HelmholtzZentrum münchen
German Research Center for Environmental Health

Efficient computational methods for parameter estimation of ordinary differential equation and mixed-effect models in systems biology

Paul Stapor

May 2020

TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Mathematik — Lehrstuhl M12 (Mathematische Modellierung biologischer Systeme)

Efficient computational methods for parameter estimation of ordinary differential equation and mixed-effect models in systems biology

Paul Stapor

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende(r):

Prof. Dr. Oliver Junge

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Jan Hasenauer
2. Prof. Dr. Matthias Chung
3. Prof. Dr. Wilhelm Huisinga

Die Dissertation wurde am 25.05.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 29.12.2020 angenommen.

Für Anja und Johnny

Acknowledgements

I want to thank ...

... my supervisor Jan Hasenauer, for giving me the possibility to join his work group and to learn so many things, for providing me help and support, for being patient and understanding, for fruitful discussions, and for his humor.

... my former and present colleagues and friends at the ICB and the University of Bonn, for so many discussions, events, and for sharing all this time with me (the good and the bad days): Anna, Atefeh, Benni, Caro, Christoph, Daniel, Dantong, David, Dennis, Elba, Emad, Erika, Eva, Fabian, Fan, Hans, Ines, Jakob, Karolina, Lea, Leonard, Linda, Lisa A., Lisa B., Maren, Markus, Nikos, Phil, Philipp, Polina, Sabrina, Simon, Susanne, Valerio, Yannik, ... and all which I forget to list here.

... the Institute of Computational Biology, with its heads Fabian Theis and Carsten Marr, for being an exciting place to work and for providing such a rich environment for doing and living science.

... the ICB Office team Marianne, Elisabeth, and Sabine, and furthermore Anna and Nina for providing help for so many administrative and organizational things and for being patient with me, no matter what I needed.

... my collaboration partners Chrisitan Tönsing and Jens Timmer in Freiburg as well as Lorenz Adlung, Marcel Schilling, Luisa Schwarzmüller, and Ursula Klingmüller in Heidelberg, for fruitful discussions and a successful collaboration.

... my cooperation partners in the CanPathPro project, not only but in particular in Berlin, Oslo, and Vigo, as our collaboration laid the foundation for this PhD thesis and provided the funding for my work.

... Daniel Kaschek, Clemens Kreutz, Matthias Chung, and many other people for discussions, explanations, and giving me the possibility to learn from them.

... my family, especially my parents and my sister, and my friends for all their support in these years.

... you, my girlfriend, then fiancé, then wife, Anja, for being with me all these years and sharing all these moments with me, the happy and the sad ones, the lucky and the bad ones, for all your support, all your patience, all your time, and that you gave me the possibility to start, write and finish this thesis, and that you stayed at my side.

... and you, Johnny, my son, for being born healthy and happy in this time, for sharing your first smile and laugh and tooth with me, and for letting me enough sleep (most of the time) for being able to finish this work.

Abstract

In systems biology, mathematical models are a common tool to describe biological processes. Especially models based on ordinary differential equations (ODEs) have aided the understanding of causal relations within dynamic processes in cell biology.

ODE models typically depend on parameters, which are a priori unknown and have to be inferred from measurement data, such as reaction rate constants or initial conditions of biochemical species, which can not be measured. Although guidelines and algorithms for this inference exist, parameter estimation and uncertainty quantification for ODE models remain challenging tasks, already for commonly used small- and medium-scale models. However, on one hand, large-scale models with thousands of parameters have been developed in recent years to understand and predict the functioning of complex diseases, such as cancer. Inferring the parameters of these models is computationally extremely challenging. On the other hand, with the rise of single-cell measurement techniques, mixed-effect models based on ODEs which capture the system dynamics also at the single-cell level have been developed, and come with challenges of their own. These developments make it necessary to constantly improve existing and develop novel mathematical methods for parameter estimation and uncertainty quantification.

In this thesis, we first transfer the principle of mini-batch optimization from training of deep neural nets to ODE models and adapt it to this new problem class. For large-scale ODE models trained on large datasets, this leads to a substantial reduction of computation time while providing better parameter estimates than currently available methods. We then show that ensemble models based on these parameter estimates provide better model predictions than the more commonly used point estimates for these large-scale models.

Afterwards, we develop second order adjoint sensitivity analysis for ODE models with time-discrete measurements. Also this reduces computation time and increases the reliability of parameter estimation and uncertainty quantification. Especially the efficiency and reliability of uncertainty analysis via profile likelihoods is substantially improved.

Finally, we turn towards mixed-effect modeling with ODEs of biological processes at the single-cell level. We propose a statistical framework that combines single-cell data and bulk data, which only provides information about the averages of cell populations. Subsequently, we present a novel parametrization which facilitates parameter estimation of covariance matrices, which are essential in mixed-effect modeling. Then, we apply these techniques to develop a survival model of erythroid progenitor cells upon stimulation with Erythropoietin.

In conclusion, the methodological contributions presented in this thesis allow efficient and reliable parameter estimation and uncertainty quantification with better scaling properties than previously available approaches. Hence, larger models based on larger datasets or more complex models, possibly based on single-cell data and bulk data at the same time, can be developed and parametrized. We hope that these contributions will enable the understanding of complex biological processes at an unprecedented level.

Zusammenfassung

Mathematische Modelle sind ein gängiges Mittel zur Beschreibung biologischer Prozesse in der Systembiologie. Vor allem Modelle mit gewöhnlichen Differenzialgleichungen (DGLen) haben beim Verständnis der Zusammenhänge innerhalb von dynamischen Prozessen in der Zellbiologie weitergeholfen.

Wie es für viele andere Modelle der Fall ist, so hängen auch DGL-Modelle von Parametern ab, deren Werte nicht im Vorhinein bekannt ist und die mit Hilfe von gemessenen Daten geschätzt werden müssen. Obwohl es viele Herangehensweisen und Algorithmen für diese Aufgabe gibt, bleiben die Parameterschätzung für DGL-Modelle und die ihr nachfolgende Unsicherheitsanalyse – selbst für die üblichen klein- bis mittelskaligen Modelle – eine komplizierte Aufgaben. Dennoch wurden zum einen in jüngster Zeit großskalige Modelle mit tausenden von Parametern entwickeln, um die Abläufe in komplexen Erkrankungen, wie z.B. Krebs, verstehen und vorhersagen zu können. Die Parameter solcher Modelle zu bestimmen jedoch ist extrem rechenaufwändig. Zum anderen hat das Aufkommen neuer Technologien, die Einzelzell-Messungen ermöglichen, die Entwicklung von Mixed-Effect-Modellen auf Basis von DGLen erlaubt, mit deren Hilfe sich Prozesse auf Einzelzell-Ebene erklären lassen. Jedoch bringt dieser Modellierungsansatz seine ganz eigenen neuen Probleme mit sich. Diese Entwicklungen machen es notwendig, die vorhandenen mathematischen Methoden zur Parameterschätzung und Unsicherheitsanalyse beständig zu verbessern und neue Methoden zu entwickeln.

Im Rahmen dieser Doktorarbeit übertragen wir das Prinzip der Mini-Batch Optimierung vom Trainieren neuronaler Netze auf DGL-Modelle und passen es dem neuen Anwendungsbereich entsprechend an. Dies führt – für großskalige DGL-Modelle, die auf großen Datensätzen trainiert werden müssen – zur erheblichen Beschleunigung der Parameterschätzung und liefert gleichzeitige bessere Schätzwerte für die Parameter als bisher verfügbare Methoden. Danach zeigen wir, dass Ensemble-Methoden, die auf diesen Parameterschätzwerten beruhen, für diesen Modelltypus zu bessere Vorhersagen führen als die für DGL-Modelle üblicherweise verwendeten Punktschätzer.

Anschließend entwickeln wir Methoden zur adjungierten Sensitivitätsanalyse zweiter Ordnung für DGL-Modelle mit zeitlich diskreten Messdaten. Diese ermöglichen eine deutlich verkürzte Rechenzeit und verlässlichere Resultate bei Parameterschätzung und Unsicherheitsanalyse. Insbesondere die Unsicherheitsanalyse durch Profilberechnungen wird dadurch wesentlich effizienter und zuverlässiger.

Schließlich widmen wir uns Mixed-Effect-Modellen von biologischen Prozessen mit DGLen auf Einzelzell-Ebene. Wir führen ein statistisches Modell ein, dass das Zusammenführen von Einzelzell-Daten und nur die Populationsmittel betreffenden Messdaten erlaubt. Dann präsentieren wir einen neuen Ansatz zur Parametrisierung von Kovarianzmatrizen, welche für Mixed-Effect-Modelle notwendig sind, der deren Parameterschätzung erleichtert. Daraufhin wenden wir die eingeführten Techniken an, um eine Modell über das Überleben unter Erythropoetin-Stimulation von Erythroblasten auf Einzelzell-Ebene zu entwickeln.

Die in dieser Doktorarbeit vorgestellten Beiträge ermöglichen die effiziente und zuverälssige Parameterschätzung und Unsicherheitsanalyse mit besserer Skalierbarkeit zu größeren Modellen als bisher verfügbare Methoden. Dadurch können größere Modelle, basierend auf größeren Datensätzen, oder komplexere Modelle, gegebenenfalls basierend auf Einzelzell- und Populationsmitteldaten, erstellt und ihre Parameter geschätzt werden. Wir hoffen, dass uns diese Beiträge das Verständnis komplexer biologische Systeme und ihrer Zusammenhänge in bisher nicht gekannter Weise ermöglichen werden.

Contents

Chapter 1

Introduction

> Mathematics is a part of physics. Physics is an experimental science, a part of natural science. Mathematics is the part of physics where experiments are cheap.
>
> V. I. Arnold

Although the house of mathematics is constructed from axioms and founded on the basement of logic, it owes many of its walls and rooms to the influence from other sciences. It was the interest to understand the movement of objects, which drove Newton to develop his form of differential calculus and it was the development of quantum mechanics, which inspired von Neumann to develop wide parts of what is now known as functional analysis. However, physics was not the only driving force for developing new mathematics: Economics inspired the field of financial mathematics, including the famous Black-Scholes equation, engineering motivated optimization theory, and computer science was strongly involved in the development of scientific computing and numerics. In this way, mathematics never existed isolated from other (natural) sciences, but always as a part of those.

Since the second half of the 20th century, the interaction of mathematics with biology is on the rise: Our understanding of cellular biology has improved dramatically, mainly for two reasons: The development of new experimental techniques and the upcoming of quantitative computational (and thus mathematical) modeling. Mathematical models allow to develop and test hypotheses about the functioning of mechanisms within cells, which are still impossible to assess experimentally. Therefore, modeling allows to "make measurable what can't be measured"[1], and the – comparably young – science employing it to explain the functioning of complex biological systems has been termed systems biology [Ideker et al., 2001, Kitano, 2002, Klipp et al., 2005].

1.1 Ordinary differential equation models of cellular processes

Mathematical models, especially mechanistic models which aim at a causal description of the underlying process by using, e.g., dynamical systems, have been playing a particular role as tool for understanding biological processes. Within the field of mechanistic modeling, ordinary differential equation (ODE) models are among the most popular approaches [Klipp et al., 2005]. For many applications, they provide a reasonable tradeoff between a detailed interpretable description of the studied process and computational tractability. Starting with the neuron model by

[1] Quote attributed to Galileo Galilei

Hodgkin and Huxley [Hodgkin and Huxley, 1952], ODE models have allowed the explanation of many processes, such as cellular signal transduction through different types of signaling cascades [Kearns and Hoffmann, 2009, Kholodenko, 2007, Swameye et al., 2003], dynamics of receptor levels [Becker et al., 2010, Hass et al., 2017], cell cycle [Lloyd, 2013, Münzner et al., 2019], apoptosis [Spencer and Sorger, 2011], metabolism [Khodayari and Maranas, 2016, Smallbone and Mendes, 2013] and gene regulation [Kühn et al., 2009].

Parameter inference for ODE models

The detailed description of biological processes with ODE models comes at a price: It requires quantities which can not be measured directly, such as reaction rate constants or concentrations of experimentally inaccessible biochemical species. These quantities have to be modeled as unknown parameters and inferred from measurement data, which is a computationally demanding process [Raue et al., 2013b]. During the past decades, many mathematical and computational methods have been established for successful model development [Funahashi et al., 2008, Mendes et al., 2009], estimation of unknown parameters [Egea et al., 2007, Raue et al., 2013b] and quantification of uncertainties of model parameters and predictions [Ballnus et al., 2017, Joshi et al., 2006, Kreutz et al., 2013]. Many of these methods have been implemented in computational toolboxes [Egea et al., 2014, Hoops et al., 2006, Raue et al., 2015], which are comparably easy to use and allow for efficient model development, model visualization, parameter estimation, generation of hypotheses, design of new experiments, predictions and uncertainty analysis.

Those methods and toolboxes have been applied with great success to improve our understanding of many processes [Adlung et al., 2017, Becker et al., 2010, Kholodenko, 2007]. However, the large majority of these research projects focused on systems, which could be studied in relative isolation and which cover only a limited amount of biochemical species and reactions. These limitations were due to the fact that studying more complex systems involving, e.g., pathway crosstalk, is computationally extremely demanding. Furthermore, much more experimental data would be needed to infer all the unknown parameters, which would be necessary to describe such systems. However, collecting experimental data, which are comprehensive enough for a satisfactory description of more complex systems, is costly and time consuming.

Recent developments

With the advent of cheaper and faster measurement techniques, such as next generation sequencing, and increasing availability of data storage, public databases with measurement data have been established [Barretina et al., 2012, Cancer Genome Atlas Network, 2012, Li et al., 2017, Yang et al., 2013]. At the same time, novel measurement approaches made it possible to study biological processes in detail at single-cell level [Butler et al., 2018, Giesen et al., 2014, Herzenberg et al., 2006, Tsien, 1998]. Moreover, computing power of processors has increased and more computing clusters offer computing services for research.

These developments have made it possible to pursue new ways in mathematical and computational modeling. Among these developments, some trends can be distinguished: First, larger and more complex models are being developed, which try to capture the functioning of whole cells [Karr et al., 2012, Münzner et al., 2019], explain pathway cross-talk in complex diseases [Fröhlich et al., 2018a, Hass et al., 2017, Korkut et al., 2015], or cover more biochemical species and involved genes [Bouhaddou et al., 2018, Fröhlich et al., 2018a]. Second, approaches for single-cell modeling are being developed (see Loos and Hasenauer [2019] for a review). They explain cell-to-cell variability by either assuming cellular dynamics to be stochastic (sometimes termed intrisic

noise), e.g., for species with low copy numbers, such as mRNA [Elowitz et al., 2002, Gillespie, 1992], or by assuming the cells themselves to be different (sometimes termed extrinsic noise), e.g., by allowing model parameters to vary across cells [Karlsson et al., 2015, Llamosi et al., 2016]. Also methods trying to combine intrinsic with extrinsic noise exist [Zechner et al., 2012], approaches for accounting for different cell populations [Hasenauer et al., 2014], possibly combining the with intra-populational variability [Loos et al., 2018b], or methods which switch between stochastic and deterministic dynamics, depending on the abundance of the corresponding species [Hepp et al., 2015].

Those new developments also carry new challenges [Kapfer et al., 2019], and novel mathematical and computational methods have been developed to cope with them (see Fröhlich et al. [2019] for a summary): Employing methods such as adjoint sensitivity analysis [Fröhlich et al., 2017a, Fröhlich et al., 2018a], highly parallelized toolboxes for parameter estimation [Penas et al., 2015, Schmiester et al., 2019a], or exploiting the problem structure when dealing with relative data [Loos et al., 2018a, Schmiester et al., 2019a, Weber et al., 2011] have helped to reduce the computational burden dramatically. Yet, many problems still persist: On one hand, especially for large ODE models, large amounts of experimental data are required to constrain the model parameters. If such large datasets consist of many different experimental conditions – which is often the case – these conditions have to be simulated independently. Problematically, a high number of experimental conditions dramatically increases the computational cost for parameter estimation. For datasets from drug screens, this can mean thousands of ODE solves – for only one simulation of the dataset [Fröhlich et al., 2018a]. On the other hand, it is unclear how reliable standard modeling approaches, such as maximum likelihood estimation, generalize to this new scale of ODE models, even if parameter estimation was successful, or whether more Bayesian approaches might result in better model performance [Henriques et al., 2017], as only few ODE models with thousands of unknown parameters are available so far and have been studied. Furthermore, although large-scale models pose a particular challenge, by far not all issues have been resolved for smaller models. For some examples (see, e.g., Hass et al. [2019] for a collection of models), reliable parameter estimation is still hard and global uncertainty analysis may be extremely challenging. Finally, in the younger field of ODE based single-cell modeling, often even best practices, which already exist for standard ODE models [Raue et al., 2013b], are still missing. In this domain, methodological developments are often motivated by and tailored to specific applications. For example, in ODE based mixed-effect models, an additional class of model parameters, which capture the variability of the cell population, has to be inferred. Mathematical methods for their description have been developed [Pinheiro, 1994], but have never evaluated and compared for their applicability to ODE based mixed-effect models.

In all, a lot of methodological research still has to be done, until all problematic aspects of ODE based modeling of biological systems are well understood. Once this is the case, new possibilities will open up for understanding biological processes at a new level of detail.

Challenges in ODE modeling

In this thesis, we consider in particular the following challenges within the field of ODE modeling:

(i) The computational cost of parameter estimation scales with the number of experimental conditions to be simulated. Since large models require more data and hence more experiments, established methods for parameter estimation become computationally prohibitive, if the dataset consists of thousands of experimental conditions.

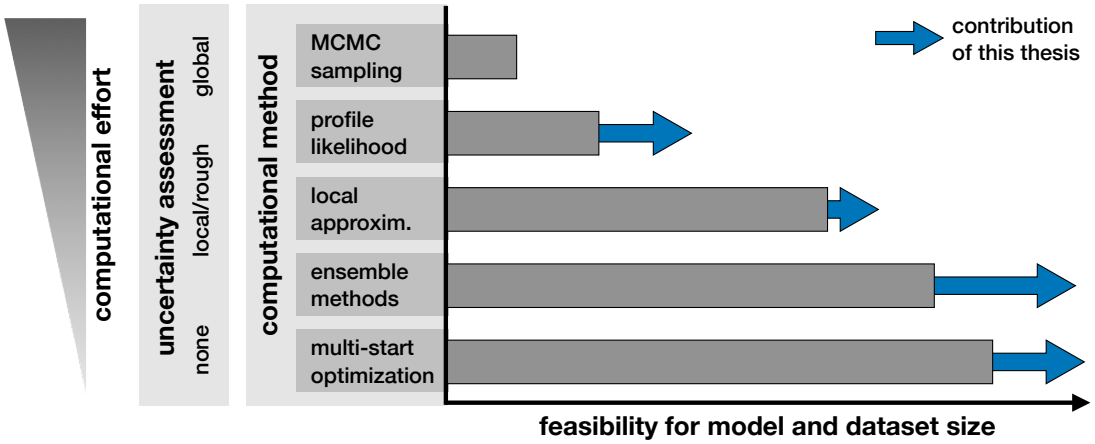

Figure 1.1: Applicability of mathematical methods for parameter estimation of ODE models to different model sizes and complexities. The more insight a method yields, the higher its computational burden tends to be. The general aim of this thesis is to push the boundaries of what is computationally feasible by introducing novel methods and by improving existing ones.

(ii) Currently, it is assumed that large-scale models tend to be "overparametrized" compared to small-scale models. Due to the high computational effort, most methods for reliable uncertainty analysis are not applicable for these models. Although a number of concepts exists how to deal with overfitting or large model uncertainties, not much is known about how effective they are for large-scale ODE models.

(iii) Already for medium-scale models, reliable parameter estimation and uncertainty analysis can be challenging for certain application examples. More robust algorithms exist (which rely on higher order derivatives), but those scale poorly with model size.

(iv) When employing mixed-effect modeling to explain data of population averages and single-cell data, it is desirable to integrate heterogeneous data types, as the model should capture the average behavior as well as the behavior of the single cells in the population. A statistical framework which allows the combination of these different data types is currently missing.

(v) When performing parameter estimation for mixed-effect models, covariance structures of model parameters have to be inferred. As the parameter estimation of mixed-effect ODE models is computationally challenging, efficient and robust computational methods are required. However, existing inference methods for parameter estimation have mostly been used to infer the trivial case of diagonal covariance matrices, which neglect the dependencies between cell-to-cell variable model parameters.

1.2 Contribution of this thesis

The overall aim of all contributions of this thesis is to improve existing computational methods and to develop novel ones to facilitate parameter optimization and uncertainty analysis for larger models on larger datasets (Figure 1.1). More precisely, the above mentioned challenges are addressed by contributions in this thesis in the following way:

(a) Mini-batch optimization is transferred from the field of deep learning and adapted to parameter estimation of ODE models to overcome the linear scaling of computation time with the

number of simulation conditions, and therefore, challenge (i) is addressed. As this reduces the total computation time substantially, free resources can be used to better explore the parameter space, which enables the use of ensemble models. Those are then compared to the more commonly employed point estimates for creating model predictions, which addresses challenge (ii).

(b) Second order adjoint sensitivity analysis is developed for ODE models with time-discrete measurements in order to efficiently compute second order derivatives of the objective function. Subsequently, the positive impact of these derivatives on optimization and profile calculation is demonstrated, therefore addressing challenge (iii).

(c) Single-cell data explaining cell populations is integrated with bulk data of population averages in a combined statistical framework. The formulation of a common likelihood function allows parameter estimation for mixed-effect ODE models based on these heterogenous data types, such that the model can describe the average population behavior as well as the observed cell-to-cell variability. This addresses challenge (iv).

(d) A Lie-theoretic approach for efficient parametrization of symmetric positive definite (covariance) matrices, which have to be inferred in mixed-effect modeling, is developed. This enables efficient parameter estimation of mixed-effect models with full and block-diagonal covariance matrices and hence addresses challenge (v).

These contributions are part of manuscripts, which have either already been published in peer-reviewed journals or are currently under review at such. Therefore, parts of this thesis are similar or even identical to the following publications:

- **Stapor, P.***, Weindl, D.*, Ballnus, B., Hug, S., Loos, C., Fiedler, A., Krause, S., Hross, S., Fröhlich, F., Hasenauer, J. (2018). PESTO: Parameter EStimation TOolbox. Bioinformatics, 34(4)
- **Stapor, P.**, Fröhlich, F., Hasenauer, J. (2018). Optimization and profile calculation of ODE models using second order adjoint sensitivity analysis. Bioinformatics, 34(13)
- **Stapor, P.**, Schmiester, L., Wierling, C., Lange, B., Weindl, D., Hasenauer, J. (2019). Mini-batch optimization enables training of ODE models on large-scale datasets. bioRxiv, 10.1101/859884, *Under review*
- **Stapor, P.***, Adlung, L.*, Tönsing, C.*, Schmiester, L., Schwarzmüller, L., Wang, D., Timmer, J., Klingmüller, U., Hasenauer, J., Schilling, M. (2019). Cell-to-cell variability in JAK2/STAT5 pathway components and cytoplasmic volumes define survival threshold in erythroid progenitor cells. bioRxiv, 10.1101/866871, *Under review*

Beyond these publications, I have contributed to the following manuscripts, some of which have already been published in peer-reviewed journals:

- **Stapor, P.***, Kapfer, E.*, Hasenauer, J. (2019). Challenges in the calibration of large-scale ordinary differential equation models. IFAC-PapersOnLine, 52(26)
- Schälte, Y., **Stapor, P.**, Hasenauer, J. (2018). Evaluation of Derivative-Free Optimizers for Parameter Estimation in Systems Biology. IFAC-PapersOnLine, 51(19)
- Wang, D., **Stapor, P.**, Hasenauer, J. (2019). Dirac mixture distributions for the approximation of mixed effects models. IFAC-PapersOnLine, 52(26)

- Lines, G., Paszkowski, L., Schmiester, L., Weindl, D., **Stapor, P.**, Hasenauer, J. (2019). Efficient Computation of Steady States in Large-Scale ODE Models of Biochemical Reaction Networks. IFAC-PapersOnLine, 52(26)

- Schmiester, L.*, Schälte, Y.*, Bergmann, F. T., Camba, T., Dudkin, E., Egert, J., Fröhlich, F., Fuhrmann, L., Hauber, A. L., Kemmer, S., Lakrisenko, P., Loos, C., Merkt, S., Pathirana, D., Raimundez, E., Refisch, L., Rosenblatt, M., **Stapor, P.**, Städter, P., Wang, D., Wieland, F.-G., Banga, J. R., Timmer, J., Villaverde, A. F., Sahle, S., Kreutz, C., Hasenauer, J., Weindl, D. (2020). PEtab – interoperable specification of parameter estimation problems in systems biology. arXiv, 2004.01154v2 [q-bio.QM], *Under review*

- Staedter, P.*, Schälte, Y.*, Schmiester, L.*, Hasenauer, J., **Stapor, P.** (2020). Assessment of ODE solver performance for biological processes, *in preparation*

Besides the mathematical contributions to these articles, I have been involved in the development and maintenance of the following computational toolboxes, which have been used in multiple peer-reviewed publications:

- **PESTO**, *Parameter EStimation TOolbox*: A toolbox for optimization and uncertainty quantification based on profile likelihood computation and Markov chain Monte Carlo sampling methods with customizable visualization features, written in MATLAB.

- **AMICI**, *Advanced Multi-language Interface to CVODES and IDAS*: An ODE-solver toolbox, which interfaces the high-performing ODE and DAE solvers CVODES and IDAS from the SUNDIALS package written in C. AMICI can be interfaced from different programming languages (C++, Python, and MATLAB), has additional functionality for computing likelihood functions and first and second order derivatives thereof using forward, adjoint and steady-state sensitivity analysis, and is implemented in C++, Python, and MATLAB.

- **parPE**, *parallel Parameter Estimation framework*: An optimization toolbox for highly parallelized evaluation of likelihood functions and their gradients based on AMICI, allowing for mini-batch and full-batch optimization methods and designed for high performance infrastructures, written in C++ and Python.

- **MEMOIR**, *Mixed-Effect MOdel InfeRence*: A toolbox computing symbolic expressions for non-linear mixed effect modeling based on multiple experiments and allowing the integration of heterogeneous data types, which can be interfaced with AMICI for mixed-effect models which rely on ODEs and with PESTO for parameter estimation. MEMOIR is written in MATLAB.

- **SPToolbox**, Sigma Point Toolbox: Allows the parametrization of covariance matrices and the simulation and approximation of cell populations by different sigma point methods or Monte Carlo sampling, written in MATLAB.

- **PEtab**, *Parameter Estimation tabular formulation*: A model and data format relying on SBML models and tsv spreadsheets, which allows a standardized formulation of parameter estimation problems. PEtab maps between experimental conditions of measured data and simulation conditions for efficient parameter estimation, with additional visualization functionalities.

1.3 Outline of this thesis

The remainder of this thesis is structured as follows: In Chapter 2, the background knowledge and notation for ODE based models of biological systems is introduced, first for the cases of general ODE models, then for mixed-effect models, which capture dynamics at the single-cell level. Then, existing methods for parameter optimization and uncertainty analysis are described.

Chapters 3 to 5 are based on published manuscripts and present the actual contributions of this thesis. Each of these chapters starts with an introductory section, motivating the contribution, presenting the corresponding research question, and giving background information on the previous state-of-the-art in the research field. In a second section, the new methodology and its implementation are described in detail and applications are presented. In a last section, the impact of the contribution is discussed.

In Chapter 3, mini-batch optimization of ODE models based on large-scale datasets is presented, corresponding to contribution (a). Chapter 4 presents second order adjoint sensitivity analysis as method to improve parameter estimation and uncertainty analysis, therefore relating to contribution (b). In Chapter 5, methods for parameter estimation of mixed-effect models based on ODEs are presented, which allows to combine single-cell data with population average data, corresponding to contribution (c), and a Lie-theoretic approach to parametrize covariance matrices is presented, which relates to contribution (d).

This thesis concludes with Chapter 6, in which a summary of the thesis' contribution is given and an outlook on possible next steps in the research of ODE models is presented.

Chapter 2

Background

In this chapter, we introduce the terminology and notation for ODE models, describe the methods, on which this thesis is founded, and present the most important challenges, which were the scientific motivation for this thesis, in detail. This chapter is based on my work in the following publications and hence, some parts may be similar or even identical to them:

- **Stapor, P.**, Fröhlich, F., Hasenauer, J. (2018). Optimization and profile calculation of ODE models using second order adjoint sensitivity analysis. Bioinformatics, 34(13)
- **Stapor, P.**, Schmiester, L., Wierling, C., Lange, B., Weindl, D., Hasenauer, J. (2019). Mini-batch optimization enables training of ODE models on large-scale datasets. bioRxiv, 10.1101/859884, *Under review*
- **Stapor, P.***, Adlung, L.*, Tönsing, C.*, Schmiester, L., Schwarzmüller, L., Wang, D., Timmer, J., Klingmüller, U., Hasenauer, J., Schilling, M. (2019). Cell-to-cell variability in JAK2/STAT5 pathway components and cytoplasmic volumes define survival threshold in erythroid progenitor cells. bioRxiv, 10.1101/866871, *Under review*

2.1 Modeling of cellular processes

2.1.1 Modeling with ordinary differential equations (ODEs)

Ordinary differential equations (ODEs) are a common approach to model the time-evolution of bio-chemical species in living cells. The underlying assumptions when using ODE models are firstly, that spatial inhomogeneities can be neglected within one cellular compartment and secondly, that all species are sufficiently abundant that the process is governed by deterministic dynamics. The opposite would be either models based on partial differential equations, which incorporate spatial information, or systems in which species have low copy numbers and which are hence governed by stochastic dynamics, respectively [Klipp et al., 2005].

The time evolution of the system

We denote the time as $t \in [0, T] \subset \mathbb{R}$ and the vector of state variables as $x(t) \in \mathbb{R}^{n_x}$. The state vector describes the concentration of $n_x \in \mathbb{N}$ biochemical species, which are, for example, (phospho-)proteins, ligands, receptor or mRNA levels, or, in some cases, volumes of cell compartments. The time evolution of x is given by a vector field f, which depends on a vector of

unknown parameters $\theta \in \mathbb{R}^{n_\theta}$ and on a vector of known input parameters $u \in \mathbb{R}^{n_u}$:

$$\frac{d}{dt}x(t,\theta,u) = f(x(t,\theta,u),\theta,u), \quad \text{with } x(0,\theta,u) = x_0(\theta,u) \tag{2.1}$$

Parameters are typically reaction rates constants, shuttling rates between compartments, or initial concentrations of proteins, which cannot be measured directly and have to be estimated. As those quantities should always be positive, it is common to parametrize them logarithmically and exponentiate those values, which ensures their positivity. Input parameters are usually external stimuli such as treatments with drugs or other ligands, measured initial concentrations of certain species, mRNA expression levels or genetic profiles. Parameters and input parameters are assumed to stay constant over time.

In the applications of this thesis, the right hand side f is Lipschitz-continuous (in most applications even smooth) on the time interval of interest $[0, T]$ and for $x_i \geq 0$, with $i = 1, \ldots, n_x$, such that the Picard-Lindelöf theorem guarantees the existence and uniqueness of a solution $x(t)$ at least in a neighborhood of the initial condition. Depending on f, θ, and u, and since the right-hand side is typically not linearly bounded, this solution does not have to exist on the whole interval $[0, T]$. In general however, there is no solution in closed form [Raue et al., 2013b]. Yet, if the studied biological system is modeled appropriately and θ is in a biologically plausible domain Ω of the parameter space $\mathbb{R}^{n_\theta}$, a solution should exist for the whole interval $[0, T]$: Biological systems usually show either oscillatory dynamics or evolve towards a steady-state, in rare cases they may show chaotic behavior. Hence, if an ODE describes the dynamics of a biological system with sufficient accuracy, its solution should have no finite time singularities in $[0, T]$ (such as possible for, e.g., Riccati equations), as this has no analogy in biology.

The lack of analytical solutions makes it necessary to use numerical methods, such as Runge-Kutta, Adams-Moulton, or BDF schemes [Maiwald and Timmer, 2008, Mendes et al., 2009, Raue et al., 2013b]. Depending on the equation, the employed ODE solver, and its settings, numerical integration of Equation 2.1 may fail, which is a common problem when using ODE models of biological process, in particular during parameter estimation [Chung et al., 2017b, Raue et al., 2013b].

