estimate how much extra run time would be needed to reduce the error by a certain factor, or to see, which simulation component dominates the simulation error.

Acknowledgements Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 - 390740016. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech).

Author Contributions Conceptualization: FH; Data curation: FH; Formal analysis: FH, PCB, DG, MS; Funding acquisition: DG, MS; Investigation: FH; Methodology: FH, PCB; Project administration: FH, DG, MS; Software: FH; Supervision: DG, MS; Visualization: FH; Writing original-draft: FH, DG, MS; Writing - review & editing: FH, PCB, DG, MS.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The data for the OpenDiHu experiments are available at <https://doi.org/10.18419/darus-3550>.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Astudillo R, Frazier P (2019) Bayesian optimization of composite functions. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th International Conference on Machine Learning, vol 97. PMLR, pp 354–363
2. Astudillo R, Frazier PI (2021) Thinking inside the box: A tutorial on grey-box Bayesian optimization. In: Proceedings of the 2021 Winter Simulation Conference
3. Balandat M, Karrer B, Jiang DR et al (2020) BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. In: Larochelle H, Ranzato M, Hadsell R et al (eds) Advances in neural information processing systems, vol 33. Curran Associates, New York, pp 21524–21538
4. Bangerth W, Rannacher R (2003) Adaptive finite element methods for differential equations. Lectures in Mathematics. Birkhäuser, Chan. <https://doi.org/10.1007/978-3-0348-7605-6>
5. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. *J Mach Learn Res* 13(Feb):281–305
6. Bergstra JS, Bardenet R, Bengio Y et al (2011) Algorithms for hyper-parameter optimization. In: Shawe-Taylor J, Zemel RS, Bartlett PL et al (eds) Advances in neural information processing systems, vol 24. Curran Associates, New York, pp 2546–2554
7. Box GEP, Tiao GC (1992) Bayesian inference in statistical analysis. Wiley, chap 1:3