Measurement data and observation functions

A frequent aim of ODE models is to describe a vector of measurable quantities $y(t) \in \mathbb{R}^{n_y}$, which we will call observables. In most cases, it is impossible to measure state variables x directly, as many measurement techniques only allow an assessment on a relative scale (e.g., quantitative polymerase chain reaction, qPCR: Gibson et al. [1996]), maybe with an additional offset (e.g., immunoblotting Renart et al. [1979]), on a (bi-)exponential scale (e.g., flow cytometry: Herzenberg et al. [2006]), or as sums of certain state-variables (e.g., if an antibody binds to a protein, independent of its phosphorylation state, see e.g., Bachmann et al. [2011]). For this reason, we introduce observation functions, henceforth denoted by h [Raue et al., 2013b]:

$$y(t,\theta,u) = h(x(t,\theta,u),\theta,u) \tag{2.2}$$

Observable functions (or "observables", for short) can be matched to experimental data D, collected at timepoints $t_j, j = 1, \ldots, n_t$ or at steady-state, if a steady-state is observed in the studied system. In this case, we denote the steady-state as:

$$x^*(\theta,u) = \lim_{t\to\infty} x(t,\theta,u) \tag{2.3}$$

The steady-state can either be computed directly by root-finding methods, such as Newton's method, or by integrating the ODE until the norm of the right hand side undergoes a previously defined threshold $\gamma > 0$ at some late timepoint t^* [Mendes et al., 2009]. This grounds on the assumption:

$$\left\| \frac{dx}{dt}(t^*, \theta, u) \right\| < \gamma \quad \Longrightarrow \quad \forall_{t>t^*} \colon \|x(t, \theta, u) - x^*(\theta, u)\| < \|x(t^*, \theta, u) - x^*(\theta, u)\| \tag{2.4}$$

Often, measured data is available for different perturbations of the system: The system can be treated with different drugs at different doses and knockouts or over-expressions of certain genes can be induced. Such perturbations are usually captured by different vectors of input parameters u^k, where k denotes the k-th of n_e experimental conditions [Raue et al., 2013b]. As u influences the dynamics of the system, the time evolution differs for distinct experimental conditions. Hence, to simulate the whole dataset once, n_e different initial value problems have to be solved [Raue et al., 2013b]. The whole dataset is then given as:

$$D = \{\bar{y}_{ij}^k\}_{\substack{i=1,\dots,n_y^k \\ j=1,\dots,n_t}}^{k=1,\dots,n_e} \tag{2.5}$$

In the following, i will always be used to enumerate the entries of the observable vector, j will be used for the different measurement timepoints, and k will always index the experimental condition. This indexing scheme will be maintained throughout this thesis.

Single-cell data and mixed-effect modeling

If single-cell data is collected in order to gain particular insights, a mathematical model should reflect this additional dimension of the data. Depending on the modeled system and the scientific question, many different approaches exist, which attempt to reflect various additional biological aspects, such as heterogeneity in cellular response upon treatment within a cell population [Spencer et al., 2009], the existence of subpopulation structures within a cell population [Hasenauer et al., 2014], or the differentiation process of single cells [Fischer et al., 2019]. A common approach which can be nicely combined with ODE modeling are mixed-effect models (MEMs) [Karlsson et al., 2015, Loos and Hasenauer, 2019]. The approach of ODE MEMs uses deterministic dynamics, assuming high copy numbers of all involved biochemical species. Conceptually, this captures cell-to-cell variability which arises from cells having different properties, e.g., different protein abundances or cell or compartment sizes [Spencer et al., 2009].

Mathematically, a MEM allows different parameter vectors for different cells by using a statistical model for the cell population. An in-silico cell population (reflected by a set of different parameter vectors) is generated according to the MEM and then propagated through the ODE [Tornoe et al., 2004]. As the ODE provides a nonlinear map from parameters to model outputs, this yields a nonlinear mixed-effect model (NLMEM). The parameter vector for the ℓ-th cell (or, more generally, the ℓ-th individual) is denoted as ϕ^ℓ and given by:

$$\phi^\ell = F\beta + Rb^\ell \tag{2.6}$$

Here, F and R are called the design matrices for fixed and random effects, $\beta \in \mathbb{R}^{n_\beta}$ is the vector of so called fixed effects (usually a parameter set describing the population mean), $b^\ell \in \mathbb{R}^{n_b}$ is the vector of random effects for the ℓ-th individual, which captures the deviations of the ℓ-th individual from the population mean. The fixed effects β influence the parameters of all

individuals, while the random effects b^ℓ are specific to a single individual. We will assume that the random effects, which are often parametrized logarithmically, follow a multivariate normal distribution with mean 0 and covariance matrix Σ, $b^\ell \sim \mathcal{N}(0, \Sigma)$, which is motivated by biological findings [Fröhlich et al., 2018b]. The covariance matrix Σ is parameterized by a vector δ, yielding the parameter vector $\theta = (\beta, \delta)$, which describes the dynamics of the whole population. Hence, when compared to a classical population average model as discussed before, in which only $\theta = \beta$ has to be inferred, the MEM approach extends the parameter estimation problem by the parameters of the population distribution which are grouped in δ.

The concept of NLMEMs (or sometimes called NONMEMs) is widely used in pharmacokinetics and pharmacodynamics (see, e.g., [Melin et al., 2020] for a recent application or [Sheiner and Beal, 1983] for an early method review) to account for differences between individuals in studies. In systems biology, most publications which use the term NLMEM work with single-cell time-lapse data [Fröhlich et al., 2018b, Karlsson et al., 2015, Llamosi et al., 2016], which is often measured by single-cell microscopy. Yet, also other approaches which are employed for analyzing single-cell snapshot data, collected by flow or mass cytometry, effectively use NLMEMs as well [Loos et al., 2018b]. However, the problem in this case is that cells from different time points in a time series of snapshot data are not identical and cannot be compared directly. Hence, only the shape of the distribution of single-cell observables can be used to infer the distribution parameters in δ. In general, this is less informative than fitting single-cell trajectories to single-cell time-lapse data, but can still yield important insights into cellular processes. However, conceptually all the mentioned data types – population average data, single-cell snapshot data, and single-cell time-lapse data – could be combined when using an ODE MEM (Figure 2.1).

When employing an ODE MEM with a population of n_c individuals and a dataset with n_e experimental conditions, $n_e \cdot n_c$ trajectories of state variables must be integrated, i.e., $n_e \cdot n_c$ initial value problems have to be solved. The time evolution of each of those is given by

$$x^{\ell,k}(t) = x(t, \phi^\ell, u^k) = \int_0^t f(x(\tau, \phi^\ell, u^k), \phi^\ell, u^k) d\tau \quad \text{with} \quad x(0)^{\ell,k} = x_0(\phi^\ell, u^k) \tag{2.7}$$

for a random effect vector b^ℓ, where f is again the vector field of the underlying ODE model. Denoting the observation functions for the distinct individuals by $\widetilde{h}$ yields observable trajectories $\widetilde{y}$:

$$\widetilde{y}^{\ell,k}(t) = \widetilde{h}(x^{\ell,k}(t), \phi^\ell, u^k) \tag{2.8}$$

From those, the observation functions h at the population level can be computed, which can be, e.g., statistical moments or percentiles:

$$y(t, \theta, u^k) = h(\widetilde{Y}^k(t), \theta, u^k), \quad \text{with } \widetilde{Y}^k(t) = (\widetilde{y}^{1,k}(t), \ldots, \widetilde{y}^{n_c,k}(t)) \tag{2.9}$$

2.1.2 Statistical inference techniques

Noise models and likelihood functions

Observable functions describe measurement data and hence link them to a computational model. But even if a model reflected the biological process perfectly, model output and measured data would not coincide exactly, as measurements are always noise-corrupted. For this reason, we have to construct a noise model for the measured data. In most cases, a time-series of a measured observable is assumed to have noise terms which are independent, identically distributed and

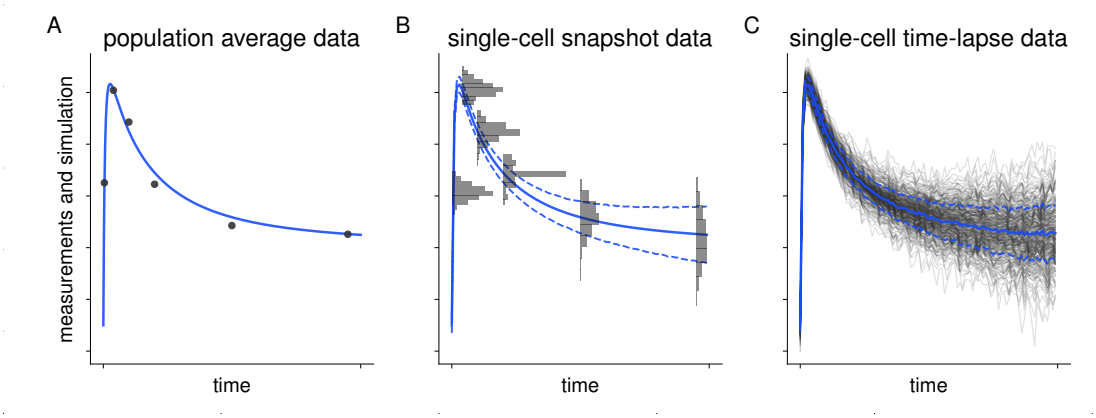

Figure 2.1: Different types of measurement data and simulations, which would be desirable to combine for parameter estimation in mechanistic single-cell modeling. **A** Exemplary time-course of population average data (grey) and corresponding simulation from ODE model (blue). **B** Exemplary time-course of single-cell snapshot data (grey histograms) and corresponding simulation of population mean (solid line) and standard deviations (dashed lines) from an ODE MEM (blue). **C** Exemplary time-course of single-cell time-lapse data (one grey line per measured cell) and summarized simulations from an ODE MEM (blue) depicting the simulated population mean (solid line) and simulated standard deviations (dashed lines).

modeled as multiplicative or additive random variables [Raue et al., 2013b]. Although other approaches have shown to be beneficial if, e.g., the dataset contains outliers [Maier et al., 2017] or correlated noise [Sommerlade et al., 2015], the most common approach is an i.i.d. additive Gaussian noise model [Raue et al., 2013b]:

$$\bar{y}_{ij}^k = y_i(t_j, \theta, u^k) + \varepsilon_{ij}^k, \qquad \text{with } \varepsilon \sim \mathcal{N}\left(0, \left(\sigma_{ij}^k\right)^2\right) \tag{2.10}$$

If the variance of the noise model is unknown, it can be modeled with additional parameters, yielding $\sigma_{ij}^k = \sigma_{ij}^k(\theta)$. Such a statistical model allows the computation of the likelihood of a measurement value $y_i(t_j, \theta, u^k)$ given a parameter vector θ. Exploiting the assumed independence of the measurement noise terms, we get:

$$\mathcal{L}(D \,|\, \theta) = \prod_{k=1}^{n_e} \prod_{i=1}^{n_y^k} \prod_{j=1}^{n_t} \frac{1}{\sqrt{2\pi}\sigma_{ij}^k(\theta)} \exp\left\{-\frac{1}{2}\left(\frac{\bar{y}_{ij}^k - y_i(t_j, \theta, u^k)}{\sigma_{ij}^k(\theta)}\right)^2\right\} \tag{2.11}$$

Maximum likelihood estimation

The statistical model makes it possible to fit the unknown model parameters θ to the measurement data by maximizing the likelihood $\mathcal{L}(D \,|\, \theta)$. However, it is more common to work with the negative logarithm of the likelihood, as firstly, most optimization algorithms are implemented as minimization algorithms and secondly, the logarithm of the likelihood tends to be more convex and is thirdly also numerically better tractable [Raue et al., 2013b]. Working with the already discussed logarithms of the model parameters increases the numerical tractability further [Hass et al., 2019]. This yields the negative log-likelihood as objective or cost function for an optimiza-

tion problem:

$$J_{\text{nllh}}(\theta) = -\log\left(\mathcal{L}(D \mid \theta)\right) = \frac{1}{2}\sum_{k=1}^{n_e}\sum_{i=1}^{n_y^k}\sum_{j=1}^{n_t}\left(\left(\frac{\bar{y}_{ij}^k - y_i(t_j,\theta,u^k)}{\sigma_{ij}^k(\theta)}\right)^2 + \log\left(2\pi\left(\sigma_{ij}^k(\theta)\right)^2\right)\right) \tag{2.12}$$

In parameter estimation, a typical aim is to first find the global minimizer of J_{nllh}, which is the maximum likelihood estimator θ^{MLE}:

$$\theta^{MLE} = \text{argmin}_{\theta\in\Omega}\, J_{\text{nllh}}(\theta) \tag{2.13}$$

Minimizing J is equivalent to optimizing the quality of the fit of the model to the data. For an additive Gaussian noise with known measurement noise (i.e., $\sigma_{ij}^k(\theta) = \sigma_{ij}^k$), finding θ^{MLE} is moreover equivalent to employing a weighted least squares algorithm.

When dealing with ODE MEMs, the likelihood needs to be formulated differently. As, depending on the data and on the modeling approach, different likelihood formulations exist, we refer to Chapter 5 for the corresponding equations.

Parameter priors and maximum a posteriori estimation

During parameter estimation, the parameter vector θ is restricted to a region $\Omega \in \mathbb{R}^{n_\theta}$ in parameter space, which is assumed to be biologically plausible. This type of restriction is often called box constraint, as Ω usually has the shape of a bounded box. In some cases, prior knowledge from biology beyond box constraints is available and in recent years, databases which collect this information have been created and made publicly available [Schomburg et al., 2013, Wittig et al., 2012]. They can be queried to find such parameter priors manually or in an automated fashion. If possible, this additional knowledge should be exploited to facilitate parameter estimation.

Mathematically, priors can be incorporated via Bayes' theorem:

$$\text{p}(\theta \mid D) = \frac{\text{p}(D \mid \theta)\cdot \text{p}(\theta)}{\text{p}(D)} \Rightarrow \text{p}(\theta \mid D) \propto \text{p}(D \mid \theta)\cdot \text{p}(\theta) \tag{2.14}$$

Here, $\text{p}(D \mid \theta)$ denotes the likelihood of observing the dataset D given the parameter vector θ, $\text{p}(\theta)$ describes the prior knowledge about the probability distribution of θ, and $\text{p}(\theta \mid D)$ is the posterior distribution of θ given D, which accounts for the additional prior knowledge. The term $\text{p}(D)$ is called evidence and describes the probability to observe the dataset D. In most cases, $\text{p}(D)$ is hard to compute [Kramer et al., 2010], but as it is just a normalization constant, it can be omitted for many purposes [Hasenauer, 2013].

Looking again at the logarithm, the evidence becomes an additive constant, which does not affect the location of the local minima of the objective function. Furthermore, many parameter inference algorithms are not affected by the evidence. As this is also the case for the methods, which are studied and developed in this thesis, the evidence will be neglected in the remainder of this work. However, the prior does affect the objective function:

$$J_{\text{nlp}}(\theta) = -\log\left(\mathcal{L}(D \mid \theta)\right) - \log\left(\text{p}(\theta)\right) = J_{\text{nllh}} - \log\left(\text{p}(\theta)\right) \tag{2.15}$$

When performing parameter estimation with prior information, the objective function is the negative logarithm of the posterior J_{nlp} instead of the negative logarithm of the likelihood J_{nllh}.

Thus, the globally optimal parameter vector is called the maximum a posteriori estimate θ^{MAP}:

$$\theta^{MAP} = \text{argmin}_{\theta \in \Omega} J_{\text{nlp}}(\theta) \tag{2.16}$$

In many cases, only flat priors with compact support according to the box constraints are available yielding $\text{p}(D \,|\, \theta) \propto \text{p}(\theta \,|\, D)$ for $\theta \in \Omega$. Then, likelihood and posterior only vary by a constant factor – and their logarithms only by an additive constant. In these cases, the location of the minima of the objective function remains unaltered by the prior. In this thesis, we will always incorporate prior knowledge if available and hence always use the term posterior instead of likelihood and denote the negative log-posterior as objective or cost function $J = J_{\text{nlp}}$.

2.1.3 Interpretation of model parameters

Exploiting Bayes' theorem also opens up the possibility to interpret model parameters in a different way. The (negative log-) likelihood $-\log\big(\mathcal{L}(D \,|\, \theta)\big)$ suggests that there is a true (or at least: best) parameter vector θ^{true}, which has to be found and that the measurement data D is just one realization of a ground truth, based on θ^{true} [Raue et al., 2013a]. This point of view can be chosen, unaffectedly of whether prior knowledge is incorporated or not and it is called the *frequentist* perspective. However, the fact that the posterior describes a probability of θ conditioned on the data D suggests that the observed data D is the fundamental property and that the model parameters θ should be regarded as stochastic. This point of view is called the *Bayesian* perspective. Both interpretations of θ and D have their advantages and disadvantages and both suggest the use of different techniques during parameter estimation. Yet, both perspectives can also profit from each other and which one is used should depend on the specific model, its applications, and the scientific question behind [Raue et al., 2013a].

Point estimates - frequentist perspective

In the frequentist perspective, no prior knowledge is incorporated and finding θ^{MLE} is of major importance [Fisher, 1922]. Hence, parameter estimation is particularly concerned with parameter optimization. Once the optimal parameter vector is found, the next main goal is to determine its uncertainty [Raue et al., 2013a]. This approach has some practical advantages: First, numerical techniques for optimization are fairly well developed and thus, parameter optimization is still a challenging but often feasible task. Second, having one best parameter vector allows a clear biological interpretation of its values as biological quantities. Third, if only one parameter vector is to be considered and not a multi-dimensional distribution of parameters, generating model predictions is straightforward. Those advantages, especially the numerical tractability, have allowed the rapid development of ODE models of many biological processes, which have supported our understanding of biology.

But there are also disadvantages: First, model predictions are typically made based on θ^{MLE} alone. This means that (possibly available) information about the shape of the objective function landscape is not reflected within those predictions, although it is known that model simulations based on θ^{MLE} don't have to yield the most probable value for a model prediction [Maier et al., 2020]. Second, assessing the uncertainties of model predictions is feasible, but not straightforward.

From a frequentist perspective, it is desirable to refine a model, until it has one sharp and clear global optimum [Maiwald et al., 2016, Raue et al., 2010]. For such models, using the best parameter vector as point estimate for model predictions may indeed be an appropriate

approximation, as most of the probability mass is located around the global optimum. However, this will generally not be the case for not exhaustively refined ODE models.

Bayesian approaches and ensemble modeling

The Bayesian perspective is less concerned with the optimal parameter vector, but more with the whole parameter distribution. Moreover, the use of prior information is an essential feature of Bayesian methods. By exploiting these two aspects, additional information about the studied system can be incorporated into model predictions. Hence, the main goal is typically to generate a representative sample from the posterior distribution. Once this has been achieved, the whole sample can be used for model predictions. This allows an easy interpretation of model uncertainties.

For this reason, the Bayesian perspective may have the advantage of being more satisfactory from the statistical point of view. On the other hand, it has the important disadvantage that generating a representative sample of the whole posterior distribution is computationally substantially more challenging than finding the best parameter vector [Ballnus et al., 2017]. Even though a large number of sampling algorithms exists, among which Markov-chain Monte Carlo algorithms are probably the most prominent ones [Hastings, 1970, Metropolis et al., 1953], it is notoriously hard to assess whether they really generated a representative sample, as there are many possible pitfalls [Ballnus et al., 2017, Bayarri and Berger, 2004, Raue et al., 2013a].

Especially for models which have not been refined iteratively, Bayesian approaches may yield different model predictions than frequentist ones [Raue et al., 2013a]. In principle, many sampling strategies are guaranteed to create a representative sample if they are run sufficiently long. However, it may be unclear what "sufficently long" means in practical applications. But even if it is impossible to assess the whole posterior distribution, lightweight approximations such as using an ensemble of parameter vectors can improve the reliability of model predictions compared to simple point estimates [Henriques et al., 2017, Villaverde et al., 2019].

2.2 Parameter estimation

2.2.1 Parameter optimization

In most cases, parameter optimization is the first step to be taken if an ODE model has been set up and measurement data has been linked to it via observable functions [Raue et al., 2013b]. Optimizing the parameters of an ODE model is often complicated, as the evaluation of the objective function depends on the (numerical) solution of an ODE and is hence not available with arbitrary precision [Klipp et al., 2005, Raue et al., 2013b]. The ODE makes the objective function highly non-linear in the parameters, in general non-convex, often multi-modal, and computationally costly to evaluate. For this reason, methods have been developed which relax the ODE constraint: Multiple shooting algorithms divide the time-interval for ODE integration into subintervals [Bock and Plitt, 1984, Peifer and Timmer, 2007], and continuous shooting relaxes the ODE constraint by approximating the ODE solution by, e.g., splines functions [Chung et al., 2017b]. However, these methods have their own challenges and this thesis will focus to the so-called single-shooting approach, i.e., ODE integration over the whole time-interval, while using implicit ODE solvers (e.g., the BDF scheme) to reduce failure of ODE integration.

From a mathematical point of view, finding the global optimum of a multi-modal, computationally expensive, and possibly high-dimensional objective function is problematic, as theoretical results for this situation are rare. Yet, a number of fairly well-working methods and best practices

has been established in recent years [Ballnus et al., 2017, Mendes et al., 2009, Raue et al., 2013b, Villaverde et al., 2018], and hopefully, these rather empirical method studies will at some point allow to better understand the mathematical properties of these parameter estimation problems. In the following, we want to briefly review the most common methods for parameter optimization and uncertainty quantification in the field of systems biology.

Global parameter optimization

For global optimization of biologically motivated ODE models, many approaches exist [Fröhlich et al., 2019]. The simplest idea, called multi-start local optimization, is to start many randomly initialized local optimizations [Raue et al., 2013b]. The results of these local searches can be stored, sorted by the final objective function value, and visualized in waterfall plots, which can – if the local optimization runs worked well – reveal the structure of the local optima of the objective function. The best local optimum is then assumed to be the global optimum (Figure 2.2).

Besides multi-start local optimization, many other methods exist for global optimization of ODE models: the most commonly used are genetic algorithms [Hansen and Ostermaier, 1996, Mitchell, 1998], swarm-based methods [Kennedy, 2011, Vaz and Vicente, 2007], or simulated annealing [Kirkpatrick et al., 1983]. Another, very popular method is scatter-search [Egea et al., 2014, Penas et al., 2015], which is a hybrid approach between a genetic algorithm and multi-start local optimization. It aims to combine the global exploration of a genetic algorithm with the local exploitation of local optimization techniques.

For ODE models in systems biology, many studies have shown that methods which employ sophisticated techniques for local optimization, tend to outperform other approaches for many example models [Egea et al., 2014, Raue et al., 2013b, Schälte et al., 2018, Stapor et al., 2018b, Villaverde et al., 2018]. For that matter, it seems to be important to exploit (accurate) derivative information of the objective function, as this substantially improves the quality of local optimization [Raue et al., 2013b, Schälte et al., 2018, Villaverde et al., 2018]. Without the use of accurate derivatives, which can be obtained via, e.g., sensitivity analysis, also multi-start local optimization often fails to achieve satisfactory results [Schälte et al., 2018]. In this thesis, we will follow the idea of multi-start optimization with (different) derivative-based local optimization strategies for all optimization tasks.

Local parameter optimization

Many algorithms for local optimization exist, but most techniques are either based on least-squares algorithms such as the Gauss-Newton-type methods, or on interior-point methods. Least-squares approaches are often combined with trust-region techniques for step-size control [Coleman and Li, 1996, Dennis et al., 1981]. Interior-point algorithms [Boyd and Vandenberghe, 2004] typically compute descent directions by Newton's method using either explicit second order derivatives [Byrd et al., 2000], by quasi-Newton methods such as (L-)BFGS [Fletcher and Powell, 1963, Goldfarb, 1970, Nocedal, 1980] or SR1 techniques [Byrd et al., 1996] using first order derivatives, or by different flavors of the conjugate gradient method [Andrei, 2009, Branch et al., 1999, Nash, 1984]. They use either line-search methods [Wächter and Biegler, 2006] or again trust-region approaches [Byrd et al., 2000, Coleman and Li, 1996] for step-size control. Besides these approaches, also gradient-free local optimization methods exist, among which hill climbing techniques [De La Maza and Yuret, 1994], iteratively updated (quadratic models) [Powell, 2009], or simplex-based methods such as the Nelder-Mead algorithm [Nelder and Mead, 1965] are most frequently used. However, these gradient-free approaches usually turn out inferior for biologically

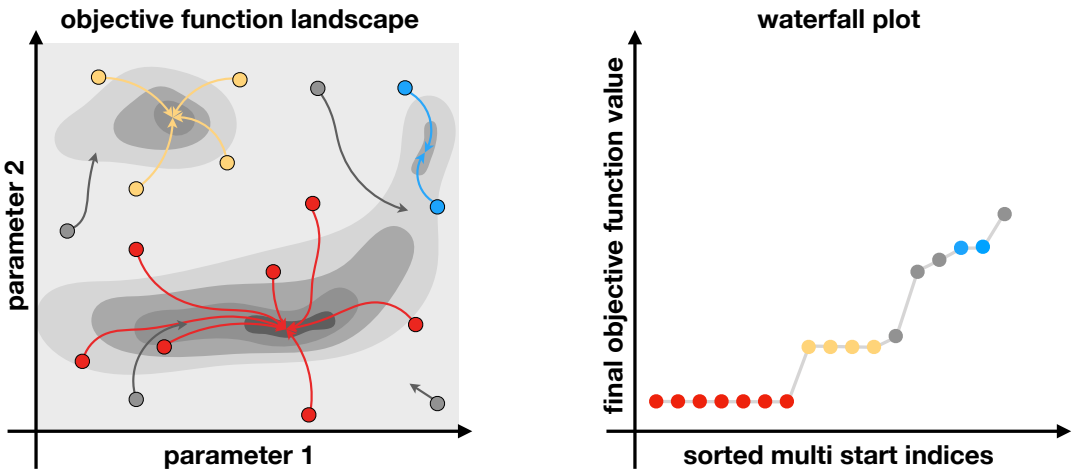

Figure 2.2: Multi-start local optimization. Left panel: Example of an objective function landscape with two parameters. Randomized initial points are drawn and local optimization are started to find the closest minimum. Right panel: Results of the sorted local optimizations visualized in a waterfall plot. The best local optimum, which was found, is assumed to be the global optimum.

motivated ODE models [Schälte et al., 2018].

Computing derivative information

Currently, three approaches are commonly used to compute objective function derivatives of ODE models, such as its gradient or its Hessian: finite difference schemes, forward sensitivity analysis, and adjoint sensitivity analysis.

Finite differences approximate the slope of a tangent by the slope of a secant. Therefore, they are straightforward to implement and can be used for derivative computation of any black-box function. However, finite differences require the choice of a step-size, which strongly influences the accuracy of the computed derivative. As the accuracy of especially the gradient is crucial for local optimization techniques to work, finite differences should only be used as last resort if more accurate methods are not available, or if a good step size can be fixed a priori [Raue et al., 2013b, Schälte et al., 2018].

The second method, forward sensitivity analysis [Leis and Kramer, 1988, Raue et al., 2013b], is a semi-analytical approach for gradient computation, and many state-of-the-art computational toolboxes rely on it [Kaschek et al., 2019, Raue et al., 2015]. When differentiating Equation 2.12 with respect to θ_r, the gradient is obtained:

$$\frac{\partial J}{\partial \theta_r} = -\sum_{k=1}^{n_e}\sum_{i=1}^{n_y^k}\sum_{j=1}^{n_t}\left(\frac{\bar{y}_{ij}^k - h_i(t_j,u^k)}{\left(\sigma_{ij}^k\right)^2} s_r^{y_i}(t_j,u^k) + \left(\frac{\left(\bar{y}_{ij}^k - h_i(t_j,u^k)\right)^2}{\left(\sigma_{ij}^k\right)^3} - \frac{1}{\sigma_{ij}^k}\right)\frac{\partial \sigma_{ij}^k}{\partial \theta_r}\right) \tag{2.17}$$

in which $s_r^{y_i}$ denotes the sensitivity, i.e., the derivative, of observable y_i with respect to parameter θ_r. The observable sensitivities are calculated from the state sensitivities s_r^x:

$$s_r^{y_i}(t_j,u^k) = \frac{dy_i}{d\theta_r}(t_j,u^k) = \nabla_x h_i(t_j,u^k) s_r^x(t_j,u^k) + \frac{\partial h_i}{\partial \theta_r}(t_j,u^k) \quad \text{with } s_r^x(t_j,u^k) = \frac{\partial x}{\partial \theta_r}(t_j,u^k) \tag{2.18}$$

The state sensitivities need to be computed by integrating the corresponding ODE, which is

obtained from differentiating Equation 2.1:

$$\frac{d}{dt}s_r^x(t,u^k) = \Big(\nabla_x f\big(x(t,u^k),u^k\big)\Big)\, s_r^x(t,u^k) + \frac{\partial f}{\partial \theta_r}\big(x(t,u^k),u^k\big)\big) \tag{2.19}$$

In forward sensitivities analysis, the error in the state sensitivities can be controlled together with the error of the state variables when integrating both ODEs (2.1) and (2.19) together, which allows accurate gradient computation [Hindmarsh et al., 2005]. However, forward sensitivities analysis requires solving an ODE of the size $n_x(n_\theta + 1)$. Hence, the computational effort of this method scales linearly in the number of parameters and in the number of state variables, which is computationally demanding for large n_x and n_θ.

The third method, adjoint sensitivity analysis [Fröhlich et al., 2017a, Sengupta et al., 2014], circumvents the integration of the state sensitivities. In this approach, only the original ODE system (2.1) is integrated forward in time and subsequently an ODE for the so called adjoint state $p(t)$ is integrated backward in time, starting at t_{n_t}:

$$\frac{d}{dt}p\big(t,x(t,u^k),u^k\big) = -\Big(\nabla_x f^T\big(x(t,u^k),u^k\big)\Big)\, p(t,u^k) \tag{2.20}$$

$$p(t_{n_t},u^k) = \sum_{i=1}^{n_y^k} \nabla_x h_i^k\big(x(t_{n_t},u^k),u^k\big)\frac{\bar{y}_{in_t}^k - h_i^k(t_{n_t},u^k)}{(\sigma_{in_t}^k)^2} \tag{2.21}$$

For time-discrete data, $p(t)$ has to be reinitialized at each measurement time-point:

$$p(t_j,u^k) = \lim_{t\to t_j^+} p(t,u^k) + \sum_{i=1}^{n_y^k} \nabla_x h_i\big(x(t_j,u^k),u^k\big)\frac{\bar{y}_{ij}^k - h_i(t_j,u^k)}{(\sigma_{ij}^k)^2}. \tag{2.22}$$

In the end, the gradient is given as:

$$\frac{\partial J}{\partial \theta_r} = -\sum_{k=1}^{n_e}\left(\sum_{i=1}^{n_y^k}\sum_{j=1}^{n_t}\frac{\bar{y}_{ij}^k - h_i(t_j,u^k)}{(\sigma_{ij}^k)^2}\frac{\partial h_i}{\partial \theta_r}(t_j,u^k) + \left(\frac{\big(\bar{y}_{ij}^k - h_i(t_j,u^k)\big)^2}{\big(\sigma_{ij}^k\big)^3} - \frac{1}{\sigma_{ij}^k}\right)\frac{\partial \sigma_{ij}^k}{\partial \theta_r}\right. \tag{2.23}$$

$$\left. + \int_{t_0}^{t_{n_t}} p^T(t,u^k)\frac{\partial f}{\partial \theta_r}(t,u^k)dt + p(t_0,u^k)^T\frac{\partial x_0}{\partial \theta_r}(u^k)\right) \tag{2.24}$$

To compute this expression, n_θ one dimensional quadratures have to be calculated during backward integration. In practice, these quadratures are typically computationally less expensive, so the linear dependence of the computation time on n_θ for adjoint sensitivity analysis can be considered to be weak, as pointed out in [Özyurt and Barton, 2005]. This yields the gradient for little more than the cost of integrating two differential equations of the size n_x. For large-scale ODE models with many parameters, this approach has shown to be the most efficient method for gradient computation [Fröhlich et al., 2017a, Kapfer et al., 2019, Sengupta et al., 2014] and subsequently for local optimization [Fröhlich et al., 2017a, Fröhlich et al., 2018a, Villaverde et al., 2018].

2.2.2 Reliability of parameter estimates: Uncertainty analysis

Independent of the paradigm for parametric modeling – frequentist or Bayesian – finding the best parameter vector is not enough. An assessment of uncertainties is crucial to know how trustworthy model predictions are. These uncertainties arise due to measurement noise on one hand, and due to limited observability of the system on the other hand: Typically, only a fraction of the system state is assessable through observations [Raue et al., 2010]. Beyond measurement noise and limited observability of the system, also further sources of uncertainties exist, such as, e.g., uncertainties in the model formulation, or biases in the measurement data. However, as these phenomena need to be assessed by different, particular methods, we will neglect them in the following.