8. Bradley CP, Emamy N, Ertl T et al (2018) Enabling detailed, biophysics-based skeletal muscle models on HPC systems. *Front Physiol* 9:816. <https://doi.org/10.3389/fphys.2018.00816>
9. Bungartz HJ, Griebel M (2004) Sparse grids. *Acta Numer* 13:147–269. <https://doi.org/10.1017/S0962492904000182>
10. Choi JW, Bedard D, Fowler R, et al (2013) A roofline model of energy. In: 27th International parallel and distributed processing symposium (IPDPS 2013), pp 661–672. <https://doi.org/10.1109/IPDPS.2013.77>
11. Ern A, Vohralík M (2013) Adaptive inexact Newton methods with a posteriori stopping criteria for nonlinear diffusion PDEs. *SIAM J Sci Comput* 35(4):A1761–A1791. <https://doi.org/10.1137/120896918>
12. Estep D, Ginting V, Ropp D et al (2008) An a posteriori-a priori analysis of multiscale operator splitting. *SIAM J Numer Anal* 46(3):1116–1146. <https://doi.org/10.1137/07068237X>
13. Feurer M, Klein A, Eggenberger K et al (2015) Efficient and robust automated machine learning. In: Cortes C, Lawrence N, Lee D et al (eds) *Advances in neural information processing systems*, vol 28. Curran Associates, New York
14. Frazier P, Powell W, Dayanik S (2009) The knowledge-gradient policy for correlated normal beliefs. *INFORMS J Comput* 21(4):599–613. <https://doi.org/10.1287/ijoc.1080.0314>
15. Frazier PI (2018) A tutorial on Bayesian optimization. arxiv
16. Gardner JR, Kusner MJ, Xu ZE, et al (2014) Bayesian optimization with inequality constraints. In: Xing EP, Jebara T (eds) *Proceedings of the 31st international conference on machine learning*, vol 32. PMLR, pp 937–945
17. Gelbart MA, Snoek J, Adams RP (2014) Bayesian optimization with unknown constraints. In: Nevin L, Zhang JT (ed) *Proceedings of the 30th conference on uncertainty in artificial intelligence*. AUAI Press, pp 250–259
18. Gelman A, Carlin JB, Stern HS et al (2013) *Bayesian data analysis*, 3rd edn. Chapman and Hall/CRC, Boca Raton. <https://doi.org/10.1201/b16018>
19. Gramacy RB, Gray GA, Le Digabel S et al (2016) Modeling an augmented Lagrangian for blackbox constrained optimization. *Technometrics* 58(1):1–11. <https://doi.org/10.1080/00401706.2015.1014065>
20. Hairer E, Wanner G (1993) *Solving ordinary differential equations I: nonstiff problems*, 2nd edn. Springer, Berlin. <https://doi.org/10.1007/978-3-540-78862-1>
21. Hennig P, Schuler CJ (2012) Entropy search for information-efficient global optimization. *J Mach Learn Res* 13(1):1809–1837
22. Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol* 117(4):500–544. <https://doi.org/10.1113/jphysiol.1952.sp004764>
23. Hutter F, Hoos HH, Leyton-Brown K (2011) Sequential model-based optimization for general algorithm configuration. In: Coello CA (ed) *Learning and intelligent optimization*. Springer, Berlin, pp 507–523. https://doi.org/10.1007/978-3-642-25566-3_40
24. Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. *J Global Optim* 13(4):455–492. <https://doi.org/10.1023/A:1008306431147>
25. Leube PC, de Barros FPJ, Nowak W et al (2013) Towards optimal allocation of computer resources: trade-offs between uncertainty quantification, discretization and model reduction. *Environ Model Softw* 50:97–107. <https://doi.org/10.1016/j.envsoft.2013.08.008>
26. López-Lopera AF, Bachoc F, Durrande N et al (2018) Finite-dimensional Gaussian approximation with linear inequality constraints. *SIAM/ASA J Uncertain Quantif* 6(3):1224–1255. <https://doi.org/10.1137/17M1153157>
27. Maatouk H, Bay X (2017) Gaussian process emulators for computer experiments with inequality constraints. *Math Geosci* 49(5):557–582. <https://doi.org/10.1007/s11004-017-9673-2>
28. MacDougall JD, Sale DG, Alway SE et al (1984) Muscle fiber number in biceps Brachii in bodybuilders and control subjects. *J Appl Physiol* 57(5):1399–1403. <https://doi.org/10.1152/jappl.1984.57.5.1399>
29. Maier B, Emamy N, Krämer A, et al (2019) Highly parallel multi-physics simulation of muscular activation and EMG. In: *Proceedings of 8th international conference on computational methods for coupled problems in science and engineering*. coupled problems 2019. International Centre for Numerical Methods in Engineering, pp 610–621
30. Maier B, Göddeke D, Huber F, et al (2022) OpenDiHu: an efficient and scalable framework for biophysical simulations of the neuromuscular system (submitted)
31. Meidner D, Rannacher R, Vihharev J (2009) Goal-oriented error control of the iterative solution of finite element equations. *J Numer Math* 17(2):143–172. <https://doi.org/10.1515/JNUM.2009.009>
32. Meißner J, Göddeke D, Herschel M (2023) Data-informed optimization for parameter and scheme selection in numerical simulations. In: *Proceedings of the 2023 international conference on management of data*. Association for computing machinery
33. Picheny V (2014) A stepwise uncertainty reduction approach to constrained global optimization. In: Kaski S, Corander J (eds) *Proceedings of the 17th international conference on artificial intelligence and statistics*, vol 33. PMLR, pp 787–795
34. Rasmussen CE, Williams CKI (2006) *Gaussian processes for machine learning*. MIT Press, Cambridge
35. Schaback R (1997) Native Hilbert spaces for radial basis functions I. In: Müller MW, Buhmann MD, Mache DH et al (eds) *New developments in approximation theory: international series of numerical mathematics*, vol 132. Birkhäuser, Cham, pp 255–282. https://doi.org/10.1007/978-3-0348-8696-3_16
36. Scheuerer M, Schaback R, Schlather M (2013) Interpolation of spatial data: a stochastic or a deterministic problem? *Eur J Appl Math* 24(4):601–629. <https://doi.org/10.1017/S0956792513000016>
37. Shahriari B, Swersky K, Wang Z et al (2016) Taking the human out of the loop: a review of Bayesian optimization. *Proc IEEE* 104(1):148–175. <https://doi.org/10.1109/JPROC.2015.2494218>
38. Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. In: Pereira F, Burges CJC, Bottou L et al (eds) *Advances in neural information processing systems*, vol 25. Curran Associates, New York p, pp 2951–2959
39. Srinivas N, Krause A, Kakade SM et al (2012) Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Trans Inf Theory* 58(5):3250–3265. <https://doi.org/10.1109/TIT.2011.2182033>
40. Stan Development Team (2021) *Stan modeling language users guide and reference manual*, 2.27. <https://mc-stan.org>
41. Strang G (1968) On the construction and comparison of difference schemes. *SIAM J Numer Anal* 5(3):506–517. <https://doi.org/10.1137/0705041>
42. Wang Z, Jegelka S (2017) Max-value entropy search for efficient Bayesian optimization. In: Precup D, Teh YW (eds) *Proceedings of the 34th international conference on machine learning*, vol 70. PMLR, pp 3627–3635
43. Williams S, Waterman A, Patterson D (2009) Roofline: an insightful visual performance model for multicore architectures. *Commun ACM* 52(4):65–76. <https://doi.org/10.1145/1498765.1498785>
44. Yang L, Shami A (2020) *On hyperparameter optimization of machine learning algorithms: theory and practice*. *Neurocomputing* 415:295–316. <https://doi.org/10.1016/j.neucom.2020.07.061>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.