Observability, identifiability and uncertainty

Observability is concerned with the question, whether the system state x can be inferred from the observable quantities y. We call a system observable if x can be inferred from y [Chis et al., 2011, Raue et al., 2009], or, more heuristically, if the system state could be inferred in the case of time-continuous noise-free measurements. Observability is linked to the concept of structural identifiability: A model is called structurally identifiable, if θ can be uniquely determined from the model output y of all considered experimental conditions [Chis et al., 2011, Ligon et al., 2018]. It is important to note that this definition explicitly takes into account experimental conditions and observables but no measurement data. Hence, structural identifiability is concerned with the question whether the model parameters can be determined in theory from the model outputs in the case of time-continuous noise-free measurements [Raue et al., 2009, 2013a]. Going beyond structural identifiability, practical identifiability is concerned with the question whether the model parameters can be determined in practice from a specific dataset D. Practical identifiability takes into account the data, the noise model, the prior knowldge (if applicable), and the likelihood function, and it is defined by fixing a confidence threshold: If the confidence interval of a certain parameter to the fixed confidence threshold does not intersect with the boundary of the box Ω, the parameter is called practically identifiable. Otherwise, it is called practically non-identifiable [Raue et al., 2009]. Identifiability can be more generally defined for any model property. A model property can be any function of the model parameters:

$$g\colon \Omega \longrightarrow \mathbb{R}, \quad \theta \longmapsto g(\theta) \tag{2.25}$$

It follows from these definitions that practical identifiability implies structural identifiability. Ideally, structural identifiability analysis is carried out before performing parameter estimation, as it may simplify the parameter optimization [Raue et al., 2013a]. But most computational methods for structural identifiability analysis suffer from drawbacks such as applicability only to special cases or limited scalability to larger systems. For this reason, it is sometimes disregarded in practice and only practical identifiability analysis is carried out after parameter optimization [Raue et al., 2009, 2013a].

Many methods for practical identifiability analysis exist which aim at different degrees of accuracy [Ballnus et al., 2017, Joshi et al., 2006, Kreutz, 2018, Kreutz et al., 2013, Villaverde et al., 2019]. Often, very accurate methods which try to give insights about the whole posterior distribution in detail are computationally very expensive and thus not applied to large-scale models [Ballnus et al., 2017, Kreutz et al., 2013]. In these cases, less accurate and more approximative methods tend to be used [Kapfer et al., 2019, Kreutz, 2018, Villaverde et al., 2019]. Usually, each

method has a different aim concerning the insight which it provides about the model at hand. In this thesis, uncertainty analysis is used as a summarizing term for these different methods of (practical) identifiability analysis, most of which also allow an answer about the structural identifiability of the model parameters or properties.

Local approximations

If a normal or log-normal noise model is used, the the parameter distribution around the global optimum can be approximated by a second order Taylor expansion:

$$J(\theta^{MAP} + \Delta\theta) \approx J(\theta^{MAP}) + \frac{1}{2}\Delta\theta^T H(\theta^{MAP})\Delta\theta \tag{2.26}$$

This requires the Hessian $H(\theta^{MAP})$ of the objective function or an approximation thereof. From this approximation of the objective function, the likelihood function $\mathcal{L}$ can be approximated as normal distribution with mean θ^{MAP} and covariance matrix H^{-1}. Based on this approximation, confidence intervals can be drawn (symmetrically) around θ^{MAP} [Fisher, 1922].

In most cases, using this type of local approximation is computationally cheap and hence also achievable for large-scale models [Kapfer et al., 2019]. However, the quality of these approximations strongly depends on how well the local structure of J around θ^{MAP} corresponds to the global structure of J [Joshi et al., 2006]. Hence, local approximations should only be used as last resort, if more accurate methods are not applicable due to their (usually higher) computational effort [Kapfer et al., 2019, Villaverde et al., 2019]. Nevertheless, they can give important first hints about the most or least identifiable directions in parameter space.

Profile likelihood analysis

A more accurate but also computationally more expensive method consists in using profile likelihoods. Profile likelihood (or short profile) calculation, which has been introduced in a systems biology context in [Raue et al., 2009], is a common method to assess these uncertainties [Kreutz et al., 2013]. It has shown to be a robust method for uncertainty analysis also in the presence of structural non-identifiabilities, a situation which poses problems to other methods [Fröhlich et al., 2014, Raue et al., 2013a]. A profile is the maximum projection of the likelihood to a chosen parameter axis: for $\theta_r, r \in \{1, \ldots, n_\theta\}$, the profile value at $\theta_r = c$ this given by

$$\mathrm{PL}_{\theta_r}(c) = \max_{\substack{\theta_r = c \\ \theta \in \Omega}} \mathcal{L}_{\mathcal{D}}(\theta). \tag{2.27}$$

Profiles have to be computed separately for each parameter $\theta_r, r = 1, \ldots, n_\theta$, which causes a linear scaling of the computational effort with the number of parameters. Moreover, although different methods for profile calculation exist, the calculation of a profile is computationally substantially more expensive than computing a local approximation for all model parameters. Although profiles can also be computed directly for model properties [Kreutz et al., 2012], which is convenient if there are less model properties than parameters, using profiles remains computationally expensive. However, profiles provide substantial and more than local insight into the model [Raue et al., 2013a].

Parameter sampling

Following a different goal than local approximations and profiles, methods exist which aim to generate a representative sample from the posterior distribution directly, but which are computational much more expensive [Ballnus et al., 2017, Hug et al., 2013]. Among these approaches, rejection and importance sampling are the most simple methods, but suffer from conceptional problems, such as low acceptance rates, for high dimensional parameter spaces. More commonly used and more robust are Markov chain Monte Carlo (MCMC) methods [Neal, 2011]. Here, the a Markov chain is constructed, which has the posterior distribution as stationary distribution. The first algorithm, which followed this idea, was the Metropolis-Hastings (MH) algorithm [Hastings, 1970, Metropolis et al., 1953], which generates new sample points from a proposal distribution and then accepts or rejects these points, with a probability which depends on their objective function values. Adaptations of the MH algorithm have been proposed, which either improve the proposal distribution [Haario et al., 2001, Lacki and Miasojedow, 2015], use multiple Markov chains such as parallel tempering [Ballnus et al., 2018, Miasojedow et al., 2013, Vousden et al., 2016] or parallel hierarchical sampling [Rigat and Mira, 2012], or inform the the process with derivatives of the objective function, such as the Metropolis-adjusted Langevin algorithm [Girolami and Calderhead, 2011] or Hamilton Monte Carlo sampling [Graham and Storkey, 2017, Hoffman and Gelman, 2014].

If sufficiently many such Markov chains are run long enough, the sampling algorithm it will eventually find the stationary distribution and hence generate a representative sample. Convergence proofs for these methods exist, but in practice, it is hard to check whether the chain has converged to its stationary distribution or is just stuck in a certain region of the parameter space [Ballnus et al., 2017]. Hence, among the discussed methods for uncertainty analysis, MCMC methods may yield the most possible insight into the posterior distribution, but are difficult to assess in terms of quality and computationally the most expensive among the discussed approaches [Ballnus et al., 2017].

Given a sample from the posterior distribution, predictions and model properties can be computed using all the parameter vectors from the sample [Raue et al., 2013a]. This allows to incorporate information beyond using just the best parameter vector when studying the system.

Ensembles from parameter optimization

Recently, ensemble methods have been proposed, which try to approximate a sample from the posterior distribution in a simplified way by a so called ensemble [Henriques et al., 2017, Villaverde et al., 2019]. These approximations should not be confounded with actual samples, as they don't aim to accurately represent the posterior distribution. Such an ensemble can be taken from the results of a multi-start local optimization and may take into account the history of each local optimization run [Villaverde et al., 2019]. No guarantees exist that such an ensemble will represent the parameter distribution accurately, but it provides upper bounds for the value of the objective function (and hence lower bounds for the likelihood value) at different points in parameter space. Given the fact that this information is available from parameter optimization without additional computational effort, it can be a valuable resource especially for large-scale models, for which more accurate approaches such as MCMC sampling or profile calculation are computationally prohibitive [Villaverde et al., 2019].

Also in this case, the ensemble can be used similar to an MCMC sample to generate predictions or compute model properties by taking into account all the parameter vectors from the ensemble. In general, this yields different predictions than using only the best parameter vector

and can hence be seen as more Bayesian approach of using a computational model.

Further methods

Beyond local approximations, profile likelihood, and sampling methods, other approaches have been proposed to quantify parameter and prediction uncertainties of ODE models: For example, bootstrapping [Joshi et al., 2006] is commonly used to assess parameter uncertainties and also implemented in computational toolboxes [Balsa-Canto and Banga, 2011]. However, it has been shown that bootstrapping can yield misleading results in the presence of structural non-identifiabilities [Fröhlich et al., 2014].

For a Bayesian point of view, other sampling strategies than Markov chain Monte Carlo methods exist, such as sequential Monte Carlo methods [Doucet et al., 2000, Robert and Casella, 2004]. To gain additional insight about parameter and predictions uncertainties, sampling methods may be combined with (global) sensitivity analysis [Eriksson et al., 2018]. However, methods based on importance sampling/resampling suffer from low acceptance rates in high dimensions, which prohibits their use for large-scale models. Furthermore, methods which assess the probability density function of the state variables and the model parameters based on partial differential equations (PDE) were developed for uncertainty quantification with controllable error [Weiße and Huisinga, 2011, Weiße et al., 2010]. However, these methods require additional information on the initial probability distribution of the state variables, which is often not available.

Most of the applications, on which this thesis focuses, are either high-dimensional or possess strong non-identifiabilities, as the key goal of this thesis is the development of scalable and efficient computational methods for large-scale models. Hence, we will mostly rely on ensemble methods an, where possible, profile likelihood analysis in the following.

Chapter 3

Mini-batch optimization for ODE models

In the previous chapter, we have seen that ODE models need to be trained with measurement data to yield meaningful predictions. Modern implementations of efficient algorithms have led to a drastic reduction of computation time and made it possible to carry out parameter estimation for many biological ODE models on usual laptops within minutes or hours [Mendes et al., 2009, Raue et al., 2015]. However, this is typically the case for comparably small models, which describe rather isolated signaling pathways or explain single features of a system [Becker et al., 2010, Swameye et al., 2003].

In some cases, the study of certain biological systems can not be reduced to one or maybe not even to a few signaling pathways, as, e.g., pathway crosstalk may be an essential feature, or the study goal is to describe a whole system with all of its parts. Typical examples would be understanding and predicting the effects of drugs and drug combinations in cancer cells [Fröhlich et al., 2018a, Hass et al., 2017] or understanding the functioning of whole cells [Babtie and Stumpf, 2017, Karr et al., 2012].

For these large-scale models, which result from studying such systems, even the most sophisticated mathematical methods are sometimes insufficient to constrain the computational burden to a tractable amount [Kapfer et al., 2019]. Offloading the work to a computing cluster and massively parallelizing the computational effort allows to reduce the waiting time for the modeler, but the accumulated computation time may still exceed ten or hundred thousand of hours [Fröhlich et al., 2018a, Schmiester et al., 2019a]. This is especially the case, if not only the ODE systems itself is large, but also many experimental conditions have to be simulated. As the number of data points, which are necessary to constrain the model parameters, typically grows with the size of the model [Aldridge et al., 2006], these two effects rarely come alone, but mostly together.

On one hand, the computational effort for solving the underlying ODE of a parameter estimation problem is expected to scale with the size of the ODE system, i.e., the number of state variables. On the other hand, the computational effort scales linearly in the number of experimental conditions. Methods for reducing the computation time for large-scale ODE systems have been developed since a long time and implementations of ODE solvers are already extremely efficient [Kapfer et al., 2019, Maiwald and Timmer, 2008]. Hence, it is likely to be easier to reduce the computation time by attempting to break the linear scaling in the number of experimental conditions.

In other scientific fields, especially in the field of deep learning, which focuses on neural

nets with hidden layers, data sets for model training may be vast [Goodfellow et al., 2016, Janowczyk and Madabhushi, 2016]. For these neural nets, stochastic gradient descent or mini-batch optimization methods are typically applied in parameter estimation [Abadi et al., 2015, Goodfellow et al., 2016]. It was shown that these methods may indeed break the linear scaling with the number of experimental conditions and hence reduce computation time substantially [Wilson and Martinez, 2003]. However, none of the common implementations of these mini-batch optimization techniques focuses on ODE models. A direct method transfer from deep learning to ODE models is hence unlikely to work right away, as the typical optimization tasks in both fields have their particular challenges.

In this chapter, we discuss a scalable parameter estimation method for large-scale datasets, which is based on adaptations of mini-batch optimization to parameter estimation of ODE models. An implementation thereof into an efficient and massively parallelizable computational framework, written in C++, was developed as part of this thesis. First, algorithmic adaptations to make mini-batch optimization better suited for optimization problems based on ODE-models are presented. Then, an optimization benchmark study on small- to medium-scale models with artificially created data is carried out. Subsequently, the gained knowledge is transferred to a large-scale ODE model, which is trained on a dataset of real measurements with 13,000 experimental conditions, which is an unprecedented scale for ODE models [Fröhlich et al., 2018a]. We show that our implementation reduces the computation time for model training by more than an order of magnitude while providing better optimization results when compared to established approaches. In a last step, we use our results for comparing ensemble models with point estimates for this large-scale model and show that ensemble models tend to outperform the classically used point estimates in this case.

This chapter is based on my work in the following publication and hence parts of it may be similar to it:

- **Stapor, P.**, Schmiester, L., Wierling, C., Lange, B., Weindl, D., Hasenauer, J. (2019). Mini-batch optimization enables training of ODE models on large-scale datasets. bioRxiv, 10.1101/859884, *Under review*

3.1 Background: Parameter estimation for large-scale models

Models in systems biology have been steadily growing and methods for parameter estimation were constantly improved to keep up with problem sizes [Egea et al., 2007, Fröhlich et al., 2017a, Raue et al., 2013b, Schmiester et al., 2019a]. Yet, with soaring availability of measurement data through public databases [Barretina et al., 2012, Cancer Genome Atlas Network, 2012, Eduati et al., 2017, Li et al., 2017] and increasing capacity of computing clusters, parameter estimation of large-scale models has gained importance in recent years. For this reason, a brief overview about state-of-the-art methods for parameter estimation of large-scale models is given in this section.

3.1.1 Existing methods for large-scale ODE models and challenges

Available methods for parameter estimation of large-scale models

The first step for making parameter estimation tractable for large-scale models consists of using an efficient optimization approach. So far, different studies have shown that employing multi-start local or scatter search techniques with sophisticated local optimization techniques based

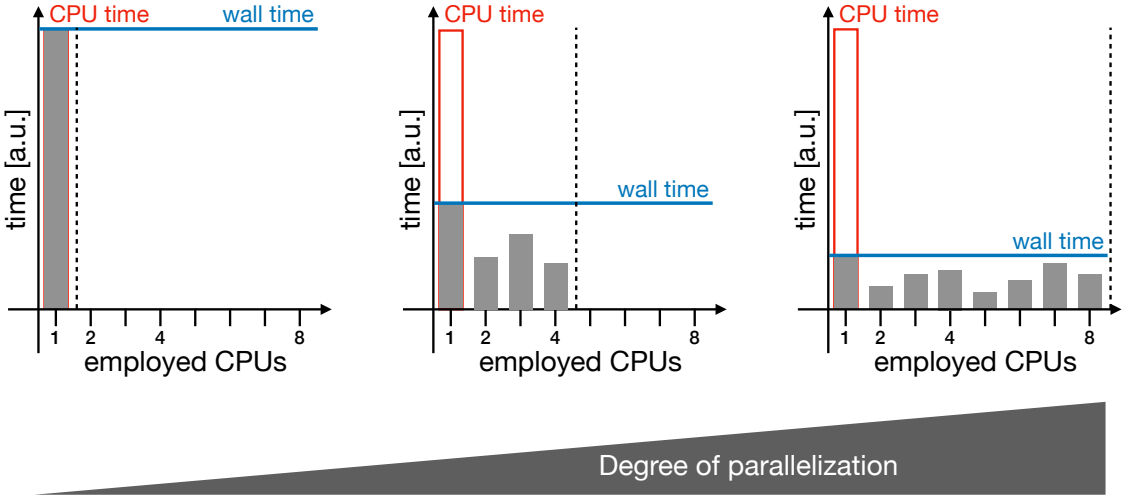

Figure 3.1: Visualization of wall time vs. (accumulated) CPU time. All three diagrams show the same CPU time, but different wall times. Wall time describes the waiting time for the user until a result is obtained and can hence be reduced by distributing tasks onto multiple CPUs. CPU time is the total computation time, which was used by all CPUs which were employed to obtain the result.

on accurately computed gradients are most efficient for ODE models [Raue et al., 2013b, Schälte et al., 2018, Villaverde et al., 2018]. Computing the necessary derivatives of the objective function is most efficiently carried out by adjoint sensitivity analysis [Fröhlich et al., 2017a, Sengupta et al., 2014]. Although forward sensitivity analysis allows to construct an approximation to the Hessian of the objective function, which is often beneficial in optimization, adjoint sensitivity analysis, which does not yield a Hessian approximation by itself, outperforms forward sensitivity analysis for sufficiently large models [Fröhlich et al., 2017a, Villaverde et al., 2018].

It has furthermore been shown that exploiting the structure of the objective function by hierarchical optimization allows a substantial reduction of computation time [Loos et al., 2018a, Weber et al., 2011]. This approach can also be combined with adjoint sensitivity analysis, which ensures efficiency for all model sizes [Schmiester et al., 2019a].

Moreover, parallelized implementations allow either a distribution of the workload over multi-starts or even over simulations of experimental conditions in objective function evaluations to different degrees [Penas et al., 2015, Raue et al., 2015, Schmiester et al., 2019a]. In scatter search, asynchronous updates of the reference set have shown to outperform synchronous updates and allow a better exploitation of parallel computing structures [Penas et al., 2015]. Hence, parallel implementations help to enable parameter estimation for large-scale models.

Challenges for large-scale models

Despite the presented advances of computational methods in recent years, the necessary computation time for parameter estimation remains a major bottleneck in computational and systems biology [Kapfer et al., 2019, Kreutz, 2016, 2019], as the presented advances allowed the development of larger and larger models.

Although parallelization allows to reduce the wall time, i.e., the waiting time for the modeler until a parameter estimation result is obtained, it does not reduce the overall computation time (Figure 3.1). Moreover, massive parallelization is only possible on large computing clusters, which restricts the applicability of these methods to researchers which have access to such clus-

ters. Moreover, as scientific computation toolboxes, which employ sophisticated parallelization techniques, often require specific knowledge from the researcher, they tend to be less used in the community. Beyond these limits, distributing computation time on multiple CPUs is only possible to a certain level, before either parallelization becomes inefficient due to the communication overhead between the CPUs [Gustafson, 1988], or as certain algorithms can only be scheduled in parallel to a limited degree [Amdahl, 1967]. As finally, high computation times lead to high energy consumption, they also raise ecological questions. For these reasons, the aim should be to reduce computation time rather than wall time.

Another problem consists of the fact that published large-scale models suffer from further problems than just being time consuming to parametrize. Often, the parameters of these models are poorly identifiable and the objective function is only weakly constrained in many directions in parameter space [Fröhlich et al., 2018a, Kapfer et al., 2019]. In such a scenario, the probability mass of the posterior distribution is not located sharply around a global optimum, but likely to be spread over a large area in parameter space, as it can be shown for smaller models [Raue et al., 2013a]. Unfortunately, the corresponding computational methods to assess rigorously whether or not the probability mass is indeed spread, such as profile likelihood computation or MCMC sampling, are not applicable to these models due to their high computational cost [Kapfer et al., 2019]. Hence, firstly, optimization methods for large-scale models have to cope with many poorly constrained directions in parameter space, which may hamper optimizer convergence [Maiwald et al., 2016], and secondly, it is unclear how well the maximum a posteriori estimate describes the dynamics of the modeled system, even if parameter estimation can be carried out [Maier et al., 2020, Murphy, 2012]. In consequence, beyond reducing computation time, it is important to also assess the appropriateness of the commonly used point estimates.

3.1.2 Mini-batch optimization methods for neural nets and challenges

Principle of mini-batching

In the field of deep learning, where also gradient-based local optimization methods are used [LeCun et al., 2002, Martens, 2010, Rumelhart et al., 1986, Sutskever et al., 2013], model training is often performed on vast datasets, requiring many independent model evaluations [Janowczyk and Madabhushi, 2016, LeCun et al., 1998]. The problematic scaling behavior with respect to the size of the dataset is addressed by mini-batch optimization [Goodfellow et al., 2016, Sutskever, 2013, Wilson and Martinez, 2003]: Classical (full-batch) optimization methods evaluate all contributions to the objective function, i.e., simulate all experimental conditions, in each iteration of parameter optimization (Figure 3.2 A). In contrast, mini-batch optimization methods evaluate only the contribution to the objective function coming from a randomly chosen subset of experimental conditions, a mini-batch, in each step [Goodfellow et al., 2016, Robbins and Monroe, 1951]. Thus, the model is only evaluated on a fraction of the dataset per optimization step, which reduces the computation time [Goodfellow et al., 2016, Wilson and Martinez, 2003] (Figure 3.2 B).

More precisely, let m be the current optimization step, then a mini-batch $S_m \subseteq \{1, \ldots, n_e\}$ is drawn at random, and estimates of the objective function and its gradient are computed only based on S_m. These estimates of the objective function and its gradient in optimization step m

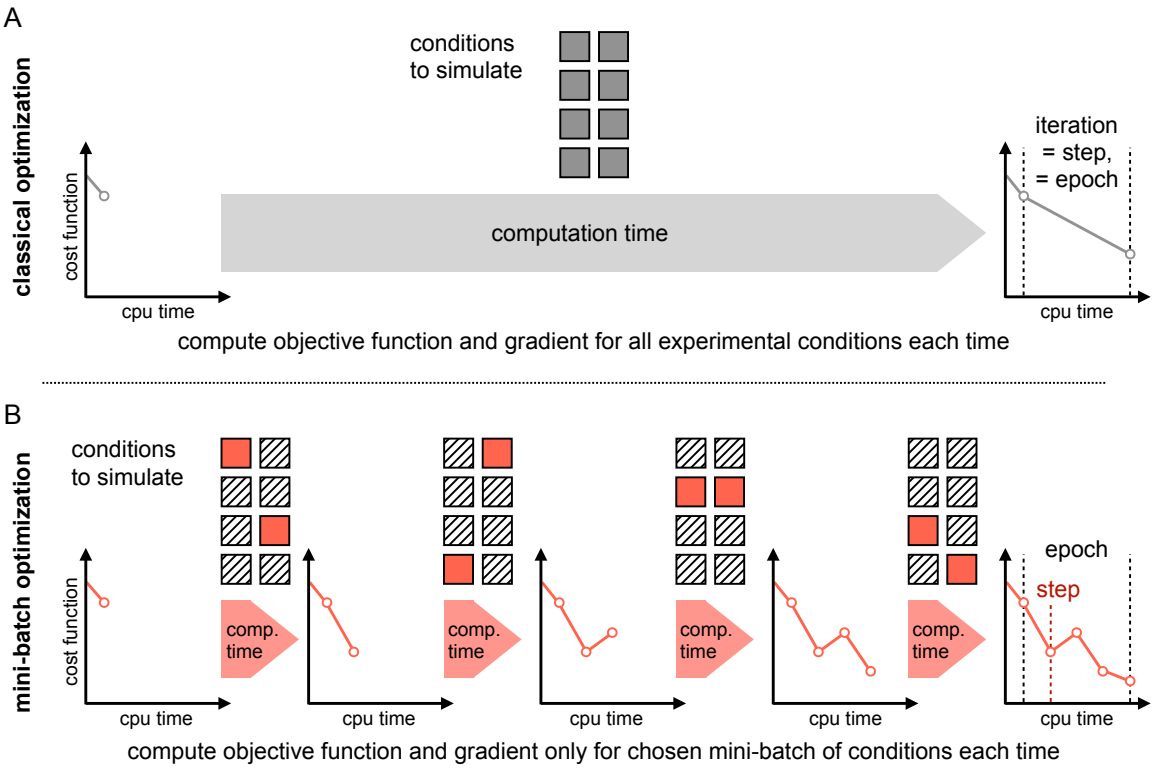

Figure 3.2: Visualization of full-batch and mini-batch optimization. **A** Classical full-batch optimization methods evaluate the contribution of all data points – and thus all experimental conditions – to the objective function in each step. The computation time scales linearly with the number of independently evaluable experimental conditions. **B** In mini-batch optimization, the independent experimental conditions are randomly divided into disjoint subsets, the mini-batches. Per optimization step, only the contribution of the chosen mini-batch is evaluated. Hence, possibly many optimization steps can be performed during one epoch, which is the time needed to evaluate the objective function on the whole dataset. This figure is adapted from Figure 1 of the author's publication [Stapor et al., 2019].

are given as:

$$J_m(\theta) = \sum_{k \in S_m} \underbrace{\frac{1}{2} \sum_{i=1}^{n_y^k} \sum_{j=1}^{n_t} \left(\left(\frac{\bar{y}_{ij}^k - y_i(t_k, \theta, u^k)}{\sigma_{ij}^k} \right)^2 + \log\left(2\pi \left(\sigma_{ij}^k\right)^2\right) \right)}_{=J^k(\theta)} = \sum_{k \in S_m} J^k(\theta) \tag{3.1}$$

$$\nabla_\theta J_m(\theta) = \ldots = \sum_{k \in S_m} \nabla_\theta J^k(\theta) \tag{3.2}$$

It is important to note that no experimental conditions are drawn twice until the full dataset has been evaluated, a time-frame which is called one epoch. Hence, the mini-batches belonging to one epoch are disjoint and form a partition of the dataset. Theoretically, also the inner sum structure (over the indices i and j) could be used for mini-batching. However, when working with ODE models, the computationally demanding part is the solution of the initial value problem for an experimental condition k. Once this is done, evaluating the different observables y_i at different time points t_j happens in negligible time. For this reason, mini-batching will only make sense over the conditions k for most ODE model applications.

In summary, mini-batch optimization is a particular paradigm of local optimization and a

priori only capable of finding local optima [Robbins and Monroe, 1951]. It allows to perform more – but less informed – optimization steps than classical full-batch approaches in the same computation time. Overall, it is likely that more optimization steps, but (hopefully) less epochs will be needed when compared to full-batch optimization.

Mini-batch optimization algorithms in a nutshell

The way how the parameter update is executed in each optimization step, i.e., how $\theta^{(m+1)}$ is computed from $\theta^{(m)}$ and the estimated gradient $\sum_{k \in S_m} \nabla_\theta J_k(\theta^{(m)})$, depends on the chosen optimization algorithm. Some of the most commonly used algorithms which we investigate closer in this chapter are:

- Vanilla stochastic gradient descent (SGD) [Robbins and Monroe, 1951], which is the simplest algorithm, using only the negative gradient of the objective function as update direction.

- Stochastic gradient descent with momentum [Polyak, 1964, Sutskever et al., 2013], a common variant, which uses a decaying average of negative gradients as direction instead of the negative objective function gradient alone.

- RMSProp [Tieleman and Hinton, 2012], a so-called adaptive algorithm, which rescales the current gradient by a decaying average of root-mean-squares over the previous objective function gradients.

- Adam [Kingma and Ba, 2015], another adaptive algorithm, which attempts to combine the benefits of RMSProp with the momentum approach by using two decaying averages.

A well-written summary of these and further mini-batch optimization algorithms can be found in [Goodfellow et al., 2016].

Despite their different approaches, all the mentioned mini-batch algorithms have some features in common. First, they are not guaranteed to – and, in fact, don't even try to – produce a series of parameter vectors, along which the objective function decreases monotonically. Hence, it is to be expected that the progress which is made in one optimization step can be (at least partly) undone in another, subsequent step. This is a striking difference to most local full-batch optimization algorithms, which at least try to or can even guarantee to produce monotonically decreasing trajectories of objective function values. Over the whole optimization process however, most of mini-batch optimization methods are guaranteed to converge to a local minimum in probability. Second, all of these mini-batch optimization algorithms rescale the proposed optimization step with a factor η, called the learning rate. If we assume a given algorithm in optimization step m to produce a parameter update $d(\theta^{(m)})$, then the next proposed parameter vector will be

$$\theta^{(m+1)} = \theta^{(m)} + \eta_m \cdot d(\theta^{(m)}), \tag{3.3}$$

with η_m being the learning rate at step m. Obviously, η affects the step size of the optimizer. But as for most algorithms $\|d(\theta^{(m)})\| \neq 1$, it is not identical to the step size. In many publications, the learning rate is either fixed to a previously chosen value, such as 10^{-3} [Ruder, 2016], which is a commonly used value for training deep neural nets, or prescheduled over the optimization process [Goodfellow et al., 2016].

Challenges in the transfer of existing algorithms to ODE models

Sophisticated implementations of many mini-batch optimization algorithms are available in state-of-the-art toolboxes for neural nets, such as Tensorflow [Abadi et al., 2015]. Conceptually, these frameworks can be employed to mimic simple ODE solver schemes, e.g., a forward Euler integration, such as done in [Yuan et al., 2019]. However, ODE models in systems biology typically exhibit stiff dynamics, which makes it necessary to employ implicit solvers with adaptive time stepping [Klipp et al., 2005, Mendes et al., 2009, Raue et al., 2013b], such as, e.g., a BDF scheme. Unfortunately, there is no straight-forward way of extending the approach presented in [Yuan et al., 2019] to adaptive time-stepping.

Approaching the problem from the other side and applying mini-batch optimization algorithms directly on parameter estimation of ODE models is more promising but also not straight forward: When estimating parameters of ODE models, it is a common problem that the trajectory of the optimizer passes through regions in parameter space in which numerical integration of the ODE fails [Chung et al., 2017b]. Such failure can occur for several reasons, but the main reasons are either the step size of the solver falling below numerical precision, resulting in exceeding the maximum number of allowed steps for ODE integration, or diverging expressions in the right hand side, causing infinities or not-a-number values in the state variables [Serban and Hindmarsh, 2005]. By default, mini-batch optimization algorithms are not designed to cope with these issues.

Another problem is hyperparameter tuning: Most local optimization methods, also mini-batch techniques, crucially depend on hyperparameters, which have to be set prior to parameter estimation. For full-batch optimization methods such as BFGS [Goldfarb, 1970] with trust-region-reflective [Coleman and Li, 1996] or interior-point algorithms [Wächter and Biegler, 2006], many hyperparameters are associated with stopping conditions and at least good rules-of-thumb exist for their choice. For mini-batch optimization, the success of optimization is governed by different hyperparameters, such as the mini-batch size, the learning rate, the optimization algorithm or algorithm-dependent tuning parameters. So far, the influence of these hyperparameters on parameter optimization of ODE models has not been studied at all and good choices first have to be found.

3.1.3 Theoretical results and practical application of mini-batch optimization

Many theoretical results on the convergence of stochastic gradient descent or derived mini-batch optimization algorithms exist [Duchi et al., 2011, Gower et al., 2019, Kingma and Ba, 2015, Polyak, 1964, Robbins and Monroe, 1951, Schmidt et al., 2017]. However, those publications are concerned with diverse scenarios of mini-batch optimization. In general, mostly two such scenarios can be discerned.

One group of manuscripts [Defazio et al., 2014, Gower et al., 2019, Schmidt et al., 2017] focusses on the situation that a finite sum of functions J^k is optimized:

$$\theta^* = \text{argmin}_{\theta \in \mathbb{R}^{n_\theta}} J(\theta) \quad \text{with } J(\theta) = \frac{1}{K} \sum_{k=1}^{K} J^k(\theta) \tag{3.4}$$

In the setting of this thesis, K corresponds to the number of experimental conditions n_e. In most publications, the J^k are assumed to be smooth and their sum J strongly convex [Defazio et al., 2014, Schmidt et al., 2017]. However, this strong convexity condition on the J can be relaxed, as long as a probabilistic version of Lipschitz continuity is fulfilled by the J^k [Gower et al., 2019]. In

this finite sum setting, it can be shown that, using a fixed learning rate η, vanilla SGD converges in expectation to the proximity of the global minimizer with linear rate of convergence, i.e., there is an $0 < \alpha < 1$ such that in optimization step m [Schmidt et al., 2017]:

$$\mathbb{E}\left[J(\theta_m) - J(\theta^*)\right] = C\alpha^m + \epsilon \tag{3.5}$$

Here, C is a positive constant and $\epsilon > 0$ can be interpreted as the approximation quality, which can be controlled be the learning rate and the mini-batch size [Gower et al., 2019]. Hence, vanilla SGD converges quickly towards the optimal solution, but stops progressing as soon as it has approached the optimum up to ϵ. Interestingly, if a decreasing learning rate scheme is employed, such as $\eta_m = 1/m$, then this ϵ-noisiness in the optimization process can be avoided and vanilla SGD converges to θ^* in probability. However, it does so with a substantially lower rate of convergence, which will be

$$\mathbb{E}\left[J(\theta_m) - J(\theta^*)\right] \leq C\frac{1}{\sqrt{m}}, \quad \text{for some} \quad C > 0 \tag{3.6}$$

if J is convex and

$$\mathbb{E}\left[J(\theta_m) - J(\theta^*)\right] \leq C\frac{1}{m}, \quad \text{for some} \quad C > 0 \tag{3.7}$$

if J is strongly convex [Goodfellow et al., 2016].

Another group of publications [Duchi et al., 2011, Kingma and Ba, 2015, Polyak, 1964, Robbins and Monroe, 1951, Tieleman and Hinton, 2012] focusses on the situation of an infinite series of convex functions and aims to find the corresponding minimizer. This can be interpreted as an infinite sequence of repeated and independent random experiments [Robbins and Monroe, 1951]. In some of those works, a regret function $R(m)$ is defined, which depends on the optimization step m [Duchi et al., 2011, Kingma and Ba, 2015]. If we consider a sum of m convex functions J^k, $1 \leq k \leq m$, then this regret is given as:

$$R(m) = \sum_{k=1}^{m} \left(J^k(\theta_k) - J^k(\theta_m^*)\right) \tag{3.8}$$

with θ_m^* being the minimizer of $R(m)$. In this situation, the aim is to find a series of parameter vectors θ_k, such that $\frac{1}{m}R(m)$ converges towards 0, which implies that $\theta_k \longrightarrow \theta_m^*$ under some assumptions on the J^k. The general convergence result, which holds for vanilla SGD as well as for more involved algorithms such as momentum based approaches, RMSProp or Adam, is [Goodfellow et al., 2016, Schmidt et al., 2017]:

$$\frac{1}{m}R(m) \leq C\frac{1}{\sqrt{m}}, \quad \text{for some} \quad C > 0 \tag{3.9}$$

In order to achieve this convergence, two assumptions on the learning rate η are needed [Goodfellow et al., 2016, Robbins and Monroe, 1951]:

$$\lim_{m\to\infty} \sum_{m=1}^{\infty} \eta_m = \infty \tag{A1}$$

$$\lim_{m\to\infty} \sum_{m=1}^{\infty} \eta_m^2 < \infty \tag{A2}$$

When analyzing the convergence rates, we see that most mini-batch optimization algorithms converge at a sublinear rate. Hence, they show a worse asymptotical behavior than (full-batch) gradient descent, which achieves a linear rate of convergence [Nesterov, 2013]. However, in practice, methods such as SGD with momentum, RMSProp, or Adam are widely used [Goodfellow et al., 2016, Sutskever et al., 2013], as they progress quickly towards an optimal solution. In this thesis, we are less concerned with the precise asymptotic properties of the optimization algorithms – although we want to ensure that the employed algorithms converge indeed asymptotically – but rather with the question, whether mini-batch optimization yields meaningful results in practical applications for parameter estimation of ODE models at all, which is not clear a priori. We also want to stretch that our applications are markedly different from the settings, in which convergence for mini-batch optimization algorithms is typically proven: We cannot expect our objective functions to be convex (indeed, they are expected to be non-convex and multi-modal). Moreover, we want to reduce the number of necessary optimization steps as much as possible, rather than looking at their asymptotic behavior, as the objective function evaluation is usually computationally costly.

3.2 Contribution: Adapting mini-batching to ODE models

3.2.1 Method adaptations and implementation

As mentioned before, we needed to adapt existing algorithms for a safe use of mini-batch optimization with ODE models. Our first adaptation was a functionality, which prevents local optimization runs from crashing, if the underlying ODE could not be integrated. Since this mechanism only gets active upon failure of the objective function or gradient evaluation, we call it the rescue interceptor. As we used pre-scheduled learning rates throughout the study, we observed after first tests that it makes sense to moreover implement an additional line-search, which is performed in each optimization step. This additional line-search should avoid too high and hence possibly inappropriate step-sizes and hence accelerate convergence in the beginning of a local optimization procedure, if the chosen learning rate was too high.

Rescue interceptor to handle ODE integration failure

We introduce an additional term, the reduction factor β, which is always multiplied to the learning rate and initialized with the value 1. The rescue interceptor is activated, if ODE integration (with gradient computation) fails (Figure 3.3 A): It undoes the last optimization step (but keeps the current mini-batch), multiplies the reduction factor β by r (here: $r = 0.2$, but any number $r < 1$ would do), and proposes a new step, for which the objective function and its gradient are computed. This procedure is repeated until either a parameter vector is found for which ODE integration is possible, or the local optimization run is finally stopped after a maximum number of repetitions. Whenever ODE integration is successful, the reduction factor is mildly increased again by multiplication with c (here: $c = 1.3$, but any number $c > 1$ would do), to at most 1. Hence, a failure of ODE integration will cause reduced step-sizes for some subsequent steps, which mimics the behavior of a trust-region implementation. A detailed pseudo-code of the rescue interceptor is given in Algorithm 1, its impact on the optimization result is assessed in Section 3.2.2.

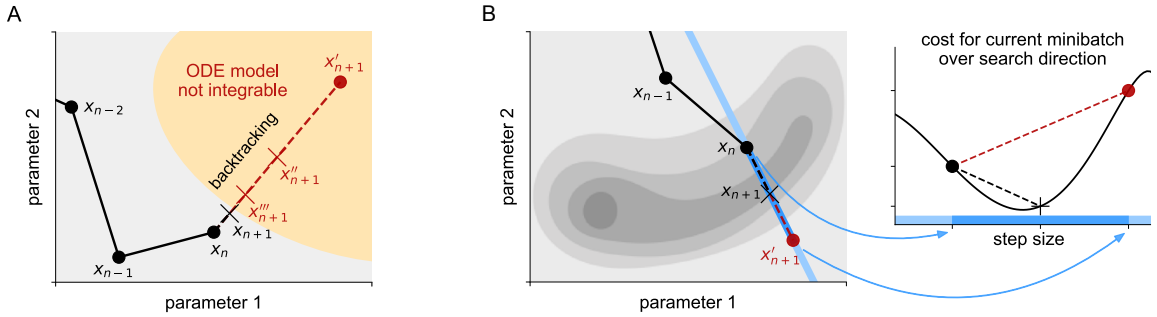

Figure 3.3: Illustration of method adaptations for mini-batch optimization. **A** Schematic of the rescue interceptor, which tries to recover a local optimization run after a failed model evaluation, based on backtracking line-search. **B** Line-search implementation for mini-batch optimizers based on backtracking with fixed mini-batch, which attempts to avoid too high step-sizes to accelerate convergence. This figure is adapted from Figure 3 of the author's publication [Stapor et al., 2019].

Additional line-search functionality to improve optimization for high learning rates

The additional line-search works independently of the rescue interceptor and is based on the interpolation method, described in of [Nocedal and Wright, 2006, Chapter 3]. It uses an additional reduction factor $0 < \gamma \leq 1$, which is multiplied to the learning rate analogously to the reduction factor β of the rescue interceptor. After a parameter update is proposed, the objective function is re-evaluated without gradient – for the new parameter vector, but on the same mini-batch – and checked for improvement (Figure 3.3 B). If the new step yields an improvement, it is accepted, otherwise a backtracking line-search is performed according to the mentioned interpolation algorithm, by shrinking the reduction factor γ. In contrast to the rescue interceptor, this step-size reduction only applies to the current optimization step and the reduction factor is again initialized with $\gamma = 1$ in the next step. A comparison of this approach to standard mini-batch optimization is given in Figure 3.4, a pseudo-code is provided in Algorithm 2, its influence on the optimization result is shown in Section 3.2.2.

Rescue interceptor and additional line-search can be combined, as they are not redundant. This can be seen in the following way: The additional line-search implementation may reduce step-sizes, if integration failure is encountered, but will finally accept one of the proposed steps after a predefined number of iterations. If the ODE can yet not be integrated, the rescue interceptor will be activated in the next optimization step. It also occurs that the ODE can be integrated for a given parameter vector, but the numerical computation of its gradient is not possible. Also such a case will trigger the rescue functionality, despite an additional line-search being used.

Implications of the algorithmic adaptations on convergence results

It was pointed out previously that the implementation of the rescue interceptor behaves similar to a trust-region algorithm which only shrinks the trust-region radius in case of ODE integration failure. Assuming that the biological system is modeled appropriately, the global optimum should be located in a biologically plausible domain in parameter space. Consequently, ODE integration failure is not expected to occur sufficiently close to the global optimum, i.e., in some open neighborhood of it. Adhering to the assumption that the learning rate – and hence also the optimization step size – shrinks over the optimization process (Assumption A2), the optimizer will at some point reach this open neighborhood and will stay in it, as the step-size converges to

Algorithm 1 Rescue functionality for mini-batch optimization
Code parts specific to rescue functionality are shown in blue

Initialization:

Set initial parameter vector $\theta \leftarrow \theta_0$
Set and initialize optimization algorithm for parameter update
Set stopping criterion, e.g., maximum number of epochs N
Set variables to store necessary information about previous optimization step:
parameter vector θ^*, gradient estimate g^*, possibly optimization algorithm dependent quantities Q^*,
initialize with $(\theta^*, g^*, Q^*) \leftarrow (\text{NaN}, \text{NaN}, \text{NaN})$
Set reduction factor $\beta = 1$
Set reduction multiplier $r < 1$
Set increase multiplier $c > 1$

Procedure for optimization with rescue functionality:

while Stopping criterion not met **do**
  Get next mini-batch
  Compute gradient estimate: $g \leftarrow \text{gradientFunction}(\theta)$
  if ODE integration was successful **then**
    Update old quantities: $(\theta^*, g^*, Q^*) \leftarrow (\theta, g, Q)$
    Update reduction factor: $\beta \leftarrow \min(c \cdot \beta, 1)$
    Perform parameter update: $\theta \leftarrow \text{parameterUpdater}(\theta, g, Q, \beta)$
  else
    if θ^* is NaN **then**
      return ODE integration failure at initial point
    end if
    while Last call to gradientFunction failed **and** maximum number of rescue steps not reached **do**
      Undo last step: $(\theta, g, Q) \leftarrow (\theta^*, g^*, Q^*)$
      Update reduction factor $\beta \leftarrow r \cdot \beta$
      Perform parameter update: $\theta \leftarrow \text{parameterUpdater}(\theta, g, Q, \beta)$
      Compute gradient estimate: $g \leftarrow \text{gradientFunction}(\theta)$
    end while
    if Last call to gradientFunction failed **then**
      return ODE integration failure not recoverable
    end if
  end if
end while

Changes in parameterUpdater-*functions*:

Replace parameter update $(\theta \leftarrow \theta + \eta \cdot \delta)$ by $(\theta \leftarrow \theta + \eta \cdot \beta \cdot \delta)$

Algorithm 2 Line-search functionality for mini-batch optimization
Code parts specific to line-search functionality are shown in blue

Initialization:

Set initial parameter vector $\theta \leftarrow \theta_0$
Set and initialize optimization algorithm for parameter update
Set stopping criterion, e.g., maximum number of epochs N
Set variable to store information from previous optimization step: parameter vector θ^*
Set line-search reduction factor $\gamma = 1$

Procedure for optimization with line-search functionality:

while Stopping criterion not met **do**
  Get next mini-batch
  Compute objective and gradient estimate: $\left(j,g\right) \leftarrow$ gradientFunction(θ)
  Set line-search reduction factor $\gamma = 1$
  Perform parameter update: $\theta \leftarrow$ parameterUpdater(θ, g, Q, γ)
  Compute objective: $j^{(1)} \leftarrow$ objectiveFunction(θ)
  if Improvement: $j^{(1)} < j$ **then**
    Accept step: $\theta^* \leftarrow \theta$
  else
    Undo last step: $\theta \leftarrow \theta^*$
    Compute optimal line-search factor $\gamma \leftarrow$ based on quadratic interpolation for $g, j, j^{(1)}$
    Perform parameter update: $\theta \leftarrow$ parameterUpdater(θ, g, Q, γ)
    Compute objective: $j^{(2)} \leftarrow$ objectiveFunction(θ)
    if Improvement: $j^{(2)} < j$ **then**
      Accept step: $\theta^* \leftarrow \theta$
    else
      Undo last step: $\theta \leftarrow \theta^*$
      Compute optimal line-search factor $\gamma \leftarrow$ based on cubic interpolation for $g, j, j^{(1)}, j^{(2)}$
      Perform parameter update: $\theta \leftarrow$ parameterUpdater(θ, g, Q, γ)
      Compute objective: $j^{(3)} \leftarrow$ objectiveFunction(θ)
      Set $m = 3$
      while No improvement **and** maximum number of line-search steps not reached **do**
        Undo last step: $\theta \leftarrow \theta^*$
        Compute optimal line-search factor $\gamma \leftarrow$ based on cubic interpolation for $g, j, j^{(m-1)}, j^{(m)}$
        Perform parameter update: $\theta \leftarrow$ parameterUpdater(θ, g, Q, γ)
        Compute objective: $j^{(m+1)} \leftarrow$ objectiveFunction(θ)
        Increment $m \leftarrow m + 1$
      end while
      Accept step: $\theta^* \leftarrow \theta$
    end if
  end if
end while

Changes in parameterUpdater-*functions*:

Replace parameter update $(\theta \leftarrow \theta + \eta \cdot \delta)$ by $(\theta \leftarrow \theta + \eta \cdot \gamma \cdot \delta)$

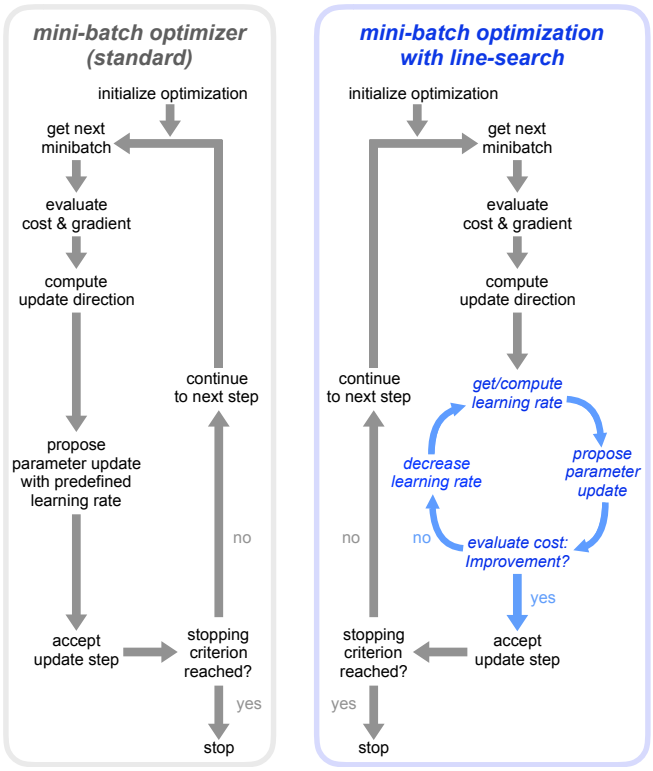

Figure 3.4: Comparison of standard mini-batch optimization and mini-batch optimization with line-search. Left panel: Standard mini-batch optimization often uses a prescheduled learning rate, which determines the step-size during optimization regardless of whether an optimization step leads to an improvement or not. Right panel: If line-search is enabled, the objective function is re-evaluated on the same mini-batch and checked for improvement. If no improvement is achieved, the learning rate is reduced until either an improvement is achieved or until the maximum number of line-search steps is reached. This figure is adapted from Figure 7 of the author's publication [Stapor et al., 2019].

0. Then, the reduction factor β reaches and remains at its initial value 1. Thus, for a given mini-batch optimization algorithm, the corresponding convergence proof remains unchanged, if the method is combined with the rescue interceptor. Hence, the rescue interceptor does not interfere with the convergence of a chosen mini-batch optimization algorithm and has no negative influence on its asymptotic properties.

In contrast to the rescue interceptor, the additional line-search functionality only has an effect on the current optimization step, not on the subsequent ones. However, the optimizer step-size may be affected also in proximity to the global optimum. Hence, in order to keep existing convergence results, the line-search functionality can be slightly adapted: If a lower bound $\gamma_{\min}$ on the reduction factor γ is enforced, Assumption A1, which is necessary in the convergence proofs of most of the common algorithms, will not be violated and hence, convergence is ensured. This lower bound could be realized by, e.g., setting $\gamma = \gamma_{\min}$ and stopping the iterative line-search algorithm, as soon as a reduction factor $\gamma < \gamma_{\min}$ is proposed.

Implementation in the toolbox parPE

We implemented the mentioned mini-batch algorithms together with the discussed adaptations in the computational toolbox parPE. parPE is a C++ library [Schmiester et al., 2019a], which provides the means for parallelized objective function evaluation and optimization of ODE models

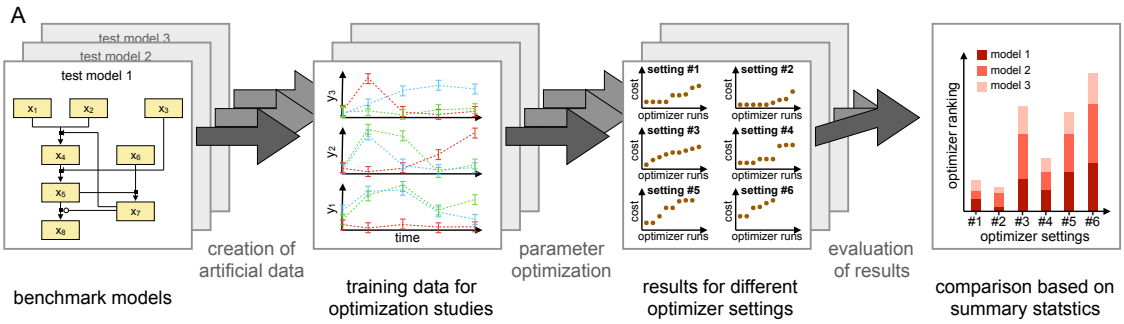

Figure 3.5: Schematic overview of bechmarking full-batch and mini-batch optimization methods on small- to medium-scale models. Benchmark models were chosen and observation functions adapted, noisy artificial data created based on simulations with the reported parameter vectors, 100 initial points for optimization were drawn randomly, multi-start local optimization was carried out, and multi-start results were compared against each other to compute summary statistics. This figure is adapted from Figure 2 of the author's publication [Stapor et al., 2019].

generated by the ODE solver toolbox AMICI [Fröhlich et al., 2019] using optimizers such as Ipopt [Wächter and Biegler, 2006].

parPE was originally developed as part of the publication [Schmiester et al., 2019a]. The implementation, adaptation and testing of mini-batch optimizers is a contribution of this thesis.

3.2.2 Benchmark study on test models: Identifying the most important hyperparameters

The implementation of mini-batch optimization in parPE allowed us to directly compare full-batch against mini-batch optimizers, to benchmark mini-batch algorithms against each other, and to assess the influence of mini-batch hyperparameters on the optimization of ODE models. Furthermore, this is – to the best of the author's knowledge – the first implementation of mini-batch optimization schemes tailored to ODE models, which employs state-of-the-art techniques, such as stiff ODE integrators and adjoint sensitivity analysis.

Application examples

To evaluate the adapted mini-batch optimization algorithms for ODE models, we considered three benchmark problems (Table 3.1). These small- to medium-scale examples were chosen based on a recently published collection of benchmark models [Hass et al., 2019]. To facilitate the analysis of the scaling behavior with respect to the number of experimental conditions, we decided to generate artificial (i.e., simulated) data (Figure 3.5). We hence picked models with different system sizes, which allowed such to generate large artificial datasets, which were sufficiently heterogeneous. This should ensure clear differences in objective function values and gradients, when different mini-batches were used. In order to allow the creation of heterogeneous datasets, the SBML files, input parameters, and observable functions were slightly altered. Additive Gaussian noise was added to the model simulations, using the noise levels which were reported in [Hass et al., 2019] for each observable. The precise model versions have been made freely available in the SBML/PEtab format at Zenodo, under https://doi.org/10.5281/zenodo.3556429

We used a mini-batch size of 30 experimental conditions and 50 epochs of training – corresponding to roughly 50 iterations of a classical full-batch optimizer – which are typical hyperpa-

	Fujita[1]	**Bachmann**[2]	**Lucarelli**[3]	**Fröhlich**[4,*]
State variables	9	25	33	1228
Parameters	19	40	72	4517
Time points	10	10	8	1
Conditions	600	1,200	1,500	13,000
Data points	6,000	12,000	60,000	13,000
Data type	artificial	artificial	artificial	measurements
Modeled system	EGF/AKT signaling	JAK/STAT signaling	TGF/Smad signaling	pan cancer signaling

Table 3.1: Overview of ODE models for benchmarking of mini-batch optimization.
* The model from Fröhlich et al. was not used in the benchmark study, but later as application example.

rameter choices in deep learning [Goodfellow et al., 2016]. We benchmarked the four implemented optimization algorithms: Vanilla stochastic gradient descent (SGD), stochastic gradient descent with momentum, RMSProp, and Adam. To assess the impact of the learning rate, we considered three schedules:

- Medium learning rate, logarithmically decreasing from 10^{-1} to 10^{-4}.
- Low learning rate, logarithmically decreasing from 10^{-2} to 10^{-5}.
- Constant learning rate, fixed to the value 10^{-3}.

To ensure robustness of the mini-batch optimizers against failure in ODE integration, we activated the rescue interceptor in all local optimizations, but so far did not use the additional line-search implementation. The well-established full-batch optimizer Ipopt [Wächter and Biegler, 2006] was used as benchmark and was granted 50 iterations, so all tested methods had a similar computational budget. For each model, 100 randomly chosen initial parameter vectors were created, from which all optimizers were started. To assess the overall performance of each optimizer setting, we sorted the starts by their final objective function value and each of the 100 starts was ranked across the optimizer settings. Computing the mean of the 100 rankings for each setting led to an averaged rank per model, which we used as a proxy for overall optimization quality (Figure 3.5).

Benchmarking algorithms and hyperparameters

We found across all algorithms that the medium, but decreasing learning rate was preferred, the low but decreasing learning rate was second and the constant learning rate resulted in the worst performance (Figure 3.6 A). A higher learning rate in the beginning of the optimization process seemed to be crucial for the mini-batch optimizers to progress quickly towards favorable regions of the parameter space. Given the medium learning rate, different algorithms were able to compete with or even outperform the full-batch optimizer Ipopt, but the adaptive algorithm RMSProp performed particularly well. In most cases, the preferred learning rates led to step-sizes during optimization which were comparable or slightly lower than those which were chosen by classical (full-batch) optimization methods.

[1] Model adapted from Fujita et al. [2010]
[2] Model adapted from Bachmann et al. [2011]
[3] Model adapted from Lucarelli et al. [2018]
[4] Model adapted from Fröhlich et al. [2018a]

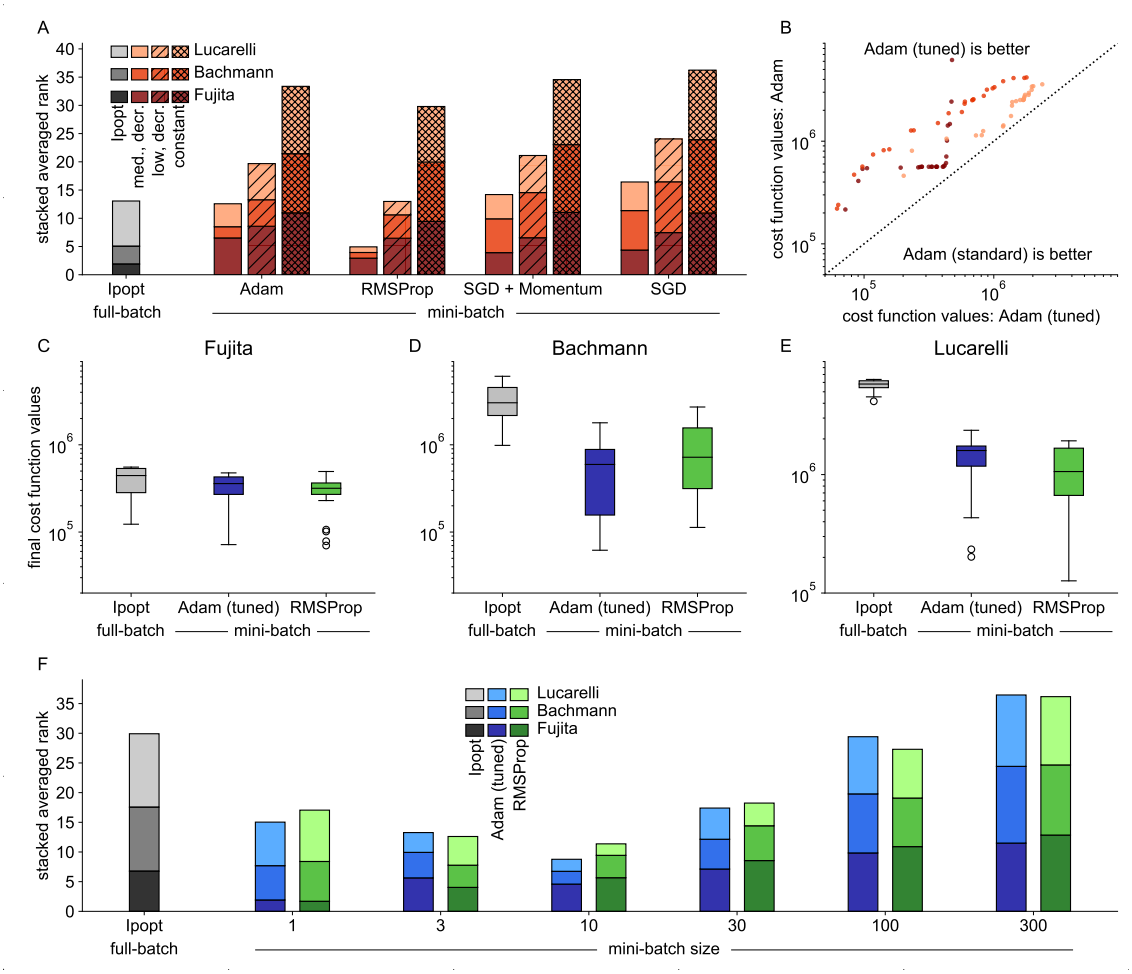

Figure 3.6: Bechmarking full-batch against mini-batch optimization methods on small- to medium-scale models. **A** Comparison of performance for different local optimizers with three different learning rate schedules (lower rank implies better performance, ranks of models are stacked). **B** Top 25 starts of the local optimizer Adam with tuning parameters taken from the literature (standard) vs. a simplified version (tuned). **C-E** Boxplots of final cost function values for the best 25 starts of the two best mini-batch optimizers, compared against the Ipopt (full-batch optimizer), for each model. **F** Comparison of performance for all starts of the best two mini-batch optimizers given the best learning rate, for different mini-batch sizes, compared against Ipopt (ranks of models are stacked). This figure is adapted from Figure 2 of the author's publication [Stapor et al., 2019].

Given these findings, we compared the optimization algorithm Adam – which is maybe the most popular algorithm for training deep neural nets – with two different tuning variants: the tuning proposed in the original publication (called standard, see [Kingma and Ba, 2015]) and a simplified scheme (called tuned), which employs the same rate for both internally used decaying averages. The analysis of the best 25 starts for all models with medium, decreasing learning rate showed that the tuned version outperformed the original one for all cases on our benchmark examples (Figure 3.6 B). When comparing the performance of the tuned version of Adam and RMSProp with medium learning rate, we see that they show a very similar performance for the best 25 starts for all three tested models and perform as good as Ipopt or even better (Figure 3.6 C-E).

We then assessed the impact of the mini-batch size on the optimization result. Again, we used an average ranking, 100 starts, and investigated 6 mini-batch sizes for each model. We restricted our analysis to the two previously best performing optimization algorithms, tuned Adam and RMSProp, with the medium, decreasing learning rate. We found that in general, small mini-batch sizes were preferred, but the optimal size seemed to be model dependent (Figure 3.6 F). While a mini-batch size of only one experimental condition worked best for the smallest example (Fujita), a mini-batch size of 10 experimental conditions performed best for the other two examples, yielding about 0.1% to 1% of the whole dataset as mini-batch. Interestingly, the mini-batch size seemed to impact both optimization algorithms to the same degree.

Impact of algorithmic adaptations

The first observation of the previous study was that higher learning rates improved the optimization process. Hence, we tested the effect of further increasing it and introduced an additional learning rate:

- High learning rate, logarithmically decreasing from 10^{-0} to 10^{-3}.

Despite the previous results, the high learning rate seemed to obstruct the optimization process (Figure 3.7 A). This finding motivated the implementation of the additional backtracking line-search: As it is a priori not clear what a good choice for the learning rate would be for a given model, we concluded that using rather high learning rates in combination with an additional line-search might resolve the problem of finding an adequate learning rate for each model.

In a next step, we wanted to asses the effect of our algorithmic adaptations. We found that ODE integration failure was present in our benchmark models, although it was less critical than expected: Only the optimization of the smallest model (Fujita) suffered substantially from these failures. However, by using the rescue interceptor, all local optimization runs could be recovered, as long as the ODE integration failure did not happen at the initial point of optimization (Figure 3.7 B).

We then evaluated both algorithmic improvements together, for Adam and the medium and high learning rate on the three benchmark models (Figure 3.7 C). Interestingly, we found the strongest improvement for the largest model, although it suffered only little from integration failure. The line-search improved the optimization process at high learning rates, which can be seen in a direct comparison (Figure 3.7 D) and in the waterfall plot (Figure 3.7 E). Considering all three models, we saw that the rescue interceptor was always helpful, whereas the line-search could also reduce the computational efficiency in case a good learning rate was chosen (Figure 3.7 C). This is not surprising, as the line-search needs additional computation time and some optimization runs were stopped prematurely due to imposed wall-time limits. However, these adverse effects at lower learning rates were mild when compared against the positive effects at

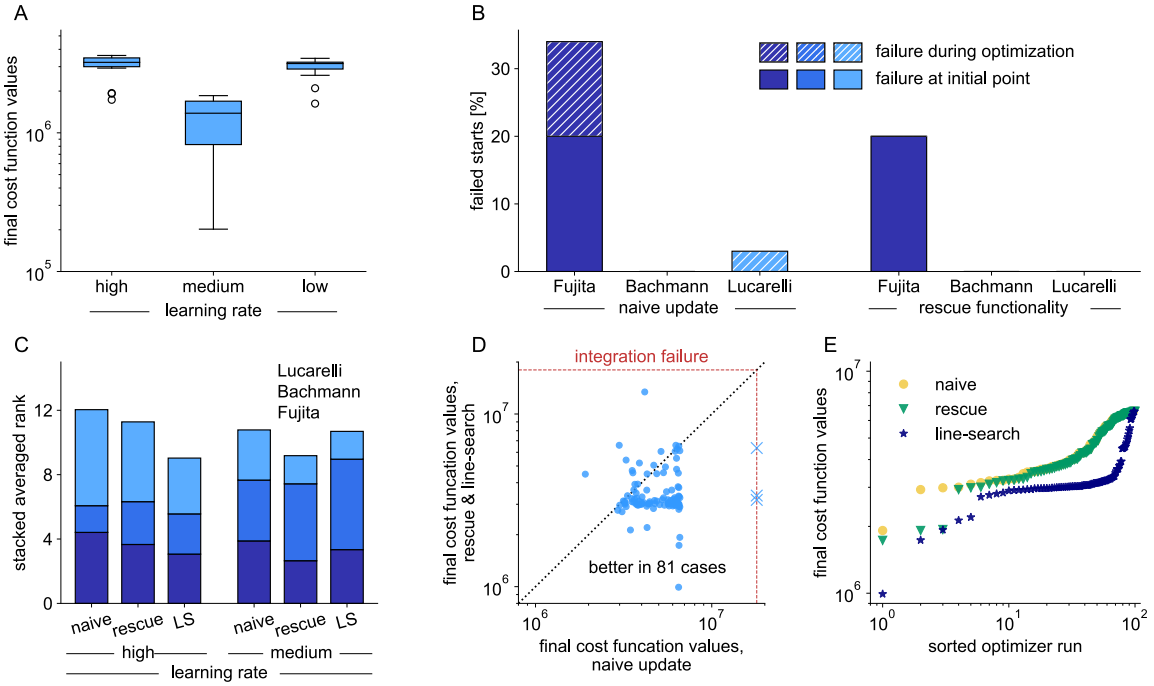

Figure 3.7: Influence of line-search methods on optimizers performance and reliability. **A** Boxplots for the best 25 starts of mini-batch optimizers Adam for three different learning rates for the largest example (Lucarelli), showing that too high learning rates obstruct the optimization process. **B** Percentage of failed local optimizations per model (with optimizer Adam) due to non-integrability of the underlying ODE. Failure at the initial point of optimization can not be recovered, but failure during the optimization process is prevented when applying the rescue functionality. **C** Comparison of performance for optimizer Adam, given different learning rates, for naive implementation, with rescue functionality, and rescue functionality and line-search (ranks of models are stacked). **D** All starts of the local optimizer Adam for the largest of the three examples (Lucarelli), naive implementation compared against rescue functionality and line-search, employing a high learning rate. **E** Waterfall plot for the largest of the three examples (Lucarelli), for naive implementation of Adam, with rescue functionality and with rescue functionality and line-search. This figure is adapted from Figure 3 of the author's publication [Stapor et al., 2019].

high learning rates. As the selection of a good learning rate is currently a trial-and-error process, the adaptation is overall beneficial. However, if a decent learning rate has been found for a specific model, the additional line-search does not improve optimization any further.

3.2.3 Application 1: Large-scale ODE model of cancer signaling

After successfully testing mini-batch optimization on the small- to medium-scale models, we evaluated the method on a large-scale model with real measurement data, for which we expected the method to have the highest benefit. Therefore, we considered the largest publicly available ODE model of cancer signaling [Fröhlich et al., 2018a] (see also Table 3.1). It comprises various pathways and their cross-talk and captures 1,228 biochemical species and 2,686 reactions. The generic chemical reaction network can be adapted to cancer cell-lines and treatment conditions using input parameter vectors, which describe the mutation and expression status of different genes, certain initial conditions, growth medium components, and drug binding affinities. We could extract and match overall 16,308 data points of viability readouts for different cell-lines under various (single drug) treatment conditions from the Cancer Cell Line Encyclopedia [Barretina et al., 2012] to the model. We split these data into a training set of 13,000 data points

and an independent test set of 3,308 data points. To the best of the author's knowledge, this was the first time that an ODE model has been trained on a dataset with so many experimental conditions.

Impact of learning rate and backtracking line-search on large-scale application example

To confirm our findings from the smaller models, we revisited them on the large-scale application example. We first restricted the parameter estimation for this model to 20 multi-starts with different optimizers. Compared to the common practice for smaller models [Raue et al., 2013b, Villaverde et al., 2018], 20 local optimization runs are unlikely to be sufficient to safely infer a global optimum, but this was the highest number of runs which has so far been carried out in previous studies for a model of this size, due to the high computation times of more than 60,000 hours [Schmiester et al., 2019a]. We hence used the optimization with the local full-batch optimizer Ipopt as benchmark, and granted it 150 iterations, applying an L-BFGS scheme as Hessian approximation, as done in [Schmiester et al., 2019a].

As mini-batch optimizer, we used Adam, running for 20 epochs with two different learning rates: We had previously seen that the medium learning rate performed best. However, for Adam, the Euclidean norm of the initial optimization step $\Delta\theta$ scales roughly with the square root of the parameter dimension [Kingma and Ba, 2015], which can be seen in the following way: An optimization step of Adam is computed component-wise for each entry of the parameter vector θ_r as a ratio of two decaying averages: The numerator is a decaying average over the past r-th entries of the gradient, controlled by the tuning parameter ρ_1, with $0 < \rho_1 < 1$. The denominator is the square root of a decaying average over the past squared r-th entries of the gradient (root-mean-square), controlled by the tuning parameter ρ_2, with $0 < \rho_2 < 1$, stabilized by a small constant ε. Denoting the r-th entry of the objective function gradient with g_r, the learning rate with η, and the update direction with d, this yields for the first optimization step:

$$\|\Delta\theta\|_2 = \|\eta d\|_2 = \sqrt{\sum_{r=1}^{n_\theta} \frac{\left(\frac{(1-\rho_1)g_r}{1-\rho_1}\right)^2}{\left(\sqrt{\frac{(1-\rho_2)g_r^2}{1-\rho_2}} + \varepsilon\right)^2}} = \sqrt{\sum_{r=1}^{n_\theta} \frac{g_r^2}{(|g_r| + \varepsilon)^2}} = \sqrt{\sum_{r=1}^{n_\theta} \frac{1}{1 + \frac{2\varepsilon|g_r| + \varepsilon^2}{g_r^2}}} \approx \sqrt{n_\theta} \tag{3.10}$$

This was the reason why we tested the two following learning rate schemes:

- Medium learning rate, logarithmically decreasing from 10^{-1} to 10^{-4}.
- Low learning rate (only reduced initial learning rate), logarithmically decreasing from 10^{-2} to 10^{-4}.

We furthermore tested both learning rates with and without additional line-search, as we did not have any prior information about a good learning rate beyond the aforementioned heuristics. Moreover, we also enabled the rescue interceptor. Mini-batch and full-batch optimizers were initialized at the same parameter vectors to ensure comparability of the optimization results. We additionally took snapshots of the optimization process with Ipopt at computation times that were as close as possible to those used by the mini-batch optimizations, to have a direct comparison.

We found that in terms of final objective function values and correlation with the training data, mini-batch optimization at lower learning rates yielded slightly better results than the

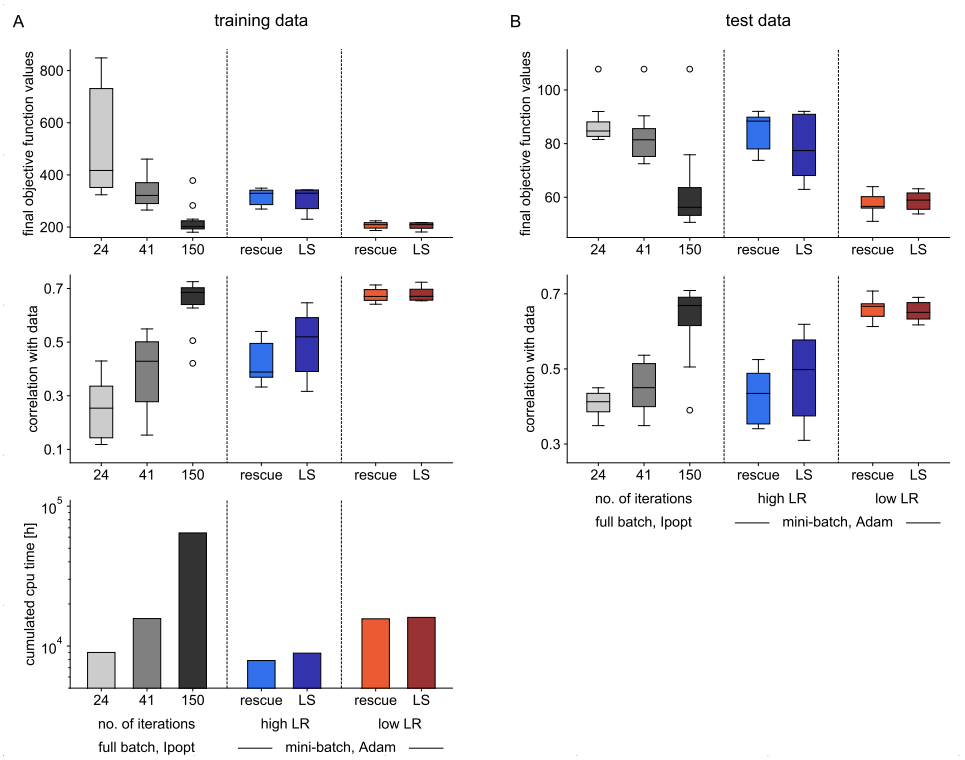

Figure 3.8: Optimization results for the large-scale cancer model, for different learning rates (LR), with rescue functionality only (rescue) and with additional line-search (LS). **A** Boxplots of the 10 best runs out of 20, for Ipopt (at three stages of optimization) and for Adam: final objective functions values (upper panel), correlation of model simulation with measurements (middle panel), and total CPU time for 20 runs (lower panel). **B** Boxplots of the 10 best optimization runs, for the same settings as in A. This figure is taken from the author's publication, Figure 5 [Stapor et al., 2019].

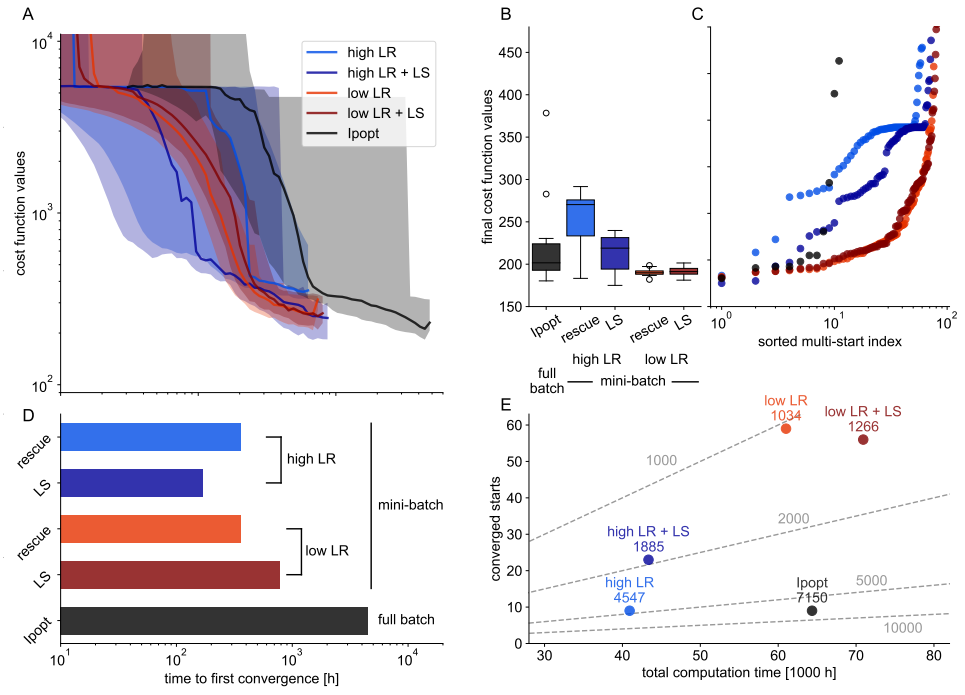

Figure 3.9: In-depth analysis for Figure 3.8, with 100 multi-starts for mini-batch optimization. **A** Traces of objective function value during optimization (extrapolated to full dataset, smoothened) for the ten best starts against computation time. Shadings depict the envelopes of the ten traces, solid lines depict the medians. **B**, **C** Box plots and waterfall plots of all optimization runs. **D** Bar plots of with computation time, until first start converged. **E** Visualization of converged starts against total CPU time, yielding CPU time per converged start. This figure is taken from the author's publication, Supplementary Figure 10 [Stapor et al., 2019].

Ipopt runs with 150 iterations (Figure 3.8 A). In terms of total computation time, the mini-batch approach with low learning rates was faster by a factor of 4.1. Optimization with medium learning rates yielded inferior results in terms of objective function values and correlation with the data, but further reduced the overall computation time. Additional line-search markedly improved the optimization quality for this model at the medium learning rate, while increasing the computation time by less than 13%. For the runs with lower learning rate, the line-search had almost no effect on computation time and final objective function values. When assessing the fit to independent test data, we found again that mini-batch optimization with the lower learning rates showed similar or better results than Ipopt (Figure 3.8 B). As before for high learning rates, the optimization at medium learning rates was improved by line-search, but turned out to be inferior overall when compared to optimization at lower learning rates.

We investigated two further threshold-dependent characteristics to assess optimization performance: the computation time until convergence was reached for the first time and the number of converged starts per computation time. Both are common metrics for optimization performance [Villaverde et al., 2018]. As threshold for convergence, we defined a value-to-reach based on the ten best optimization results from Ipopt (the benchmark) and fixed it to the mean plus one standard deviation over final objective function values. We now granted 100 starts to the mini-batch optimizers, to allow them a similar budget of total computation time as for Ipopt.

Considering the computation time until the first start converged, mini-batch optimization at medium learning rate with line-search was fastest, outperforming Ipopt by a factor of up to 27 (Figure 3.9 A and D).

When comparing the best ten starts and the waterfall plot (Figure 3.9 B and C), we see that mini-batch optimization at low learning rates clearly outperformed Ipopt. Mini-batch optimization at medium learning rates showed a result that was comparable to Ipopt, but only if the additional line-search was enabled. This highlights the fact that even simple schemes for learning rate adaptation can be beneficial. When comparing the number of converged starts per computation time, mini-batch optimization was up to a 6.9-fold faster than full-batch optimization (Figure 3.9 E). This time, optimization with lower learning rates performed better. Hence, lower learning rates yield more reasonable step-sizes for large-scale models. Importantly, in the metric of converged starts per computation time, all of the four approaches clearly outperformed Ipopt.

Overall, these observations confirm the finding that the choice of the learning rate is a crucial hyperparameter when working with mini-batch optimizers for ODE models and that, if too high learning rates are chosen, line-search can substantially improve the optimization result (Figure 3.9).

Impact of mini-batch size on large-scale application example

To evaluate the robustness of mini-batch optimization with respect to mini-batch size, we ran optimizations with mini-batch sizes 10, 100, 1000, and 13000 (full-batch), granting 10, 20, 50, and 150 epochs of optimization time and 100, 100, 50, and 25 local optimizations, respectively. This time, we used Adam at low learning rate without the line-search feature, as the previous study indicated it to have little to no impact for the chosen learning rate. Moreover, especially for optimization with smaller batch sizes, lower learning rates implicitly make sense, as more optimization steps can be performed due to the lower mini-batch size.

For the large-scale model of cancer signaling, we found a clear benefit of smaller mini-batch sizes. Objective function and correlation values were substantially improved (Figure 3.10 A). As the total computation time was reduced by smaller mini-batch sizes due to the lower number of

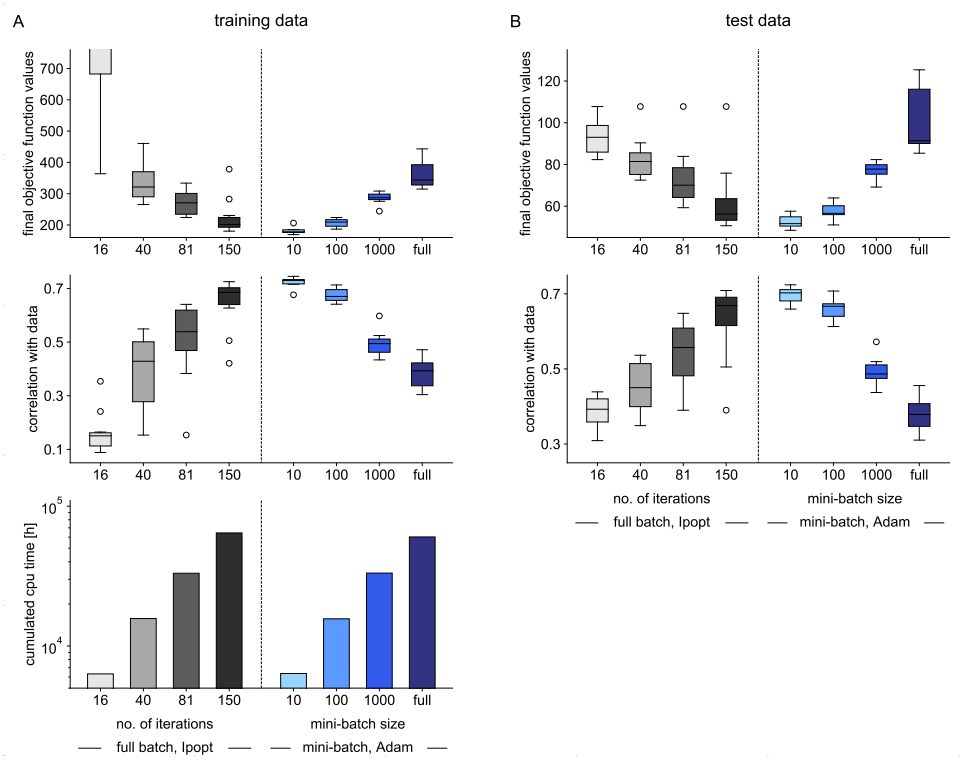

Figure 3.10: Optimization results for the large-scale cancer model, for different mini-batch sizes at low learning rate. **A** Boxplots of the 10 best runs out of 20, for lpopt (at four different stages of the optimization process) and for Adam: final objective functions values (upper panel), correlation of model simulation with measurements (middle panel), and total CPU time for all 20 runs (lower panel). **B** Boxplots of the 10 best optimization runs, for the same settings as in A. This figure is taken from the author's publication, Figure 6 [Stapor et al., 2019].

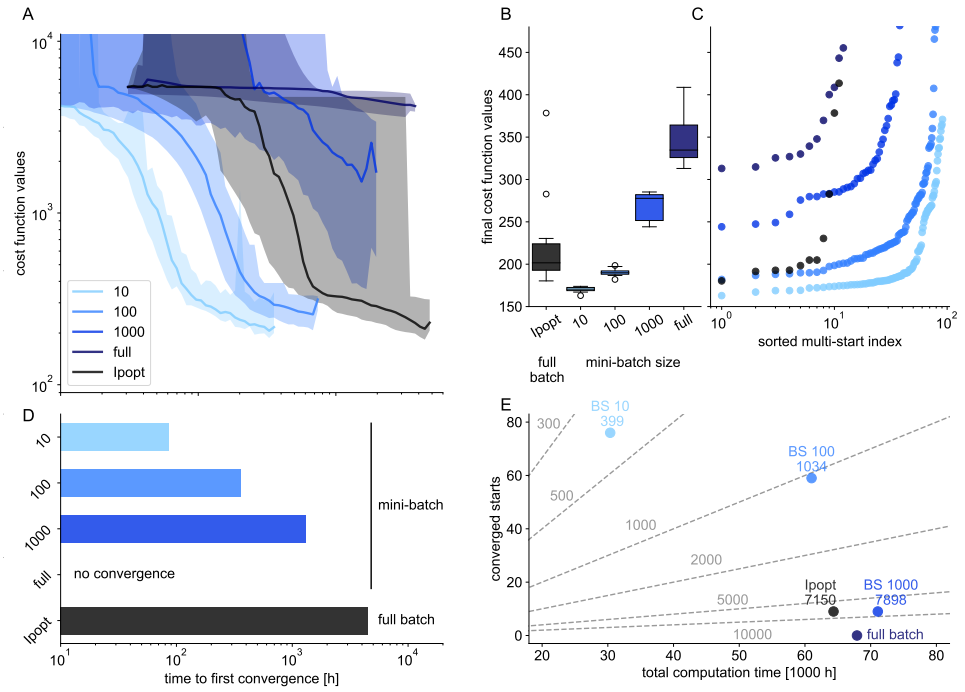

Figure 3.11: In-depth analysis for Figure 3.10, with up to 100 multi-starts for mini-batch optimization. **A** Traces of objective function value during optimization (extrapolated to full dataset, smoothened) for the ten best starts against computation time. Shadings depict the envelopes of the ten traces, solid lines depict the medians. **B**, **C** Box plots and waterfall plots of all optimization runs. **D** Bar plots of with computation time, until first start converged. **E** Visualization of converged starts against total CPU time, yielding CPU time per converged start. This figure is taken from the author's publication, Supplementary Figure 11 [Stapor et al., 2019].

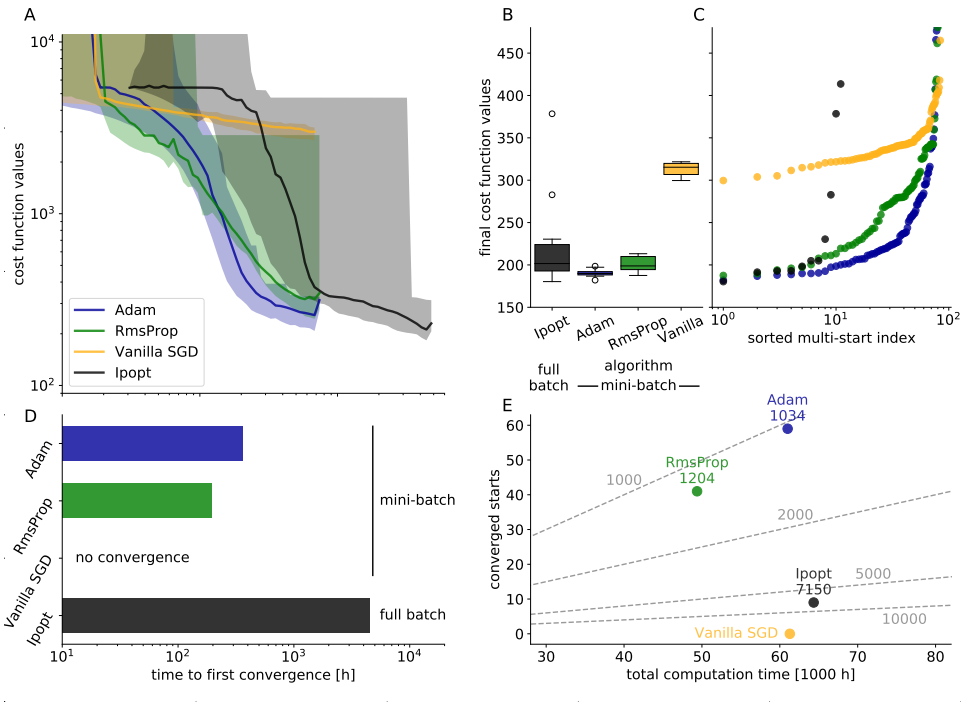

Figure 3.12: In-depth analysis for the optimizer comparison of different mini-batch algorithms, with 100 multi-starts for mini-batch optimization. **A** Traces of objective function value during optimization (extrapolated to full dataset, smoothened) for the ten best starts against computation time. Shadings depict the envelopes of the ten traces, solid lines depict the medians. **B**, **C** Box plots and waterfall plots of all optimization runs. **D** Bar plots of with computation time, until first start converged. **E** Visualization of converged starts against total CPU time, yielding CPU time per converged start. This figure is taken from the author's publication, Supplementary Figure 12 [Stapor et al., 2019].

epochs, we faced the counter-intuitive effect of a seeming anti-correlation of optimization performance and total computation time. Optimization with the smallest mini-batch size outperformed Ipopt in terms of final objective function values while reducing the total computation time by more than a factor of 10. The findings on the optimization quality persisted when looking at the performance on the set of independent test data (Figure 3.10 B). Again, the smallest mini-batch size yielded the best results, showing even a better generalization to independent test data than all previously tested approaches. A possible explanation of this effect might be a regularizing effect of small batch-sizes, which may lead to less overfitting of the training data.

Overall, the two smallest mini-batch sizes achieved better optimization results than Ipopt. Especially the waterfall plot for the smallest mini-batch size showed that not only computation time was reduced, but also the optimization quality was markedly improved (Figure 3.11 B and C). When comparing computation time to first convergence, mini-batch optimization was up to 52 times faster than Ipopt (Figure 3.11 A and D). In terms of converged starts per computation time, we found an 18-fold improvement when using mini-batch optimization (Figure 3.11 E).

As additional test, we assessed the influence of the optimization algorithm on the optimization result (Figure 3.12). This indicated again that the chosen algorithm was less important than the choice of the learning rate or the mini-batch size: Only the most simple algorithm, Vanilla SGD, showed a substantially worse performance for the application example, but the two more complex algorithms, RMSProp and Adam, were comparable in all of the investigated metrics.

3.2.4 Application 2: Comparing ensemble methods with point estimates for large-scale ODE models

The fact that mini-batch optimization substantially reduced the computation time per local optimization run allowed to increase the number of optimization runs during parameter estimation. This is important, as firstly, more runs enable a better exploration of the parameter space. Secondly, having more optimization results also allows to build richer ensembles from the multi-start result.

For small ODE models, using ensembles of multi-start results is uncommon, as mathematically more sound methods are available to assess the reliability of model predictions, such as profile likelihood analysis or MCMC sampling. However, for the considered large-scale application example, such methods would need millions of hours of computation time and weeks to months of wall time, which puts them out of reach. Ensembles can hence help to improve the performance of the trained model and similar approaches have been tested in the context of logic modeling [Costello et al., 2014] or when facing an automated model selection for a high number of models [Henriques et al., 2017]. Recently, ensembles have also been proposed for uncertainty quantification of ODE models [Villaverde et al., 2019].

To assess the influence of ensembles on model simulations and predictions, we relied on the mini-batch optimization results from 100 multi-starts, batch size of 100, and low learning rate with additional line-search. We considered this a realistic setting, which we achieved for our application example without an extensive ex-post tuning of hyperparameters and which could have been obtained for other application examples as well.

We used three metrics as comparison for how well the model simulation describes the training data to which it was fitted. Firstly, we considered the Pearson correlation between measurement data and model simulation (Figure 3.13 A). Then, we considered a cell-line under a specific treatment conditions in the dataset to be responsive, if the viability of the corresponding cell-line was reduced by more than a factor of two. This allowed us to compute a receiver-operator-characteristic (ROC), based on the model simulations. We hence used the area under the ROC (AUROC) as second measure (Figure 3.13 B). Based on the ROC, a classification threshold was determined by finding the tangential point of the ROC with a straight of slope 1 [Hastie et al., 2005]. Using this threshold, we classified model simulations as (non-)responsive for certain cell-lines and treatments. By this classification, we could compute an overall classification accuracy for model simulations, which we used as third quality measure for the model output.

We compared the model simulations for each of the best 50 optimization results alone (point estimates) against ensembles, which were built from the best parameter vectors, up to a specific multistart index, yielding 50 ensembles. Our analysis showed that model simulations based on the ensembles consistently outperformed the those from single point estimate in all three metrics (Figure 3.13 C). Point estimates always yielded best results for the best optimization run. For ensembles, the performance increased for all metrics when more optimization results were included and declined mildly when too many weak results were added. Hence, for each metric, there existed an optimal ensemble size, which ranged from 12 (for the correlation) to 38 (for the AUROC). These findings persisted, when we investigated the situation on independent test data, for which we used the classification thresholds, which had been inferred from the training data (Figure 3.13 D). Although the optimal ensemble size differed from those on the training data, ensembles outperformed point estimates for all metrics, and the performance of the ensemble increased with ensemble size, before gradually decreasing again.

These results suggest that the training data can be used to infer a reasonable ensemble size, based on one or more different metrics. It seems to be reasonable to generate large ensembles,

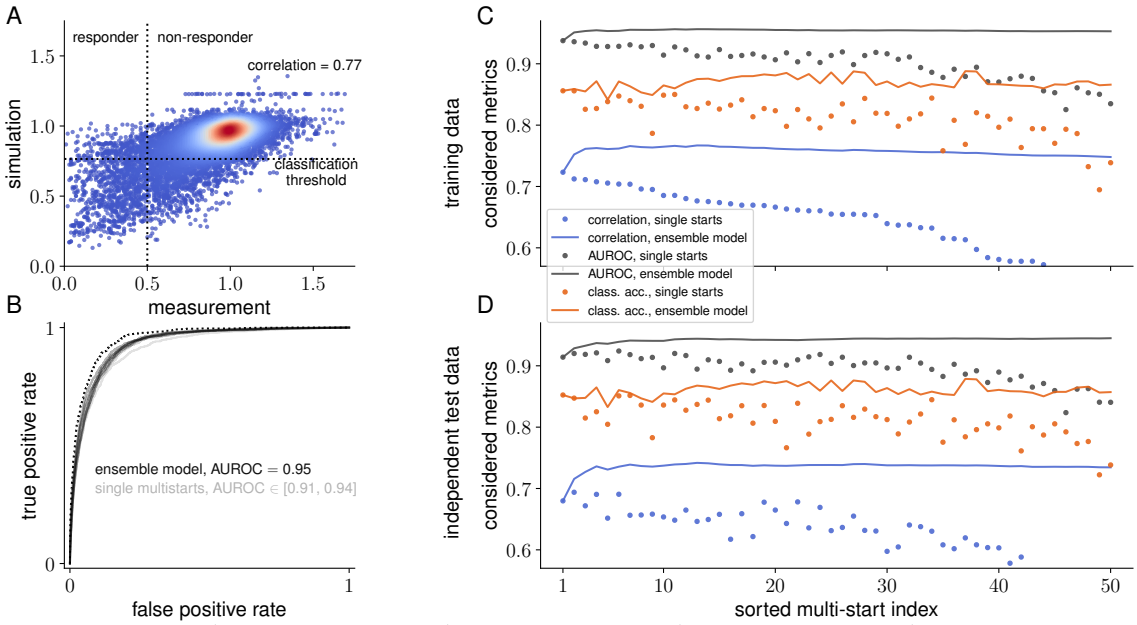

Figure 3.13: Comparison of model simulation quality in three different metrics. **A** Correlation of model simulation with measured data for the training set. Cell-lines with treatments, which reduced viability by more than 50%, were labeled as responsive and a threshold for classifying a model simulation as (non-)responsive was inferred. **B** Receiver-operator-characteristics (ROCs) for the ten best optimization results (grey, solid lines) and for an ensemble model (black, dotted line) of the 12 best starts. **C** Assessing the quality of model simulations on the training set via: correlation (blue), area under ROC (AUROC, grey), and classification accuracy (orange), which was computed for each of the 50 best starts (dots) and for ensembles, created from the best optimization results up to a certain start (solid lines). **D** Assessing the quality of the model predictions on the test set, for the same metrics and simulation settings as in C. Classification accuracy was computed for thresholds inferred from the training set. Subfigure A and B of this figure are adapted from Figure 4 of the author's publication [Stapor et al., 2019].

as larger collections might better represent the posterior distribution of the model parameters. However, including too many suboptimal parameter vectors in the ensemble reduces its overall performance.

3.3 Discussion

3.3.1 Summary and conclusion

In this chapter, we introduced mini-batch optimization for ODE models, presented algorithmic adaptations to make this method better suitable for the considered problem class, and implemented these algorithms into an open source framework. Then, we identified the most important hyperparameters of the new method in a benchmark study of small- to medium-size models with artificial data and transferred the knowledge to a large-scale application example model, which we trained on a large-scale dataset of publicly available measurement data. On this application example, the proposed method was able to outperform established approaches for parameter estimation by more than an order of magnitude in terms of computation time, while providing better optimization results. Finally, we showed that the results from mini-batch optimization can be combined with ensemble methods when generating model simulations and predictions and that these ensemble methods clearly outperformed the more commonly used point estimates for

the investigated application example.

We identified the choices of learning rate and mini-batch size to be the most influential hyperparameters for parameter optimization and made the three following observations: Firstly, learning rates for mini-batch optimization which yield step-sizes slightly smaller than those used by established optimization techniques are a good choice. This might indicate that good step-sizes for mini-batch optimization can be identified with similar means as this is done in full-batch optimization. Secondly, surprisingly small mini-batch sizes were preferred in all of our application examples. Hence, it makes sense for a modeler to test batch sizes with few experimental conditions first. Thirdly, the choice of the optimization algorithms seems to be less important, as at least the two adaptive algorithms Adam and RMSProp performed equally well on all examples.

We have seen that mini-batch optimization can substantially speed-up parameter estimation for ODE models, if large datasets are used, which addresses Challenge (i) from the Introduction. However, this means that applications of mini-batch optimization will mostly be restricted to models, which need such datasets to constrain model parameters, as optimization techniques for smaller-scale models are already efficient. Typically, large datasets will be mostly employed for large-scale ODE models.

Moreover, as mini-batch optimization allows running more local optimizations than classical optimization approaches, due to the reduced computation time, it is naturally suited for a combination with ensemble methods. We could show that ensemble methods yield better model simulations and predictions than point estimates, at least for our large-scale application example, therefore addressing Challenge (ii). We hypothesize that this might be due to the high number of non-identifiable directions in parameter space for large-scale models [Fröhlich et al., 2018a, Kapfer et al., 2019]. Poorly identifiable parameters may (but don't have to) lead to misguided model simulations, which may be a reason for the observed lower performance of model simulations based on point estimates. In the case of low parameter identifiability, model simulations based on a consensus of parameter vectors might increase the reliability model simulations, as it was recently pointed out in [Villaverde et al., 2019].

3.3.2 Open problems in mini-batch optimization

Although the presented implementation already outperformed established standards, it is far from being mature. Many problems persist when applying mini-batch techniques to ODE models and the work in this thesis should only be considered as a proof of applicability and first step towards a new method, which opens up further directions for parameter estimation of ODE models.

Adaptivity of hyperparameters

The learning rate and the mini-batch size are crucial for a successful optimization, but so far, no clear rules beyond heuristics for their tuning exist. We hence think that learning rates and mini-batch sizes would be promising candidates for self-tuning schemes.

Concerning the learning rate or step-size, many options exist, which could be adapted from full-batch optimization strategies, where self tuning schemes are commonly applied to fix the optimizer step-size [Boyd and Vandenberghe, 2004, Byrd et al., 1988, Nocedal and Wright, 2006]. Combining those with mini-batch optimization might lead to substantial improvements. We also proposed and tested a corresponding line-search implementation for mini-batching, which may serve as a starting point.

For the mini-batch size, auto-tuning may be less straight forward, but also here, first approaches exist, which are based on assessing the variance of the objective function gradient

across a chosen mini-batch and possibly enlarging the mini-batch size [Lei and Jordan, 2019]. For specific algorithms, also first theoretical results have been proposed [Gazagnadou et al., 2019]. Probably any scheme for automatically choosing a good batch size would be highly beneficial, as we have seen that the mini-batch size is likely to be the most important hyperparameter for an efficient optimization process.

Finding adequate stopping criteria

A further open problem is finding adequate stopping criteria for mini-batch optimization. The number of epochs over which to optimize has direct influence on the learning rate and the optimization success. Thus, although not being considered explicitly as hyperparameter in our study, it also has to be fixed prior to optimization. Ideally, a self tuning approach could help to eliminate this hyperparameter. A promising approach might be combining the presented algorithms with early-stopping methods, which assess the variance of the gradient-estimate over the mini-batch [Mahsereci et al., 2017].

3.3.3 Outlook and next steps

Hierarchical mini-batch optimization

Another algorithmic improvement – more specific to ODE models – would be combining mini-batch optimization with hierarchical optimization for observation-specific parameters, such as scaling factors or parameters for measurement noise [Loos et al., 2018a, Schmiester et al., 2019a]. This approach has allowed substantial improvements in parameter optimization for ODE models and it is to be expected that also mini-batch optimization would benefit from it. A combination of the two methods would make it necessary to draw the mini-batches more carefully, such that the dependencies between the observation specific parameters of the model are respected. This should in principle be possible, but may require differently sized mini-batches, which depend on the structure of the training data.

Improving convergence by variance reduction or second order techniques

A complementary approach would be to implement optimization algorithms with so called variance reduction techniques [Lei and Jordan, 2019]. These approaches either store and reuse computed gradient estimates from past mini-batches or compute occasionally a gradient estimate based on a substantially larger mini-batch, in order to reduce the randomness in the gradient estimate per mini-batch. Some of these methods enjoy good theoretical convergence properties but are demanding in terms of memory consumption, which might make them prohibitive for applications in deep learning, but possibly still well suited for training of ODE models. Especially the ideas behind SAG [Schmidt et al., 2017] and SAGA [Defazio et al., 2014] allow a combination with sophisticated stopping criteria or self-tuning of the optimizer step-size [Gazagnadou et al., 2019], which makes them attractive for further development in the field of ODE models.

Another approach would be optimization techniques based on (approximate) second-order derivatives, such as proposed in recent publications [Bottou et al., 2017, Chung et al., 2017a, Wang et al., 2017]. So far, stochastic Newton or quasi-Newton methods have been less used in the context of mini-batch optimization [Bottou et al., 2017]. One reason may be that the matrix inversion in Newton-type methods can cause instabilities: Exact Newton-type methods have been shown to work less reliably when combined with mini-batching and may even fail to converge to the optimal solution [Chung et al., 2017a]. However, in quasi-Newton methods,

this problem can be circumvented by updating the approximation of the inverse of the Hessian directly [Chung et al., 2017a]. The particular advantage of mini-batch second-order optimization methods in the context of ODE models would be that techniques for step-size control could be applied which are very similar to those in full-batch optimization, such that these methods would substantially simplify the choice of a good learning rate. Moreover, second-order techniques can also be combined with variance reduction techniques (as, e.g., done in [Wang et al., 2017]), which makes them even more appealing.

Mini-batch optimization and uncertainty analysis

The last and probably most challenging problem is about uncertainty analysis for parameters and predictions of large-scale models [Kapfer et al., 2019]. In principle, the most insightful methods are profile likelihoods and MCMC sampling, which are widely applied for smaller models. For large-scale models however, either completely different approaches or at least adaptations have to be found, due to the high computational effort which is related to these methods. A possible approach could be ensembles, which we discussed in this chapter in the context of model simulations, but which have recently also been proposed to assess uncertainties of parameters and predictions [Villaverde et al., 2019]. Going beyond optimization results, ensembles could also be created from the optimization history, as done in [Villaverde et al., 2019]. Again, mini-batch optimization would be best suited for this, as it creates a richer optimization history (due to the higher number of optimization steps, which can be performed).

It has also recently been pointed out that mini-batch optimization with Vanilla SGD, fixed step-size, and batch size 1 can be interpreted as a special case of the Hamiltonian Monte Carlo method [Mandt et al., 2018], a particularly popular family of MCMC algorithms [Hoffman and Gelman, 2014]. From the point of view of ensemble methods, it makes heuristically sense: Initializing a mini-batch optimization for one or few epochs at the global optimum with fixed learning rate and small step-size easily creates a rich ensemble for uncertainty quantification. Nevertheless, it is important to note that this heuristic can be made precise, as done in [Mandt et al., 2018]. However, an application of this approach still has to be evaluated for ODE models.

Finally, mini-batching is not restricted to optimization: An application of this idea to the Metropolis-Hastings (MH) algorithm, i.e., MCMC sampling, has been proposed recently [Seita et al., 2018]. In the past decades, many adaptations to the MH algorithm have been presented, which increase its efficiency (see, e.g., [Ballnus et al., 2017] for a review that focuses on ODE models), which might also be transferable to the ideas presented in [Seita et al., 2018]. In this case, a sufficiently parallelized implementation of mini-batch MCMC methods could resolve the high computational burden of parameter sampling and maybe pave the way towards rigorous uncertainty assessment for large-scale ODE models, which are trained on vast datasets.

Chapter 4

Second-order derivatives in parameter estimation

In the previous chapters, we already pointed out that accurate derivative information is crucial for many local optimization strategies to work. The objective function gradient gives an indication in which direction a descent is possible, but it gives no indication about a favorable step-size for the next step. Although optimization algorithms exist which work only with gradients, such as gradient descent with backtracking line-search, it is clear that additional information opens up additional possibilities to design efficient algorithms. As second order derivatives encode the curvature of the objective function landscape, they allow to develop criteria for choosing the step-size during optimization [Nocedal and Wright, 2006]. Hence, most common optimization techniques exploit at least some kind of second order information, but mostly in an approximated fashion. As moreover the standard approach to profile computation also relies on solving a series of local optimization problems, second order derivatives are also crucial in this field.

It seems natural to assume that more accurate second order derivatives may allow a better planning of optimizer step-sizes than approximations. However, for typical ODE models in systems biology, this has not been investigated in detail. One reason for this is that accurate second order derivatives are usually expensive to compute, especially for models based on differential equations [Boiger et al., 2016]. Hence, it is often assumed that the additional computational effort for accurate second order derivatives will not be sufficiently balanced by an improved optimization performance. Beyond that, it is also a considerable effort to implement computational methods that calculate second order derivatives with more involved techniques than finite differences. For these reasons, algorithms exploiting sensitivity based second order derivatives are rarely used, with only few notable exceptions [Balsa-Canto et al., 2001, 2004].

In this chapter, we introduce second order adjoint sensitivity analysis to compute accurate Hessians of the objective function which depend on the solution of an ODE and therefore address challenge (iii) from Chapter 1.1. We show that this approach enjoys better scaling properties than currently available methods while providing accurate Hessians. We then perform a benchmark study of different methods in order to assess the benefit of exact Hessians for multi-start local optimization and profile calculation. This study is based on two example models with real measurement data taken from publications, which are commonly used as benchmark examples [Hass et al., 2019]. Additionally, we propose a novel approach for computing profiles, which efficiently exploits exact Hessians and which we show to outperform existing methods.

This chapter is based on my work in the following publications. Thus, some of its parts may be similar to those:

- **Stapor, P.**, Fröhlich, F., Hasenauer, J. (2018). Optimization and profile calculation of ODE models using second order adjoint sensitivity analysis. Bioinformatics, 34(13)
- **Stapor, P.***, Weindl, D.*, Ballnus, B., Hug, S., Loos, C., Fiedler, A., Krause, S., Hross, S., Fröhlich, F., Hasenauer, J. (2018). PESTO: Parameter EStimation TOolbox. Bioinformatics, 34(4)

4.1 Background: Second order derivatives for ODE models

4.1.1 Methods for computing second-order derivatives

A number of different methods exist to either approximate or compute the Hessian of an objective function which depends on the solution of an ODE. Excluding second order adjoint sensitivity analysis, which we introduce in this chapter, the following approaches are commonly used:

1.) The Fisher Information Matrix (FIM) [Fisher, 1922], which can be interpreted as an approximation based on observable sensitivities.

2.) Quasi-Newton methods, such as the the Broyden-Fletcher-Goldfarb-Shanno (BFGS) scheme [Goldfarb, 1970], which iteratively approximate the Hessian from gradients which are obtained during optimization.

3.) Finite differences based on gradients from first order adjoint sensitivity analysis, yielding the Hessian, up to numerical errors.

4.) Second order forward sensitivity analysis [Vassiliadis et al., 1999], yielding the Hessian with adjustable numerical accuracy.

Approximative methods

The FIM is related to the asymptotic covariance of maximum likelihood estimates [Swameye et al., 2003] and provides an approximation to the Hessian of the negative log-likelihood function. The approximation converges quadratically in the size of the residuals $(\bar{y}_{ij}^k - h_i(t_j, u^k))/\sigma_{ij}^k$ [Raue, 2013]. Assuming Gaussian measurement noise, the FIM can be computed from observable sensitivities. We recall that those are given by $s_r^{y_i}(t_j, u^k) = \frac{d}{d\theta_r} h_i(t_j, \theta, u^k)$. If the measurement noise is not parameter dependent, its computation is straight forward:

$$\mathrm{FIM}_{r,q}(\theta) = \sum_{k=1}^{n_e} \sum_{i=1}^{n_y^k} \sum_{j=1}^{n_t} \frac{s_r^{y_i}(t_j, u^k) s_q^{y_i}(t_j, u^k)}{\left(\sigma_{ij}^k\right)^2} \tag{4.1}$$

As the FIM is constructed from observable sensitivities, it requires forward sensitivity analysis, which is feasible for small- to medium-scale models, but becomes prohibitive for large-scale models. Although the FIM provides only an approximation, it is often used in optimization, as many toolboxes rely on forward sensitivity analysis and hence the FIM is obtained at almost no additional cost. If however the measurement noise is unknown, various different first-order approximations of the Hessian are possible.

The BFGS scheme is an algorithm which sequentially computes a positive-definite approximation to the Hessian during an optimization process. This approximation relies on the secant-method and only requires gradients, which are computed in each optimization step. It is hence

also applicable for large-scale models, as it can be combined with adjoint sensitivity analysis. Some variants of this algorithm circumvent computing and storing the full Hessian by directly computing the Newton step via a limited amount of stored gradients [Nocedal, 1980], which makes them even more efficient for large-scale problems. Implementations can be found in many state-of-the-art optimization toolboxes, like, e.g., Ipopt [Wächter and Biegler, 2006], or fmincon [MathWorks, 2016].

Alternative (quasi-Newton) methods, which also rely on the principle of sequentially approximating the Hessian matrix or its inverse but use different updates rules, would be symmetric rank 1 [Byrd et al., 1996], the Davidon-Fletcher-Powell formula [Fletcher and Powell, 1963], or Broyden's method [Broyden, 1965].

Finite differences

Central finite differences compute the Hessian based on perturbations in each parameter direction by a small step ϵ:

$$\frac{\partial^2 J}{\partial\theta_r \partial\theta_q}(\theta) \approx \frac{\frac{\partial J(\theta+\epsilon e_q)}{\partial\theta_r} - \frac{\partial J(\theta-\epsilon e_q)}{\partial\theta_r}}{2\epsilon} \tag{4.2}$$

where e_q is the unit vector with 1 at the q-th position. The accuracy of this method depends on the step size ϵ. Good choices of ϵ depend in turn on the error tolerances of the ODE solver and are thus not easy to determine (see Hanke and Scherzer [2001] and the references therein). Hence, using finite differences to compute derivatives often leads to less successful results in local optimization than using more accurate methods [Raue et al., 2013b, Schälte et al., 2018].

Forward sensitivity analysis

Second order forward sensitivity analysis extends, similar to first order forward sensitivity analysis, the considered ODE system, now including first order and second order derivatives of the state variables. Integrating those second order state sensitivities makes it possible to compute an analytically derived expression for the Hessian.

For the sake of readability, we will omit the indices j and k for the timepoints and experimental conditions and the summing over them in the equations of this part. At the same time, we will just sum over an index i, which enumerates all data points, comprising different observables and timepoints. The dependence on the i-th data point will be denoted by a subscript (e.g., h_i) instead of listing dependencies on t, x, θ, and u. Furthermore, we will denote, where necessary, differentiations by a transposed Nabla operator with respect to a vector v of size n_v, in the sense of

$$\nabla_v^T \colon \mathcal{C}^2(\mathbb{R},\mathbb{R}) \longrightarrow \mathcal{C}^1(\mathbb{R},\mathbb{R}^{1\times n_v}), \quad f \longmapsto \left(\frac{\partial f}{\partial v_1},\dots,\frac{\partial f}{\partial v_{n_v}}\right) \tag{4.3}$$

where we assume the corresponding functions to be sufficiently smooth (i.e., at least twice continuously differentiable). Double differentiations by $\nabla_v\nabla_v^T$ or $\nabla_v^T \otimes \nabla_v^T$ with the respective meaning, where $\otimes$ denotes a Kronecker product.

We rewrite the equation for the gradient from Equation (2.17) in our simplified notation:

$$\frac{\partial J}{\partial\theta_r} = \sum_{i=1}\left(\left(\frac{1}{\sigma_i} - \frac{(\bar{y}_i - h_i)^2}{\sigma_i^3}\right)\frac{\partial\sigma_i}{\partial\theta_r} - \frac{\bar{y}_i - h_i}{\sigma_i^2}\left(\nabla_x h_i(s_r^x)_i + \frac{\partial h_i}{\partial\theta_r}\right)\right) \tag{4.4}$$

Another differentiation with respect to θ_q gives the Hessian.

$$\begin{aligned}
\frac{\partial^2 J}{\partial\theta_r\partial\theta_q} =& \sum_{i=1}\left(-\frac{1}{\sigma_i}\frac{\partial\sigma_i}{\partial\theta_q}+3\frac{(\bar{y}_i-h_i)^2}{\sigma_i^4}\frac{\partial\sigma_i}{\partial\theta_q}+2\frac{\bar{y}_i-h_i}{\sigma_i^3}\left(\left(\left(s_q^x\right)_i\right)^T\nabla_x h_i+\frac{\partial h_i}{\partial\theta_q}\right)\right)\frac{\partial\sigma_i}{\partial\theta_r}\\
&+\sum_{i=1}\left(\left(\frac{1}{\sigma_i}-\frac{(\bar{y}_i-h_i)^2}{\sigma_i^3}\right)\frac{\partial^2\sigma_i}{\partial\theta_r\partial\theta_q}-\frac{\bar{y}_i-h_i}{\sigma_i^2}\left(\left(\left(s_q^x\right)_i\right)^T\frac{\partial\nabla_x h_i}{\partial\theta_r}+\frac{\partial^2 h_i}{\partial\theta_r\partial\theta_q}\right)\right)\\
&+\sum_{i=1}\left(\frac{1}{\sigma_i^2}\left(\left(s_q^x\right)_i^T\nabla_x h_i+\frac{\partial h_i}{\partial\theta_q}\right)+2\frac{\bar{y}_i-h_i}{\sigma_i^3}\frac{\partial\sigma_i}{\partial\theta_q}\right)\frac{\partial h_i}{\partial\theta_r}\\
&+\sum_{i=1}\left(\frac{1}{\sigma_i^2}\left(\left(s_q^x\right)_i^T\nabla_x h_i+\frac{\partial h_i}{\partial\theta_q}\right)+2\frac{\bar{y}_i-h_i}{\sigma_i^3}\frac{\partial\sigma_i}{\partial\theta_q}\right)\left(\nabla_x^T h_i(s_r^x)_i\right)\\
&-\sum_{i=1}\frac{\bar{y}_i-h_i}{\sigma_i^2}\left(\left(\left(s_q^x\right)_i^T\nabla_x^T\nabla_x h_i+\frac{\partial\nabla_x^T h_i}{\partial\theta_q}\right)(s_r^x)_i+\nabla_x^T h_i(s_{r,q}^x)_i\right) \qquad (4.5)
\end{aligned}$$

In this expression, the second order state sensitivities $(s_{r,q}^x)_i$ show up, which have to be computed. The corresponding ODE system is given by:

$$\dot{s}_{r,q}^x=\left(\nabla_x^T f\right)s_{r,q}^x+\left(\left(\nabla_x^T\otimes\nabla_x^T\right)f\right)\left(s_r^x\otimes s_q^x\right)+\frac{\partial\nabla_x^T f}{\partial\theta_r}s_q^x+\frac{\partial\nabla_x^T f}{\partial\theta_q}s_r^x+\frac{\partial^2 f}{\partial\theta_r\partial\theta_q} \qquad (4.6)$$

The whole ODE system, which has to be integrated for second order forward sensitivities, has $(n_\theta^2+n_\theta+1)n_x$ equations. If the symmetry of the Hessian is exploited, it can be reduced to $(n_\theta/2+1)(n_\theta+1)n_x$ equations, but the computational complexity of the problem still scales quadratically in the number of parameters and linearly in the number of state variables, which limits this method to small-scale applications. However, second order forward sensitivity analysis yields accurate Hessians, since the error of the second order state sensitivities can be controlled during ODE integration.

4.1.2 Exploiting second order derivatives for parameter optimization

In local optimization, derivatives may be exploited for computing a search direction and for determining the size of the next optimization step [Boyd and Vandenberghe, 2004, Nocedal and Wright, 2006]. As search direction, an optimizer can use either the negative gradient (yielding gradient descent), or perform a (dampened) Newton step (or an approximation thereof). Indeed, most implementations interpolate between those two options, yielding techniques such as the dogleg method or Krylov subspace methods [Nocedal and Wright, 2006]. All of these techniques need the objective function gradient to work and, with exception of gradient descent, they all use some kind of Hessian information. Additionally, all of them use some kind of second order derivatives, with the exception of plain gradient descent.

For computing the step-size, trust-region [Sorensen, 1982] or line-search methods [Armijo, 1966] are employed. For trust-region, many approaches exist, while line-search is typically performed using an exact step length or by backtracking. Again, all methods use first and second order derivatives or approximations thereof: They compute a local model of the objective function, which approximates the true objective function. Then, this local model is optimized either along the search direction or within the trust-region (Figure 4.1). Optimizing this local model is often called the inner problem of an optimization step, while the outer problem is the original optimization task [Nocedal and Wright, 2006].

Beyond the search direction and the step-size, optimizers typically have to deal with box-

constraints, which are typical for ODE models in systems biology. Here, two methods are common: reflection based methods [Coleman and Li, 1996], which rescale the step-size depending on the distance to the boundary of the box, and interior-point methods [Boyd and Vandenberghe, 2004], which add a penalizing barrier function to the original optimization problem, which is relaxed during the optimization process.

In the beginning of a local optimization, it is most important that the optimizer chooses a descent direction, but the accuracy of step-size computation is less important, as long as the step-size is not too long. Hence, the accuracy of the local model (or the inner problem) is less crucial in this part of the optimization process. However, although approximative derivatives may suffice in the beginning, the closer the optimizer gets to a local optimum, the more important becomes the agreement of the inner and the outer problem. Hence, in the last part of the optimization problem, the accuracy of the derivatives, also the second order derivatives, is of major importance [Nocedal and Wright, 2006]. Especially in this phase, the computation of exact Hessians is likely to be beneficial.

Most computational toolboxes only use approximations of the Hessian matrix by default, but can be provided with exact second order derivatives, if computed by the user. The MATLAB optimization toolbox [MathWorks, 2016], which was used for the studies in this chapter, allows the specification of three algorithms, which are commonly used for optimization problems in systems biology:

1. fmincon – interior-point (based on [Byrd et al., 2000]),

2. fmincon – trust-region-reflective (based on [Branch et al., 1999, Coleman and Li, 1996]),

3. lsqnonlin – trust-region-reflective (based on [Coleman and Li, 1996]).

The first two expect a scalar valued objective function and allow the user to provide the Hessian, but use approximations by default (a BFGS approximation in the first, a conjugate gradient method in the second case). The latter, lsqnonlin, expects the user to provide residuals and their sensitivities and constructs a predefined Hessian approximation based on a Gauss-Newton algorithm. This restricts its usability to a specific kind of objective functions (weighted least squares), but exploits the problem structure to a higher degree.

It is now an interesting – and non-trivial – question, which of these algorithms works best in which situation and whether – and if so, when – providing them with exact second order derivatives is beneficial. In other words: Does providing accurate second order derivatives improve the convergence of commonly used optimizer implementations? And if this is the case: Are second order adjoint methods efficient enough such that the computation of accurate second order derivatives pays off?

4.1.3 Profile likelihood computation

Profile likelihood (or short profile) calculation [Kreutz et al., 2013] as discussed in the Chapter 2, is a common method to quantify uncertainties of parameters or predictions and assess practical identifiability of small- to medium-scale models [Kreutz et al., 2013]. We repeat Equation (2.27) from Chapter 2 here, which defines the points on the profile of parameter θ_r at $\theta_r = c$:

$$\mathrm{PL}_{\theta_r}(c) = \max_{\substack{\theta_r = c \\ \theta \in \Omega}} \mathcal{L}_D(\theta) \propto \exp\left(- \min_{\substack{\theta_r = c \\ \theta \in \Omega}} J(\theta) \right). \tag{2.27}$$

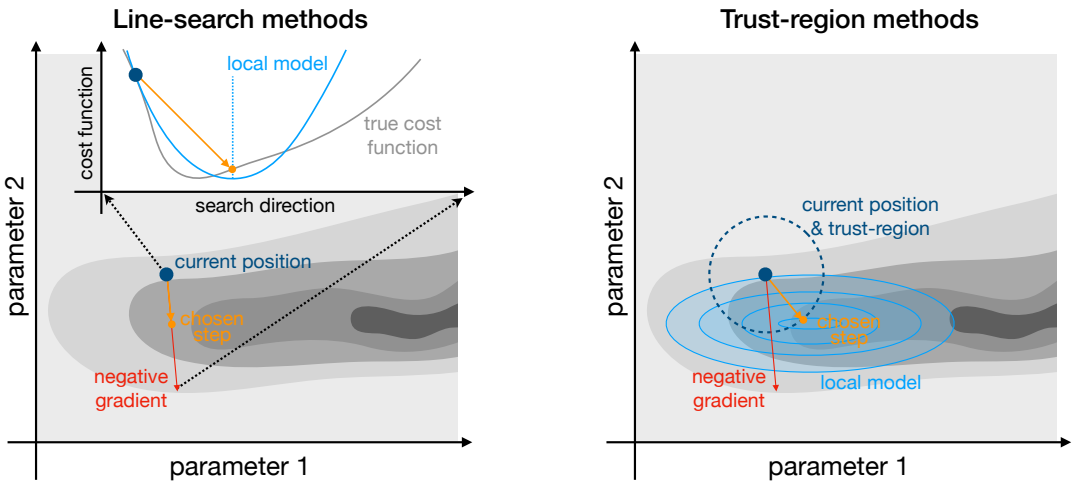

Figure 4.1: Methods for determining step-sizes during a local optimization. Left: Line-search methods first fix a search direction (in this case the negative gradient). Then, typically a local model of the cost function along the search direction is constructed from a Hessian-vector product or an approximation of the Hessian. Optimizing the local model gives the step-size. Right: Trust-region methods also construct a local model of the cost function, which is usually two- or higher dimensional. They optimize the local model within the trust-region (e.g., a ball) around the current position, to compute the search direction and the step-size at the same time and update the trust-region radius for the next step.

Currently, two approaches exist to compute profiles for model properties: An optimization-based and an integration-based approach (Figure 4.2).

The optimization-based approach (as implemented in [Raue et al., 2015]) exploits the fact that the points on a profile are conditionally optimal and computes the profile for θ_r via a sequence of optimization problems [Raue et al., 2009]. In each step, all parameters besides θ_r are optimized while θ_r is fixed to a value c. For each new step, c either increased or decreased (depending on the profile calculation direction) and a new (local) optimization is initialized based on the previously found parameter values. As long as the function $\mathrm{PL}_{\theta_r}(c)$ is smooth, this initial point will be close to the optimum and it suffices to run one local optimization which typically converges within few iterations. Yet, as many optimizations have to be performed to obtain a full profile and usually all profiles have to be computed, this process is computationally demanding.

An efficient alternative to the optimization-based is the integration-based approach [Chen and Jennrich, 1996, 2002] (as implemented in [Kaschek et al., 2019]), which circumvents repeated optimization by using a dynamical system which evolves along the optimal path (2.27). For a model property $g\colon \Omega \longrightarrow \mathbb{R}$ (e.g., $g(\theta) = \theta_r$), which is constrained to $g(\theta) = c$, the dynamical system is obtained by differentiating the optimality condition with respect to the value of the constraint c. This yields:

$$\nabla_\theta J(\theta) + \lambda \nabla_\theta g(\theta) = 0 \tag{4.7}$$

Here, λ is a Lagrange multiplier. After differentiation, we get the following differential algebraic equation (DAE):

$$\begin{pmatrix} \nabla_\theta \nabla_\theta^T J + \nabla_\theta \nabla_\theta^T g & \nabla_\theta g \\ \nabla_\theta^T g & 0 \end{pmatrix} \begin{pmatrix} \frac{d\theta}{dc} \\ \frac{d\lambda}{dc} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{4.8}$$

The matrix on the left hand side is called the mass matrix. Multiplying this equation with its inverse from the left (assuming it exists) yields an ODE formulation. The resulting ODE can in principle be integrated with established differential equation solvers if the Hessian $\nabla_\theta^2 J$

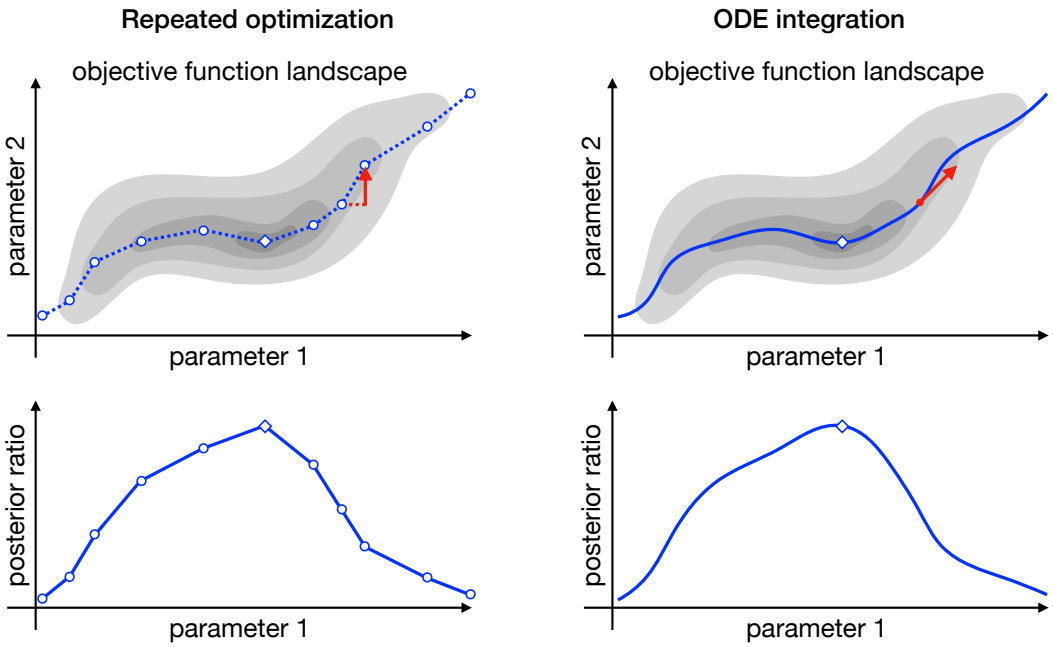

Figure 4.2: Methods for computing profile likelihoods for a parameter. Left: The (standard) optimization-based approach starts from the global optimum and moves along a parameter axis in small intervals. It uses a local optimization method to optimize all but the profiled parameter. Right: The integration-based method defines an ODE which describes the profile path through parameter space starting at the global optimum. Integrating this ODE yields the profile path and hence the profile likelihood.

or an approximation thereof is computed [Chen and Jennrich, 2002]. In principle, also a direct integration of the DAE would be possible, but it has been shown that using the ODE formulation tends to be more efficient [Boiger et al., 2016]. However, integrating this ODE is non-trivial, as the mass matrix may have singularities, which may lead to discontinuities in the profile path. This results in small step sizes during ODE integration. Moreover, the trajectory of the ODE solver may deviate from the true profile path of Equation (2.27) due to numerical errors or approximations being used. These aspects can make the integration-based method fail, or, if error tolerances are relaxed or ODE integration is even carried out with a fixed step-size, may yield incorrect results. moreover, as computing the Hessian is a challenging task, this method is currently less used for profile computation. However, in cases in which optimization is intricate, the integration-based method may even work better than the optimization-based approach.

4.2 Contribution: Second order adjoint sensitivity analysis

To assess the potential of second order adjoint sensitivity analysis, the corresponding equations for time-discrete measurements were derived and implemented in the ODE solver toolbox AMICI. We compared accuracy and computation time for the Hessian with those from established methods. Furthermore, we evaluated parameter optimization and profile calculation methods using Hessians for published models.

4.2.1 Derivation and implementation

To the best of the author's knowledge, second order adjoint sensitivity analysis has not been used together with time-discrete measurements and the corresponding equations have not been

derived so far. Furthermore, second order adjoint sensitivity analysis has, as far as the author can tell, not yet been applied in the field of systems and computational biology and no ready-to-use implementation of it is available.

Along the lines of first order adjoint sensitivity analysis, which is explained in Chapter 2, second order adjoint sensitivity analysis computes Hessians with better scaling properties than second order forward sensitivity analysis. Again, the error of the Hessian can be controlled during ODE integration, yielding as accurate results as those from second order forward sensitivity analysis. To compute Hessians, the idea of the adjoint method is applied with an additional differentiation with respect to θ. In a first step, the system defined by (2.19) is integrated forward in time. Subsequently, the corresponding adjoint system is integrated backwards in time, using the information from the forward simulation. This system consists of the original adjoint system plus the n_θ derivatives of the adjoint state with respect to θ_r.

Equations of the second order adjoint system

To derive the equations for the second order adjoint system, we first look at the equation which defines the contribution of the j-th timepoint to the gradient, which we call J_j. Again, we omit all dependencies (except those of t, as they are important for the derivation) and neglect the sum over the experimental conditions k:

$$\begin{aligned}
\frac{\partial J_j(\theta)}{\partial \theta_r} =& \sum_{i=1}^{n_y} \left(\left(\frac{1}{\sigma_{ij}} - \frac{(\bar{y}_{ij} - h_i(t_j))^2}{\sigma_{ij}^3} \right) \frac{\partial \sigma_{ij}}{\partial \theta_r} - \frac{\bar{y}_{ij} - h_i(t_j)}{\sigma_{ij}^2} \left(\nabla_x^T h_i(t_j) s_r^x(t_j) + \frac{\partial h_i(t_j)}{\partial \theta_r} \right) \right) \\
=& \sum_{i=1}^{n_y} \left(\left(\frac{1}{\sigma_{ij}} - \frac{(\bar{y}_{ij} - h_i(t_j))^2}{\sigma_{ij}^3} \right) \frac{\partial \sigma_{ij}}{\partial \theta_r} - \frac{\bar{y}_{ij} - h_i(t_j)}{\sigma_{ij}^2} \left(\nabla_x^T h_i(t_j) s_r^x(t_j) + \frac{\partial h_i(t_j)}{\partial \theta_r} \right) \right) \\
&+ \int_{t_j}^{t_{j+1}} p(t)^T \left(\underbrace{\dot{s}_r^x(t_j) - \frac{\partial f(t)}{\partial \theta_r} - \nabla_x^T f(t) s_r^x(t_j)}_{=0} \right) dt \qquad (4.9) \\
=& \sum_{i=1}^{n_y} \left(\left(\frac{1}{\sigma_{ij}} - \frac{(\bar{y}_{ij} - h_i(t_j))^2}{\sigma_{ij}^3} \right) \frac{\partial \sigma_{ij}}{\partial \theta_r} - \frac{\bar{y}_{ij} - h_i(t_j)}{\sigma_{ij}^2} \frac{\partial h_i(t_j)}{\partial \theta_r} \right) - \int_{t_j}^{t_{j+1}} p(t)^T \frac{\partial f(t)}{\partial \theta_r} dt \\
&- \int_{t_j}^{t_{j+1}} \left(\dot{p}(t)^T + p(t)^T \nabla_x^T f(t) \right) s_r^x(t) dt - \sum_{i=1}^{n_y} \frac{\bar{y}_{ij} - h_i(t_j)}{\sigma_{ij}^2} \nabla_x^T h_i(t_j) s_r^x(t_j) \\
&+ \lim_{t \to t_{j+1}^-} p(t)^T s_r^x(t) - \lim_{t \to t_j^+} p(t)^T s_r^x(t) \qquad (4.10)
\end{aligned}$$

In the first step, we add the ODE of the first order state sensitivities, i.e., a term equal to $0 \in \mathbb{R}^{n_x}$, take the scalar product with another vector $p(t)$ (which will be the adjoint state later on), and integrate over the time interval $[t_j, t_{j+1}]$. In the last step, we use integration by parts in order to get the time derivative of the adjoint state and regroup all expressions with state sensitivities together.

To compute the Hessian, we differentiate Equation (4.10) with respect to θ_q and get an

expression for the contribution of the j-th timepoint to the Hessian:

$$
\begin{aligned}
&\frac{\partial^2 J_j(\theta)}{\partial\theta_r\partial\theta_q} \\
=&\sum_{i=1}^{n_y}\left(-\frac{1}{\sigma_{ij}}\frac{\partial\sigma_{ij}}{\partial\theta_q}+3\frac{(\bar{y}_{ij}-h_i(t_j))^2}{\sigma_{ij}^4}\frac{\partial\sigma_{ij}}{\partial\theta_q}+2\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^3}\left(\left(s_q^x(t_j)\right)^T\nabla_x h_i(t_j)+\frac{\partial h_i(t_j)}{\partial\theta_q}\right)\right)\frac{\partial\sigma_{ij}}{\partial\theta_r} \\
&+\sum_{i=1}^{n_y}\left(\left(\frac{1}{\sigma_{ij}}-\frac{(\bar{y}_{ij}-h_i(t_j))^2}{\sigma_{ij}^3}\right)\frac{\partial^2\sigma_{ij}}{\partial\theta_r\partial\theta_q}-\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^2}\left(\left(s_q^x(t_j)\right)^T\frac{\partial\nabla_x h_i(t_j)}{\partial\theta_r}+\frac{\partial^2 h_i(t_j)}{\partial\theta_r\partial\theta_q}\right)\right) \\
&+\sum_{i=1}^{n_y}\left(\frac{1}{\sigma_{ij}^2}\left(\left(s_q^x(t_j)\right)^T\nabla_x h_i(t_j)+\frac{\partial h_i(t_j)}{\partial\theta_q}\right)+2\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^3}\frac{\partial\sigma_{ij}}{\partial\theta_q}\right)\frac{\partial h_i(t_j)}{\partial\theta_r} \\
&-\int_{t_j}^{t_{j+1}}\left(\frac{\partial p(t)^T}{\partial\theta_q}\frac{\partial f(t)}{\partial\theta_r}+p(t)^T\frac{\partial\nabla_x^T f(t)}{\partial\theta_r}s_q^x(t)+p(t)^T\frac{\partial^2 f(t)}{\partial\theta_r\partial\theta_q}\right)dt \\
&+\left(\sum_{i=1}^{n_y}\left(\frac{1}{\sigma_{ij}^2}\left(\nabla_x^T h_i(t_j)s_q^x(t_j)+\frac{\partial h_i(t_j)}{\partial\theta_q}\right)+2\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^3}\frac{\partial\sigma_{ij}}{\partial\theta_q}\right)\nabla_x^T h_i(t_j)\right. \\
&\left.-\sum_{i=1}^{n_y}\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^2}\left(\left(s_q^x(t_j)\right)^T\nabla_x^T\nabla_x h_i(t_j)+\frac{\partial\nabla_x^T h_i(t_j)}{\partial\theta_q}\right)\right)s_r^x(t_j) \\
&-\int_{t_j}^{t_{j+1}}\left(\frac{\partial\dot{p}(t)^T}{\partial\theta_q}+\frac{\partial p(t)^T}{\partial\theta_q}\nabla_x^T f(t)+p(t)^T\left((\nabla_x^T\otimes\nabla_x^T)f(t)(\mathbb{I}_n\otimes s_q^x(t))+\frac{\partial\nabla_x^T f(t)}{\partial\theta_q}\right)\right)s_r^x(t)dt \\
&-\int_{t_j}^{t_{j+1}}\left(\dot{p}(t)^T+p(t)^T\nabla_x^T f(t)\right)s_{r,q}^x(t)dt-\sum_{i=1}^{n_y}\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^2}\nabla_x^T h_i(t_j)s_{r,q}^x(t_j) \\
&+\lim_{t\to t_{j+1}^-}p(t)^T s_{r,q}^x(t)-\lim_{t\to t_j^+}p(t)^T s_{r,q}^x(t)+\lim_{t\to t_{j+1}^-}\frac{\partial p(t)^T}{\partial\theta_q}s_r^x(t)-\lim_{t\to t_j^+}\frac{\partial p(t)^T}{\partial\theta_q}s_r^x(t)
\end{aligned}
\tag{4.11}
$$

We can use Equation (4.11) to define the equations for the second order adjoint state analogously to the first order adjoint state:

$$
\frac{\partial\dot{p}(t)}{\partial\theta_q}=-\frac{\partial p(t)}{\partial\theta_q}\nabla_x(f(t)^T)-\left((s_q^x(t)\otimes\mathbb{I}_n)(\nabla_x\otimes\nabla_x)(f(t)^T)+\frac{\partial\nabla_x(f(t)^T)}{\partial\theta_q}\right)p(t) \tag{4.12}
$$

$$
\begin{aligned}
\frac{\partial p(t_j)}{\partial\theta_q}=&\lim_{t\to t_j^+}\frac{\partial p(t)}{\partial\theta_q}+\sum_{i=1}^{n_y}\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^2}\left(\nabla_x^T\nabla_x h_i(t_j)s_q^x(t_j)+\frac{\partial\nabla_x h_i(t_j)}{\partial\theta_q}\right) \\
&-\sum_{i=1}^{n_y}\left(\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^3}\frac{\partial\sigma_{ij}}{\partial\theta_q}\nabla_x h_i(t_j)+\frac{1}{\sigma_{ij}^2}\left(\nabla_x^T h_i(t_j)s_q^x(t_j)+\frac{\partial h_i(t_j)}{\partial\theta_q}\right)\nabla_x h_i(t_j)\right)
\end{aligned}
\tag{4.13}
$$

$$
\frac{\partial p(t_{n_t})^T}{\partial\theta_q}=0 \tag{4.14}
$$

Using these equations, those from the adjoint state and summing up over all time points finally

yields the Hessian:

$$
\begin{aligned}
&\frac{\partial^2 J_j(\theta)}{\partial\theta_r\partial\theta_q} \\
=&\sum_{j=1}^{n_t}\sum_{i=1}^{n_y}\left(-\frac{1}{\sigma_{ij}}\frac{\partial\sigma_{ij}}{\partial\theta_q}+3\frac{(\bar{y}_{ij}-h_i(t_j))^2}{\sigma_{ij}^4}\frac{\partial\sigma_{ij}}{\partial\theta_q}+2\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^3}\left(\nabla_x^T h_i(t_j)s_q^x(t_j)+\frac{\partial h_i(t_j)}{\partial\theta_q}\right)\right)\frac{\partial\sigma_{ij}}{\partial\theta_r} \\
&+\sum_{j=1}^{n_t}\sum_{i=1}^{n_y}\left(\left(\frac{1}{\sigma_{ij}}-\frac{(\bar{y}_{ij}-h_i(t_j))^2}{\sigma_{ij}^3}\right)\frac{\partial^2\sigma_{ij}}{\partial\theta_r\partial\theta_q}-\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^2}\left(\frac{\partial\nabla_x^T h_i(t_j)}{\partial\theta_r}s_q^x(t_j)+\frac{\partial^2 h_i(t_j)}{\partial\theta_r\partial\theta_q}\right)\right) \\
&+\sum_{j=1}^{n_t}\sum_{i=1}^{n_y}\left(\frac{1}{\sigma_{ij}^2}\left(\nabla_x^T h_i(t_j)s_q^x(t_j)+\frac{\partial h_i(t_j)}{\partial\theta_q}\right)+2\frac{\bar{y}_{ij}-h_i(t_j)}{\sigma_{ij}^3}\frac{\partial\sigma_{ij}}{\partial\theta_q}\right)\frac{\partial h_i(t_j)}{\partial\theta_r} \\
&-\int_{t_0}^{t_{n_t}}\left(\frac{\partial p(t)^T}{\partial\theta_q}\frac{\partial f(t)}{\partial\theta_r}+p(t)^T\frac{\partial\nabla_x^T f(t)}{\partial\theta_r}s_q^x(t)+p(t)^T\frac{\partial^2 f(t)}{\partial\theta_r\partial\theta_q}\right)dt-p(t_0)^T s_{r,q}^x(t_0)-\frac{\partial p(t_0)^T}{\partial\theta_q}s_r^x(t_0)
\end{aligned}
\tag{4.15}
$$

The computation of the Hessian by second order adjoint sensitivity analysis requires solving two ODE systems of size $n_x(1+n_\theta)$ and n_θ^2 one dimensional quadratures. Again, these quadratures are fast to evaluate compared with the ODE systems. Hence, the scaling behavior is expected to be almost linear in the number of state variables and the number of parameters.

Implementation in the toolbox AMICI

The code for computing gradients by first order forward and adjoint sensitivity analysis has already been implemented in the MATLAB, Python, and C++ based toolbox AMICI (Advanced Multilanguage Interface to CVODES and IDAS, Fröhlich et al. [2019]). In its core part, AMICI relies on the ODE solver CVODES [Serban and Hindmarsh, 2005] from the SUNDIALS package [Hindmarsh et al., 2005]. The implementation of the presented equations for the computation of Hessians by second order adjoint sensitivity analysis is part of the contribution of this thesis.

AMICI generates C++ code for ODE integration that can be interfaced from MATLAB (and Python) to ensure computational efficiency. This is important, since during parameter estimation, ODE integration is the computationally most expensive part, although being carried out in a compiled language. So far, the generation of the C++ files necessary for second order adjoint sensitivity analysis is only available via the MATLAB interface of AMICI. A refactoring of the corresponding code in Python 3 is envisaged for the near future.

Beyond the possibility to compute the full Hessian, the MATLAB interface of AMICI also allows to compute directional second order derivatives, i.e., Hessian vector products, directly. As this method does not even need the full first order forward sensitivity matrix, it scales only weakly in the number of parameters and would hence also be applicable to large scale models [Özyurt and Barton, 2005]. Also this implementation has been a part of this thesis.

4.2.2 A hybrid method for profile calculation

Switching between methods

For profile computation, we propose a hybrid method which switches between the optimization- and the integration-based approach depending on which method is more promising in the current situation. This hybrid optimization- and integration-based approach employs by default the

integration-based technique using a high-order Adams-Bashforth scheme [Shampine and Reichelt, 1997], which can handle moderately stiff ODE systems. If the mass matrix in (4.8) is singular, a (Moore-Penrose) pseudo inverse with adjustable conditioning is used, which allows to balance the stiffness of the ODE against the accuracy with which the profile path is followed. However, other (low-rank) approximations of the inverse would be possible and might be beneficial [Chung et al., 2015]. If the step size falls below a previously defined threshold, integration will be stopped and a few optimization-based steps are carried out to circumvent numerical integration problems. This generally accelerates profile computation, since the integration-based method is usually the faster alternative, but becomes slower if the profile ODE becomes stiff. After some optimization based steps, ODE integration is reinitialized at the profile path. During profile integration, the remaining gradient is monitored. If it exceeds a predefined threshold, one single optimization step is started to regain the profile path on which profile integration reinitialized.

Implementation in the toolbox PESTO

The algorithm for hybrid profile calculation was implemented in the MATLAB toolbox PESTO (Parameter EStimation TOolbox, Stapor et al. [2018b]). The code version used for the study in this thesis is available via Zenodo, at https://doi.org/10.5281/zenodo.1162326. An updated version of the code is available on Github.

The implementation allows the use of either user-provided Hessians (which may either be approximations, such as the FIM, or accurate Hessian from second order sensitivity analysis), Hessians based on finite-differences from user-provided gradients or objective function values, and Hessian approximations based on quasi-Newton schemes (BFGS, SR1, or DFP), which are automatically computed and updated by PESTO.

4.2.3 Application examples

For the assessment of the methods, we considered five published models and corresponding datasets (called M1 - M5). The models possess 3 to 26 state variables, 9 to 116 unknown parameters and a range of dataset sizes and identifiability properties. Four models describe signal transduction processes in mammalian cells, one describes the central carbon metabolism of E. Coli. An overview about the model properties is provided in Table 4.1.

Table 4.1: Overview of considered ODE models and their properties

	M1[1]	**M2**[2]	**M3**[3]	**M4**[4]	**M5**[5]
State variables	6	3	9	18	26
Parameters	9	28	16	116	86
Time points	8	7	16	51	16
Conditions	1	3	1	1	10
Data points	24	72	46	110	960
modeled system	Epo receptor signaling	RAF/MEK/ERK signaling	JAK/STAT signaling	E. Coli carbon metabolism	EGF, TNF signaling

[1] Model adapted from Becker et al. [2010]
[2] Model adapted from Fiedler et al. [2016]
[3] Model adapted from Swameye et al. [2003]
[4] Model adapted from Chassagnole et al. [2002]
[5] Model adapted from MacNamara et al. [2012]

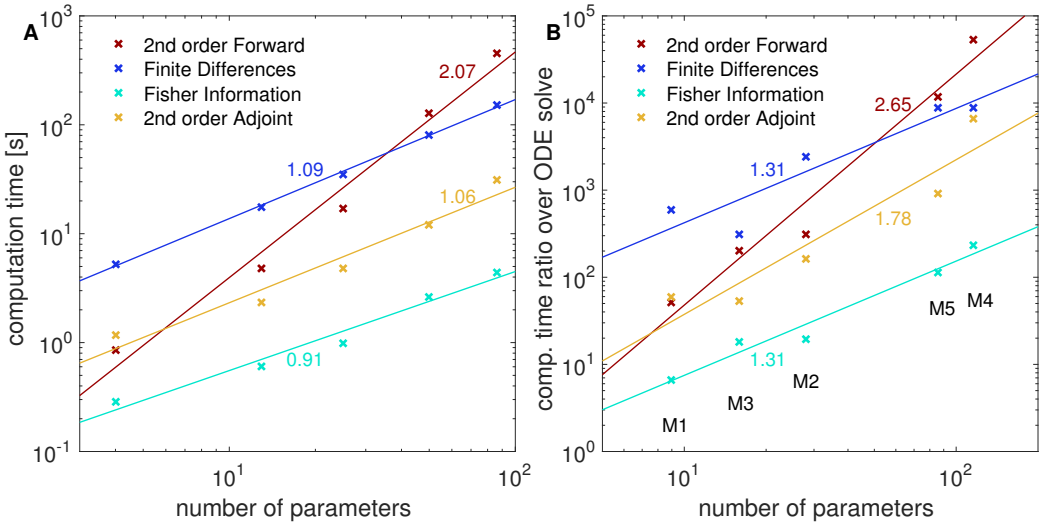

Figure 4.3: Scaling of computation times of the four investigated methods to compute or approximate the Hessian (at global optimum for each model) including linear fits and their slopes. All reported computation times were averaged over 10 runs. A) For model M5, the number of parameters was fixed to different values. B) The ratio of the computation times for Hessians or its approximation over the computation time for solving the original ODE is given for the five models from Table 4.1. This figure is taken from the author's publication [Stapor et al., 2018a], Figure 1.

4.2.4 Hessian computation

Scaling behavior of the method

To verify the theoretical scaling of the discussed methods, we evaluated the computation times for the model with the largest number of state variables (M5). This evaluation revealed that the practical scaling rates were close to their theoretical predictions (Figure 4.3 A). Second order adjoint sensitivity analysis, FIM, and finite differences based on first order adjoint sensitivity analysis exhibited a roughly linear scaling with respect to the number of parameters. Second order forward sensitivity analysis exhibited the predicted quadratic scaling. The FIM showed the lowest computation time for all models. The proposed approach, second order adjoint sensitivity analysis, was the fastest method to compute the true Hessian, taking in average about 4 times as long to compute as the FIM.

We also evaluated whether the same scaling holds across models (Figure 4.3 B). Interestingly, we found similar but slightly higher slopes for all considered methods, although the number of state variables between models differs substantially. This suggests that in practice the number of parameters is indeed a dominating factor. Overall, second order adjoint sensitivity analysis was the most efficient method for the evaluation of the Hessian.

Accuracy of the Hessian matrix

To assess the accuracy of Hessians and their approximations provided by the different methods, we compared the results at the global optimum. In general, we observed a good agreement of Hessians computed using second order adjoint and forward sensitivity analysis (Figure 4.4 A). For the Hessian computed by finite differences, we found – as expected – numerical errors (Figure 4.4 B), which depended non-trivially on the combination of ODE solver accuracy and the step size of the finite differences. The FIM usually differed substantially from the Hessians, even

though this approximation is often considered to be good close to a local optimum (Figure 4.4 C).

Since most optimization algorithms internally compute a Newton-type update $\Delta\theta = -H^{-1}g$, in which H is the Hessian and g is the gradient, we also evaluated the quality of the Hessian pre-image. For this purpose, we compared the inverses of the regularized Hessians computed with different methods with the one computed using second order adjoint sensitivity analysis in the operator norm. If the Hessian was not invertible, we used the Moore-Penrose-Pseudoinverse. This analysis revealed that the pre-images from second order forward and adjoint sensitivity analysis coincide well, whereas those from finite differences and the FIM differed substantially from the results based on second order sensitivity analysis (Figure 4.4 D).

In combination, our assessment of scaling and accuracy revealed that second order adjoint sensitivity analysis provides the most scalable approach to obtain accurate Hessians. Rough approximations of the Hessian in terms of the FIM could however be computed at a lower computational cost.

4.2.5 Application 1: Parameter optimization

As our results revealed a trade-off between accuracy and computation time for computing Hessians, we investigated how this affects different optimization algorithms. To this end, we compared optimization with different setups: Newton and quasi-Newton variants of the interior-point algorithm and the trust-region-reflective algorithm, which are implemented in the MATLAB routine fmincon:

- Residuals and their sensitivities were computed with first order forward sensitivity analysis and provided to the least-squares algorithm lsqnonlin, which used the trust-region-reflective algorithm for step-size computation and constraint handling.
- Gradient and FIM were computed using first order forward sensitivity analysis and provided to fmincon, which used the trust-region-reflective algorithm for step-size computation and constraint handling.
- Gradient and Hessian were computed with second order adjoint sensitivity analysis. A positive definite transformation of the Hessian was provided to fmincon, using the trust-region-reflective algorithm for step-size computation and constraint handling (which needs a strictly positive definite Hessian to work, which we achieved by enforcing a lower threshold for the eigenvalues of the Hessian).
- Gradients were computed using first order forward sensitivity analysis and provided to fmincon, using the interior-point algorithm for constraint handling with a BFGS approximation of the Hessian.
- Gradient and FIM were computed with first order forward sensitivity analysis and provided to fmincon, using the interior-point algorithm for constraint handling.
- Gradients and Hessians were computed with second order adjoint sensitivity analysis and provided to fmincon, using the interior-point algorithm for constraint handling.

The optimization study was carried out using the MATLAB toolbox PESTO for the models M2 and M3. Those two models were chosen, since they are small enough for a rigorous benchmark study, but exhibit dynamics which are more involved than simple mass action kinetics and

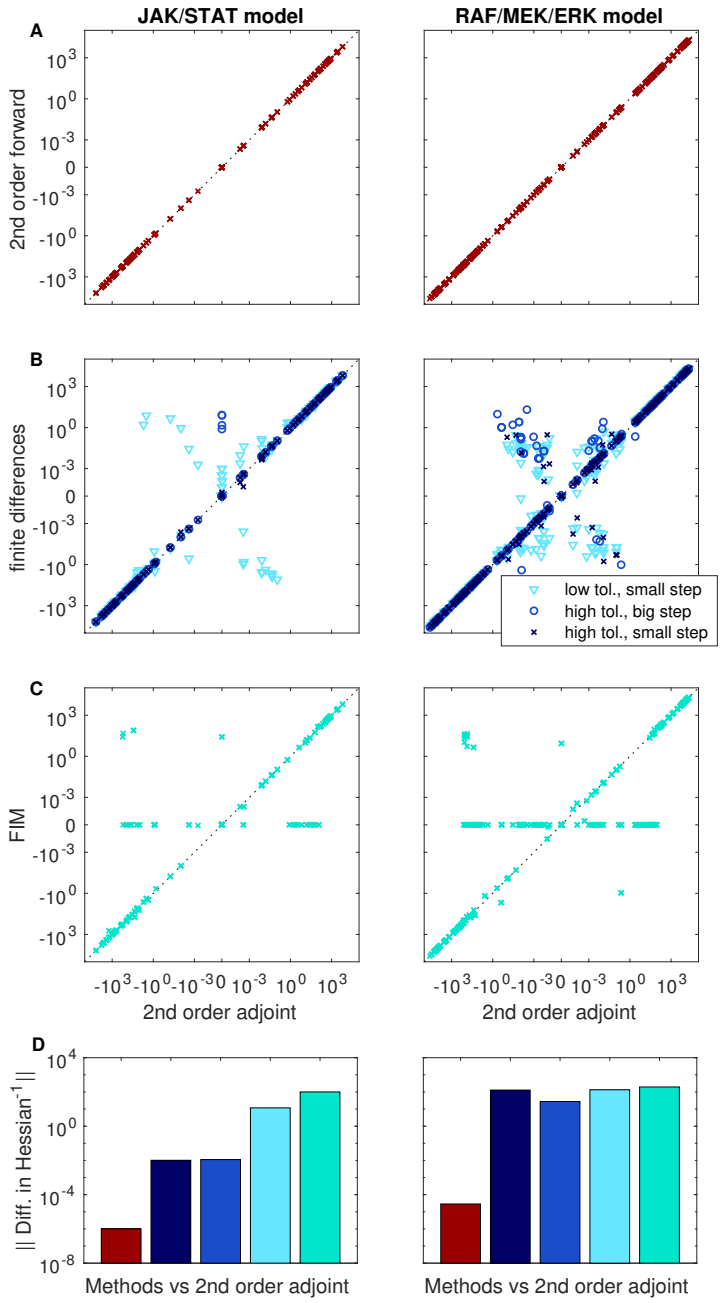

Figure 4.4: Accuracy of different methods to compute or approximate the Hessian at the global optimum for the models M2 and M3. Each point represents the numerical value of one Hessian entry as computed by two different methods: A) second order forward analysis vs. second order adjoint analysis. B) finite differences (different finite difference step sizes and ODE solver tolerances were considered) vs. second order adjoint sensitivity analysis. C) Fisher information matrix vs. second order adjoint analysis. All computations were carried out with relative and absolute tolerances set to 10^{-11} and 10^{-14}, respectively. For finite differences, lower accuracies of 10^{-7} and 10^{-10} were tested, together with the step sizes 10^{-5} and 10^{-2}. D) Operator norm of the differences in the inverse of the regularized Hessians computed with second order adjoint sensitivity analysis compared to five considered methods (second forward sensitivity analysis, finite differences with different tolerances and step sizes, Fisher information matrix). This figure is taken from the author's publication [Stapor et al., 2018a], Figure 2.

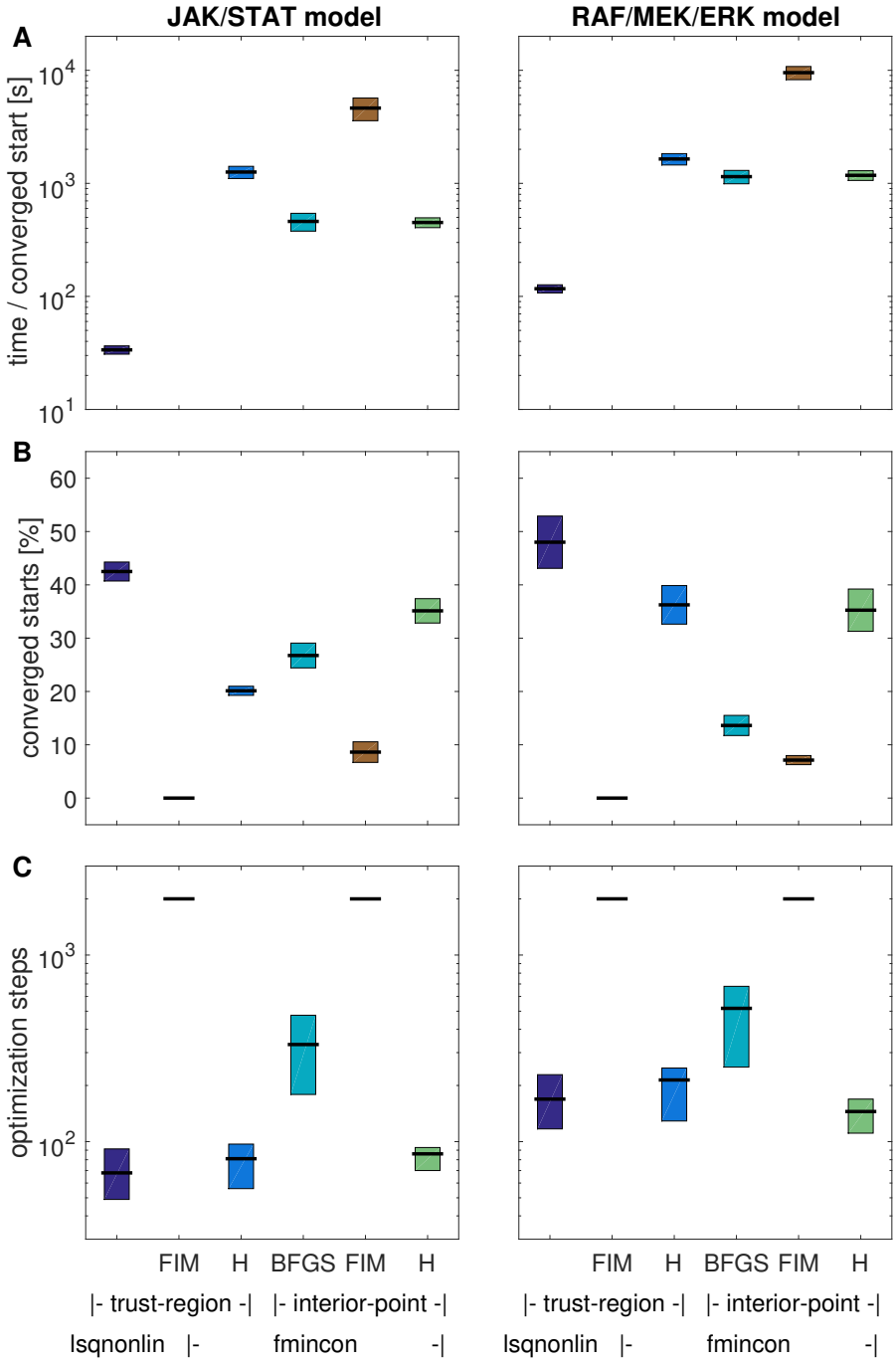

Figure 4.5: Performance measures of different local optimization methods (lsqnonlin with trust-region-reflective algorithm and fmincon with trust-region-reflective and interior-point algorithm, using either Hessians (H), Fisher information matrix (FIM), or the BFGS scheme). The multi-start optimization was carried out multiple times using different starting points for the local optimizations. Mean and standard deviation for A) the ratio of computation time over converged optimization starts and B) the number of converged starts are shown. C) Median and standard deviation of the number of steps over all optimization runs. This figure is taken from the author's publication [Stapor et al., 2018a], Figure 3.

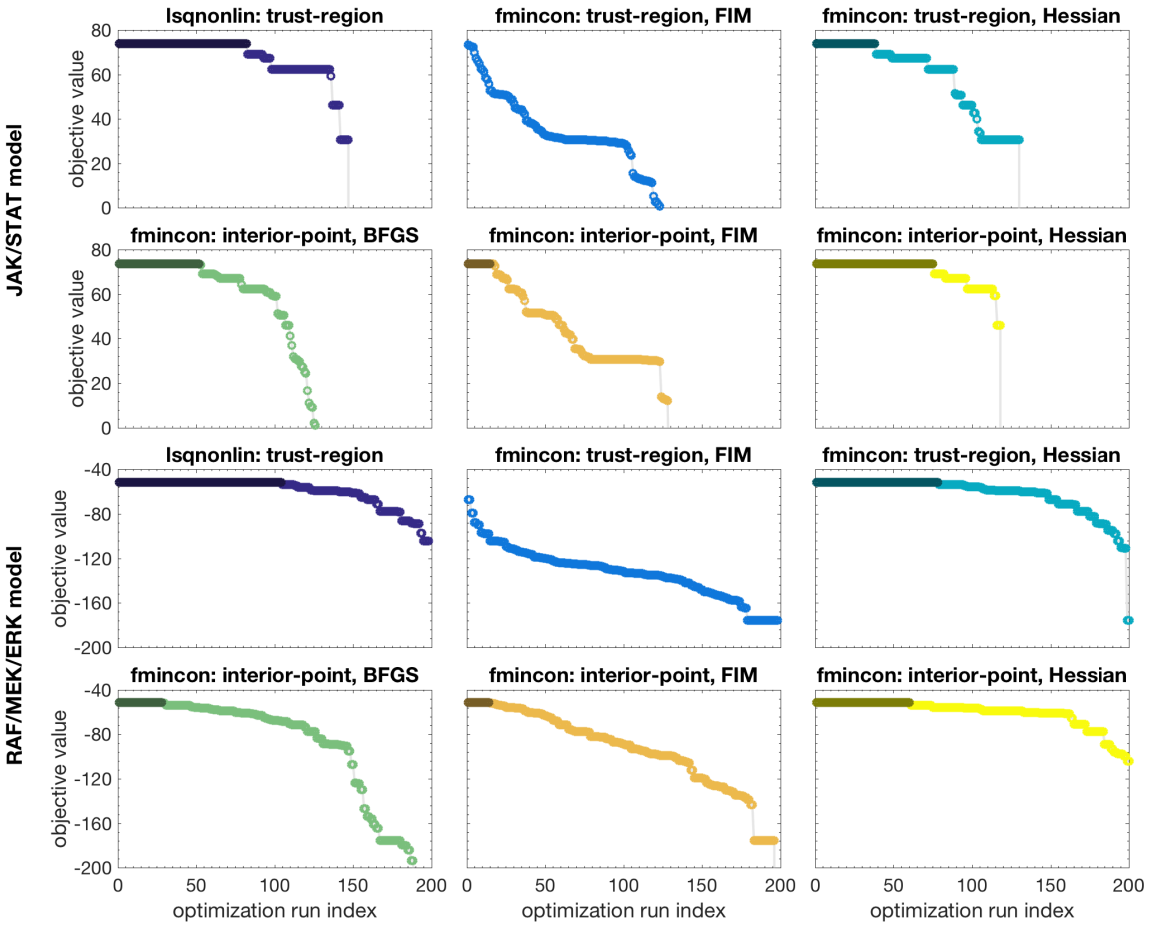

Figure 4.6: Waterfall plots of different local optimization methods. For the models of JAK/STAT and RAF/MEK/ERK signaling, four multi-start local optimizations (of which the first is shown here) with six different local optimization methods were carried out (using the least-squares optimization algorithm lsqnonlin with the trust-region-reflective algorithm, the constraint optimization algorithm fmincon with a trust-region-reflective algorithm provided with either the FIM or the Hessian computed with second order adjoint sensitivity analysis, and fmincon with an interior-point method with either a BFGS approximation of the Hessian, the FIM, or the Hessian computed with second order adjoint sensitivity analysis). Each multi-start consisted of 200 local optimization runs, of which the final objective function values were sorted and depicted. Plateaus in the waterfall plot indicate local optima of the objective function. This figure is taken from the author's publication [Stapor et al., 2018a], Supplementary Figure 1.

thus represent challenging optimization tasks. For each of these local optimization methods, we performed four multi-start local optimizations with different initializations and 200 starting points each. Mean and standard deviation depicted in Figure 4.5 were computed over these four repetitions. For the numerical experiments, we provided the following tolerances to the local optimization algorithms:

- tolerance for step size in parameter: 10^{-10} for the JAK/STAT model, 10^{-12} for the RAF/MEK/ERK model
- tolerance for change in the objective function: 10^{-10} for both models
- tolerance for the remaining gradient: 10^{-6} for both models

Moreover, the following settings have been used for optimization:

- maximum number of optimization steps (per step at least one gradient evaluation): 2000
- maximum number of objective function evaluations: $1000 \cdot n_\theta$
- PrecondBandWidth = inf, i.e. the inner optimization problem was solved by factorization

For all other optimization options, the MATLAB default settings were used.

We considered the least-squares algorithm lsqnonlin as gold standard for the considered problem class, as this method has previously been shown to be very efficient [Raue et al., 2013b]. Here, we studied the effect of using exact Hessians on the optimization algorithms trust-region-reflective and interior-point implemented in fmincon. As performance measure of the optimization methods, we considered the computation time per converged start (i.e. starts which reached the global optimum), the total number of converged starts and the number of optimization steps. The definition of a threshold of the objective function value to which an optimization was accepted as converged to the global optimum was based on the corresponding waterfall plot for each experiment (see Figure 4.6).

The least-square solver lsqnonlin outperformed, as expected, the constrained optimization method fmincon (Figure 4.5 and Figure 4.6). Among the constrained optimization methods, the methods using exact Hessians computed with the second order adjoint method performed equal or better than the alternatives regarding overall computational efficiency (Figure 4.5 A). Indeed, these methods reached a higher percentage of converged starts (Figure 4.5 B and Figure 4.6) than fmincon using the FIM or the BFGS scheme. This is important, as convergence of the local optimizer is often the critical property [Raue et al., 2013b]. In addition, the number of necessary function evaluations was reduced (Figure 4.5 C). Furthermore, we found differences in convergence and computational efficiency for fmincon, depending on the chosen algorithm.

4.2.6 Application 2: Profile computation

To assess the benefits of Hessians in uncertainty analysis, we compared the performance of optimization- and integration-based profile calculation methods for the models M2 and M3. For the optimization-based approach, we employed the algorithm implemented in PESTO, which uses first order proposal with adaptive step-length selection [Boiger et al., 2016]. We compared the local optimization strategies described in the previous section (omitting the methods based on the FIM, due to their poor performance). For the hybrid approach, we used the MATLAB default tolerances for ODE integration. We compared the hybrid scheme using Hessians and the FIM. All profiles were computed to a confidence level of 95%.

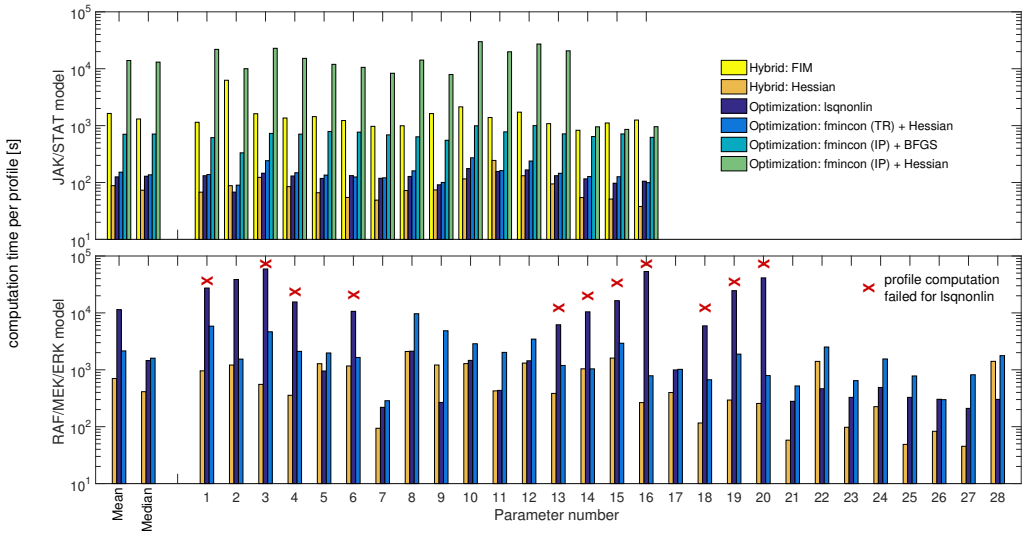

Figure 4.7: Computation time for parameter profiles using either the optimization-based method (lsqnonlin or fmincon with trust-region (TR) or interior-point (IP) algorithm and Hessian or BFGS approximation), or the hybrid method with either FIM or Hessian. Top: Total computation time for all profiles of the application example M3 (JAK/STAT signaling model). Bottom: Total computation time for all profiles of the application example M2 (RAF/MEK/ERK signaling model). This figure is taken from the author's publication [Stapor et al., 2018a], Supplementary Figure 7.

The comparison of the profile likelihoods calculated using different approaches revealed substantial differences (Figure 4.7). The optimization-based approaches worked well for the JAK/STAT model but mostly failed for the RAF/MEK/ERK model (Figure 4.7A). For the RAF/MEK/ERK model, only fmincon with the trust-region-reflective algorithm and exact Hessians worked reliably among the optimization-based methods. Even lsqnonlin yielded inaccurate results for 11 out of 28 parameter profiles (Figure 4.9). A potential reason is that the tolerances – which were previously also used for optimization – were not sufficiently tight. Purely integration-based methods failed due to numerical problems, e.g. jumps in the profile paths (Figure 4.8). Even extensive manual tuning and the use of different established ODE solvers (including ode113, ode45, ode23, and ode15s) did not result in reasonable approximations for all profiles. In contrast, the hybrid approach computed accurate profiles for all parameters and all models, when provided with exact Hessians. However, when provided with the FIM, the hybrid approach failed as soon as it had to perform optimization steps.

In addition to the accuracy, also the computation time of the methods varied substantially. The hybrid method using exact Hessians was substantially faster than the remaining methods (see Figure 4.7). The second fastest method was the optimization-based approach using exact Hessians with the trust-region-reflective algorithm for optimization. lsqnonlin was slightly slower and fmincon using the interior-point algorithm substantially slower (for both, the BFGS scheme and Hessian), although they – as mentioned above – did not provide accurate profiles.

Overall, the proposed hybrid approach using exact Hessians outperformed all other methods. Compared to the best reliable competitor (optimization-based profile calculation using fmincon with the trust-region-reflective algorithm and exact Hessians), the computation time was reduced by more than a factor of two. This is substantial for such highly optimized routines and outlines the potential of exact Hessians for uncertainty analysis.

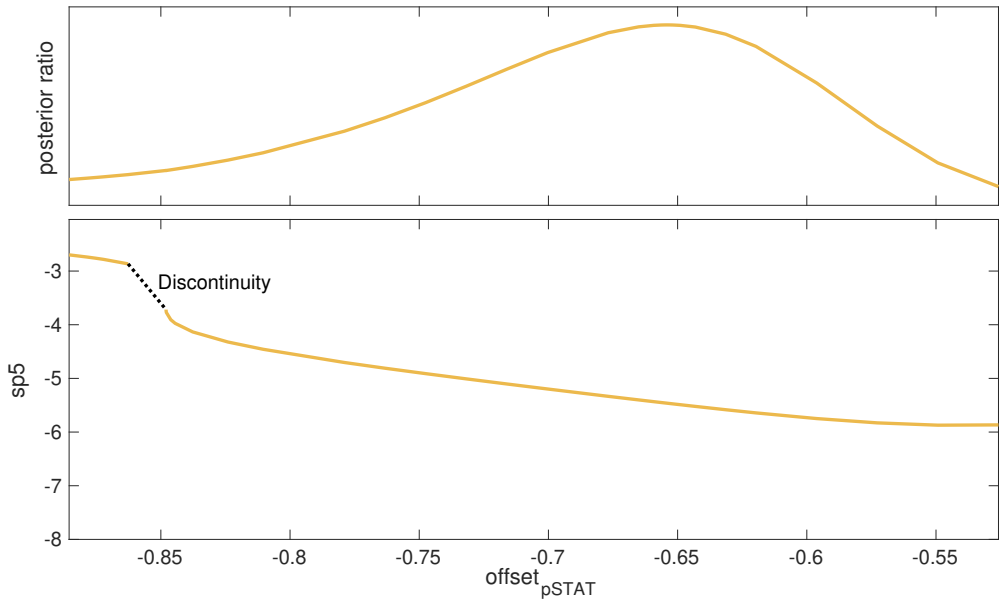

Figure 4.8: Discontinuity in the profile path. The profile of the parameter $\text{offset}_{\text{tSTAT}}$ in the JAK/STAT example is shown in the upper figure. The path of the parameter profile for $\text{offset}_{\text{tSTAT}}$ is projected to the subspace spanned by $\text{offset}_{\text{tSTAT}}$ and another parameter (sp5) and depicted in the lower figure. The parameter values are depicted on the axes. The jump in the profile path of $\text{offset}_{\text{tSTAT}}$, which had to be computed using optimization steps, is shown as dashed line. This figure is taken from the author's publication [Stapor et al., 2018a], Supplementary Figure 5.

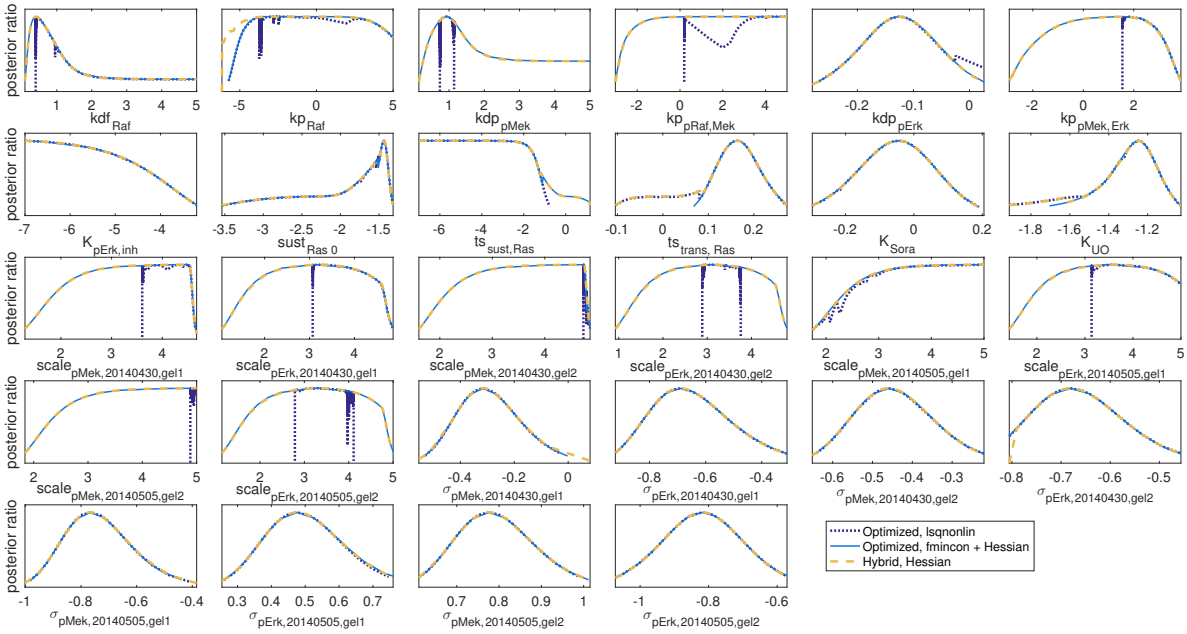

Figure 4.9: Profile likelihoods for the 28 parameters of the RAF/MEK/ERK example. The posterior ratio in the profile likelihood is depicted against the parameter value for each profile, profiles are cut off at a confidence level of 95%. The two methods using the Hessian (light blue and yellow, optimization based and hybrid) are in good agreement with each other. The optimization based profiles computed with lsqnonlin show an unfavorable behavior for 11 out of 28 profiles, as those profiles are not fully computed and cut off due to convergence problems during optimization. This figure is taken from the author's publication [Stapor et al., 2018a], Supplementary Figure 3.

4.3 Discussion

4.3.1 Summary and conclusion

ODE models in systems and computational biology rely on estimated parameter values, which are inferred from experimental data with numerical optimization methods. In this chapter, we presented second order adjoint sensitivity analysis, a method to compute accurate Hessians at low computational cost. This means that the method scales linearly in the number of model parameters and state variables. We also provided a ready-to-use implementation thereof in the freely available toolbox AMICI, and benchmarked its efficiency on a set of published models from systems biology. Then, we demonstrated that the efficiency of some of the most common methods in parameter optimization in systems biology can be improved by using those Hessians. Finally, we introduced a novel method for profile computation, which outperformed existing approaches by efficiently exploiting the Hessian of the objective function.

We showed that second order adjoint sensitivity analysis possesses better scaling properties than common methods to compute Hessians while yielding accurate results, rendering it a promising alternative to existing techniques. Moreover, as the Hessian can be used for optimization, we demonstrated that state-of-the-art constraint optimization algorithms yield more robust results when using exact Hessians. However, computing and exploiting those exact Hessian was not beneficial in all situations: The analysis of the optimizer performance revealed that least-squares algorithms (such as lsqnonlin), which exploit the problem structure to a higher degree, are difficult to outperform and still remain the methods of choice, if applicable. Many parameter estimation problems considered in systems biology do however not possess this structure. This is for instance the case for problems with additional constraints [Mitra et al., 2018], applications considering the chemical master equation [Fröhlich et al., 2016], or ODE constrained mixture models [Hasenauer et al., 2014]. For these problem classes, the trust-region-reflective and interior-point optimization algorithms as implemented in fmincon are the state-of-the-art methods.

For the computation of profile likelihoods, we demonstrated that Hessians from second order adjoint sensitivity analysis reduce computation time and improve robustness for all of the investigated computational methods. This makes sense, as in optimization-based profile computation, local optimizations are started close to a local optimum. In the vicinity of an optimum, the accuracy of second order derivatives is more crucial, as the convergence of an optimization algorithm depends on a good correspondence of the outer and the inner optimization problem. In this situation, providing the optimization algorithm with exact Hessians allows to outperform even highly problem tailored algorithms such as lsqnonlin.

Finally, we presented a hybrid method for profile computation, which can efficiently handle problems that are poorly locally identifiable and have ill-conditioned Hessians. We also provided an implementation of this method in the freely available parameter estimation toolbox PESTO. Although being a reliable approach for uncertainty analysis [Fröhlich et al., 2014], profile likelihoods are often disregarded due to their high computational effort. The presented hybrid method based on exact Hessians may be an approach the tackle this problem, as already the still simple implementation used in this thesis outperformed all established approaches.

4.3.2 Further applications for second order adjoint sensitivities

Beyond the presented applications in this thesis, second order adjoint sensitivity analysis together with the presented implementation can be used for further applications. As some of these

applications are straightforward to implement, we expect them to be investigated soon.

Methods for exploiting Hessians in optimization more efficiently

In recent years, novel optimization algorithms are being steadily developed, which exploit curvature information about the objective function (see Fröhlich et al. [2019] for a review of optimization methods).

Either directions of negative curvature can be used to escape saddle-points efficiently [Dauphin et al., 2014], or third-order approximations of the objective functions are constructed iteratively from Hessians along the trajectory of optimization to improve the convergence order [Cartis et al., 2011, Martinez and Raydan, 2017, Nesterov and Polyak, 2006]. As this information is most efficiently computed by second order adjoint sensitivity analysis, an application would be a natural next step.

Another approach might be a hybrid optimization algorithm which switches the type of the second order derivative: Such an algorithm could start with a quasi-Newton-method in the beginning of the optimization process and switch to exact Hessians as soon as the algorithm comes close to a first-order stationary point. This might improve optimization performance, as the computationally still more demanding Hessian is only computed when it is likely to be beneficial. Such approaches might outperform current optimization strategies, which are often not designed to exploit, e.g., directions of negative curvature that may be present in non-convex problems. These methods will be particularly interesting subjects of further studies when being combined with second order adjoint sensitivity analysis.

Applications for directional second order derivatives

While this chapter focused on the efficient calculation of the Hessian, second order adjoint sensitivity analysis can also be used to compute Hessian vector products [Özyurt and Barton, 2005]. This information can be exploited by optimization methods such as truncated Newton [Nash, 1984] or accelerated conjugate gradient [Andrei, 2009] algorithms, which are suited for large-scale optimization problems. As the corresponding implementation for directional second order adjoint sensitivity analysis is already available within AMICI, applying it for optimization would be straight forward. It was however disregarded for this thesis, as the optimization algorithms implemented in MATLAB are not designed to efficiently exploit Hessian vector products and hence, adequate optimization algorithms would still have to be implemented first.

Further applications

Besides optimization and profile calculation, Hessians can also be used for local approximations of the likelihood function, which allows an approximative assessment of practical identifiability properties and the approximation of the parameter confidence intervals for large-scale models [Fröhlich et al., 2018a, Kapfer et al., 2019]. In addition, Hessians can be employed by for certain MCMC sampling methods [Girolami and Calderhead, 2011, Lan et al., 2015]. Also bootstrapping methods [Joshi et al., 2006] or the radial penalization method [Kreutz, 2018] would profit from accurate Hessians, as the performance of optimization is improved, which is used in these methods. Hence, the approaches presented here may accelerate various uncertainty analysis methods and make their application feasible for model sizes, for which these methods have not been applicable before.

4.3.3 Outlook and further research

Beyond simply exploiting the proposed second order adjoint sensitivity method, also further research in the field of adjoint sensitivity analysis itself with applications in systems biology is a promising next step. Either the proposed methods can be tailored more specifically to certain problem classes, or completely novel problem classes can be tackled by extending adjoint methods.

Adjoint sensitivity analysis for steady-state data

Especially for large-scale ODE models, it is common that measurement data is only available for the steady-state of the system [Fröhlich et al., 2018a, Khodayari and Maranas, 2016, Schmiester et al., 2019a]. Mathematically, this is a particular case, which exhibits a particular problem structure. Taking advantage of this structure could lead to substantially more efficient algorithms: Inferring the steady-state of the system is often possible by direct root finding approaches, such as Newton's method, as implemented in many computational toolboxes [Fröhlich et al., 2019, Mendes et al., 2009]. It has been shown that these approaches can outperform classical ODE integration by orders of magnitude [Lines et al., 2019].

Adjoint sensitivity analysis can be combined with direct steady-state computation, as for the steady-state of the original ODE system also the adjoint problem has a specific structure: It reduces to a linear ODE with constant matrix. Such problems have an analytical solution, which can be computed by solving a linear system of equations, instead of integrating an ODE system. Hence a corresponding implementation would be highly beneficial.

Adjoint sensitivity analysis and discrete events

Another extension of the current version of adjoint sensitivity analysis would be deriving and implementing the adjoint equations for ODE systems which exhibit discrete events, i.e., discontinuities in the right hand side of the ODE or even in the ODE solution. Discrete events are a common way to model certain biology systems which show extremely stiff dynamics with almost instantaneous reactions to specific triggers, such as spiking neurons [Izhikevich, 2003]. Corresponding implementations for time-dependent triggers are implemented in many state-of-the-art toolboxes [Fröhlich et al., 2019, Hoops et al., 2006, Raue et al., 2015, Somogyi et al., 2015]. However, at the moment only AMICI is able to handle more complex (e.g., parameter or state-dependent) trigger functions while also integrating the corresponding forward sensitivity equations, even up to second order [Fröhlich et al., 2017b]. Yet, the counterpart for adjoint sensitivity analysis has so far neither been derived nor implemented. Filling this gap with an appropriate mathematical and computational solution is an important next step, which would for the first time allow efficient parameter estimation for large-scale models with discrete events.

Adjoint sensitivity analysis for higher order derivatives

Recently, a novel method for approximative profile computation has been proposed [Lill et al., 2019]. It exploits the geodesic equations on the model manifold that rely on the Christoffel symbols, which are local representations of covariant derivatives. Already by approximating the geodesic equation by using constant Christoffel symbols, some important non-local effects can be captured.

For some specific problems in parameter estimation, this method can outperform existing approaches [Lill et al., 2019]. As Christoffel symbols go beyond the Hessian matrix and allow

to approximate third order derivatives of the objective function, they can currently only be computed from second order forward sensitivities. As we have seen in this chapter, computing second order forward sensitivities scales poorly in the number of model parameters. However, a corresponding adjoint formulation for the Christoffel symbols can be derived, which enjoys better scaling properties, namely a similar scaling to second order adjoint sensitivities as proposed here. Deriving and implementing this method would allow to test this approach even for medium-scale models, and might hence be a promising alternative to current techniques.

Chapter 5

Parameter estimation for ODE mixed-effect models

In the last decades of the 20th century, measurement techniques such as immunoblotting [Renart et al., 1979] or quantitative polymerase chain reaction (qPCR) [Gibson et al., 1996] allowed to collect large amounts of data about average protein and RNA levels in bulks of cells. From these data, many ODE models of biochemical reaction networks have been constructed, which in turn enhanced our understanding of many cellular processes [Bachmann et al., 2011, Becker et al., 2010, Chen et al., 2009, Kholodenko, 2007, Zheng et al., 2012]. In recent years, new measurement techniques came up, from which similar information about RNA and protein levels could be inferred for single cells [Giesen et al., 2014, Herzenberg et al., 2006, Kolodziejczyk et al., 2015]. This higher level of detail in measurement data pushed again our understanding of cell biology: It showed that genetically identical cells may develop pronounced cell-to-cell variability upon stimulation [Spencer et al., 2009], it allowed to observe how cells differentiate from one cell type to another [Haghverdi et al., 2016], and it enabled to understand how a gradual response of a cell population can be controlled by exploiting cell-to-cell variability [Mitchell and Hoffmann, 2018].

The heterogeneity which is observed in isogenic cells is mostly assumed to arise from two different sources [Llamosi et al., 2016, Swain et al., 2002]: The first one is typically attributed to the stochasticity of intra-cellular processes (e.g., in gene expression) and is sometimes termed intrinsic noise [Swain et al., 2002]. The second one is attributed to differences between cells, such as different protein abundances or variability of compartment and cell sizes across cells, governed by deterministic dynamics, and termed extrinsic noise [Llamosi et al., 2016, Rosenfeld et al., 2005]. As method development for ODE models is the focus of this thesis, the following chapter will be concerned with contributions to single-cell ODE modeling. Due to their deterministic nature, single-cell ODE models are mainly used to model extrinsic noise. In order to underline that extrinsic noise is a fundamental biological property of cells and does not arise from measurement errors, we will rather use the term cell-to-cell variability in the following.

Recently, the additional information in single-cell data led to the development of new mathematical models [Hasenauer et al., 2014, Karlsson et al., 2015, Khammash, 2009, Zechner et al., 2012]. However, due to the higher complexity of these data, the landscape even of deterministic single-cell models is more heterogeneous than this is the case for classic ODE models [Loos and Hasenauer, 2019]. One reason for this can be attributed to the various types of single-cell data themselves, such as time-lapse, time-lapse statistics, and snapshot data: Each of these data types is collected by different measurement techniques, has specific advantages, and comes

with its own challenges. Hence, different modeling approaches and computational methods have been developed [Filippi et al., 2016, Karlsson et al., 2015, Loos et al., 2018b]. For this reason, combining different types of single-cell data with each other or with population average data is currently an open problem [Loos and Hasenauer, 2019]. Moreover, single-cell data allows to infer knowledge about the distribution of measurable quantities across cells. How to best describe and estimate these distributions rather than only average values of measurements in cell populations by statistical models is another challenging question [Pinheiro and Bates, 1996].

In the comparably young field of single-cell modeling, it is common that novel mathematical methods need to be developed for applicational projects. This discerns single-cell ODE modeling from modeling with population average data, for which a number mature and highly optimized computational toolboxes are freely available and maintained since several years [Choi et al., 2018, Fröhlich et al., 2019, Hoops et al., 2006, Raue et al., 2015]. Also the contributions of this chapter were motivated by the development of an ODE mixed-effect model (ODE MEM) of JAK2/STAT5 signaling. Yet, they are applicable to a wide range of modeling problems. As a first contribution, we propose an approach how to integrate two different data types – population average and single-cell snapshot data – into one statistical model. This approach enables the estimation of model parameters and the selection between candidate models for both data types simultaneously in a statistically sound manner, which has so far been an open problem [Loos and Hasenauer, 2019]. Secondly, when working with ODE MEMs in systems biology, the covariance structure of the cell-specific parameters (the random effects) has to be inferred. We introduce a novel, Lie-theoretic approach, to parametrize and estimate covariance matrices and compare it with previously proposed methods for this task [Pinheiro and Bates, 1996].

This chapter is based on my work in the following publication:

- **Stapor, P.***, Adlung, L.*, Tönsing, C.*, Schmiester, L., Schwarzmüller, L., Wang, D., Timmer, J., Klingmüller, U., Hasenauer, J., Schilling, M. (2019). Cell-to-cell variability in JAK2/STAT5 pathway components and cytoplasmic volumes define survival threshold in erythroid progenitor cells. bioRxiv, 10.1101/866871, *Under review*

5.1 Background: ODE MEMs of cellular processes

5.1.1 Modeling and simulating population average and single-cell snapshot data

In this chapter, we focus on ODE based models which capture cell-to-cell variability in highly abundant biochemical species. Despite this restriction, various measurement techniques exist and diverse modeling approaches have been developed (see [Loos and Hasenauer, 2019] for a review). In contrast to population average measurements, single-cell measurements are typically given by a vector or a distribution of observed values rather than by a single value, which describes the average across a bulk of cells. Hence, those experiments need to be simulated differently and have to be described by other likelihood functions than this is done in classic ODE models.

Statistical models of cell populations

The simplest method to model populations of single cells is the standard two stage approach (STS) [Karlsson et al., 2015, Laird and Ware, 1982], sometimes also called the naive approach [Llamosi et al., 2016]. In this concept, the data of each cell is treated as a separate parameter estimation problem. By fitting all cells independently, a distribution of parameter vectors is

obtained. Despite being intuitive and simple to implement, the STS has major drawbacks: Firstly, the STS fails to properly discern measurement noise and biological variability, as it is agnostic about the underlying structure of the single-cell population [Fröhlich et al., 2018b, Karlsson et al., 2015]. This shortcoming can to some degree be addressed by recent improvements [Dharmarajan et al., 2019]. Secondly, it would be computationally extremely demanding when dealing with single-cell snapshot data, which was used in the application example of this chapter. In single-cell snapshot data, thousands or, if multiple experimental conditions are considered, even millions of cells can be measured. When using the STS, the model would have to be fitted to all these cells independently, which would increase the computational complexity by multiple orders of magnitude.

Mixed-effect models (MEMs) are a statistically sound approach to allow variable parameters across a population of cells [Karlsson et al., 2015, Laird and Ware, 1982]. MEMs use a common distribution assumption for the cell population, which constrains the single-cell parameters and allows it to discern biological variability from measurement noise [Fröhlich et al., 2018b]. As mentioned in Equation 2.6, each cell (indexed by ℓ) has a specific parameter vector ϕ^ℓ, which is computed as a combination of the fixed effect parameter vector β and the random effect parameter vector b:

$$\phi^\ell = F\beta + Rb^\ell \tag{2.6}$$

Here, F and R are called the design matrices for the fixed and the random effects [Pinheiro, 1994].

Due to their clear statistical foundations, MEMs are increasingly employed to model cell-to-cell heterogeneity [Almquist et al., 2015, Fröhlich et al., 2018b, Karlsson et al., 2015, Llamosi et al., 2016, Loos et al., 2018b]. Originally, the concept of MEMs was developed to study differences within a group of individuals, e.g., humans or animals [Fisher, 1918, Henderson et al., 1959], but is now adopted in many scientific domains and has become a research topic of its own [Kuhn and Lavielle, 2005, Pinheiro, 1994]. In particular, the field of pharmacokinetics and pharmacodynamics is worth being mentioned [Beal and Sheiner, 1980, Laird and Ware, 1982, Sheiner and Beal, 1983]: Some of the main computational toolboxes originate from this domain, such as MONOLIX [Antony, 2019] and NONMEM [ICON Development Solutions, 2020]. However, both are sold under proprietary licenses.

Likelihood functions for cell populations

In systems and mathematical biology, MEMs are commonly used when dealing with single-cell time-lapse data [Karlsson et al., 2015]. In this case, the likelihood contributions of the cells are multiplied, as intracellular processes are assumed to be independent across cells – which is questionable due to possible batch effects. In MEMs, the common distribution assumption for the model parameters enters the likelihood formulation, as the likelihood of the parameters of each single-cell is is conditioned on this common distribution assumption [Pinheiro, 1994]. The likelihood of the ℓ-th cell given the measurement data $\bar{y}^{\ell,k}$ for the k-th experimental condition is then given as:

$$\mathcal{L}^{\ell,k}(\beta,\delta,\lambda) = \int\limits_{b^\ell \in \mathbb{R}^{n_b}} \mathrm{p}\left(\bar{y}^{\ell,k} \;\middle|\; \beta, b^\ell, \Lambda(\lambda), u^k\right) \mathrm{p}\left(b^\ell \;\middle|\; D(\delta)\right) db^\ell \tag{5.1}$$

Here, λ denotes the parametrization of a covariance matrix of measurement noise Λ of the single-cell trajectories and δ are the parameters which describe the covariance structure of the random

effects Σ. The evaluation of this likelihood requires the integration over b^ℓ, which is numerically problematic, as the computational effort of standard quadrature rules scales exponentially with the dimension of b, which complicates maximum likelihood computation [Kuhn and Lavielle, 2005, Pinheiro, 1994]. Moreover, although this likelihood function would in principle be applicable also to single-cell snapshot data, it would require simulations for each measured cell, similar to the STS. This makes it computationally extremely expensive and hence unsuitable for single-cell snapshot data.

Alternatively and more specific to single-cell snapshot data, distribution assumptions about the measured quantities can be used to derive a likelihood function, under which each single cell is evaluated [Spencer et al., 2009]. This approach has shown to perform well in practice and can also be combined with models accounting for intrinsic noise [Filippi et al., 2016], subpopulation structures within the data [Hasenauer et al., 2014, Loos et al., 2018b], or MEMs to allow certain parameters to vary across cells [Loos et al., 2018b].

The main problem for both approaches is that the likelihood formulations differ substantially from the likelihood which is used for population average data. Hence, it is unclear how data from those two measurement techniques can be combined with each other or with population average data. Conceptually, the likelihood contributions from different experiment types could simply be multiplied like this is done for different experimental conditions in classic ODE models. However, it is likely that the likelihood from one data type will then outweigh the other contributions and hence parameter estimation may yield strongly biased results. This could be cured by introducing an additional weighting term to appropriately balance the dissimilar contributions, but so far there exists no convincing solution for this problem [Loos and Hasenauer, 2019].

Simulating single-cell snapshot data

When simulating cell populations by computational methods, in-silico populations of parameter vectors have to be created. In such an in-silico population, each individual (reflected by a parameter vector) can represent one or multiple cells. The main goal is to balance computational complexity while accurately reflecting the distribution of the measured quantities of the real cell populations. For this task, different approaches have been proposed in the literature.

The simplest and probably most accurate way to simulate a cell population is creating a massive Monte Carlo sample from the assumed distribution of random-effects (e.g., a multi-variate normal distribution), in which each individual reflects a single cell (Figure 5.1). As this leads to in-silico populations with thousands of individuals, this approach becomes computationally prohibitive for larger models, as the computation time grows linearly with the number of individuals. At the other end of the spectrum, sigma point methods such as discussed in [van der Merwe, 2004] yield a very efficient, but approximative approach: A d-dimensional distribution is approximated by $2d + 1$ individuals, which are located at the center and along the coordinate axes. For linear models, this approach matches mean and covariance exactly, but for nonlinear models such as the solutions of ODEs, the approximation quality is highly model dependent and may be poor [Weiße et al., 2010]. Dirac mixture distributions (DMDs) provide an approach in between [Gilitschenski and Hanebeck, 2013, Wang et al., 2019]: A DMD is a small set of points which is optimized to reflect the target distribution (here: a multivariate standard normal distribution) as accurately as possible. A modified Cramér-von Mises distance [Hanebeck and Klumpp, 2008] between the DMD and the target distribution has to be minimized, in order to obtain the Dirac points. As the number of these Dirac points can be freely chosen, which allows to adapt the approximation quality, a DMD can be interpreted as a small, but optimal sample.

When fitting multiple ODE MEMs with different numbers of random effects, e.g., in order to

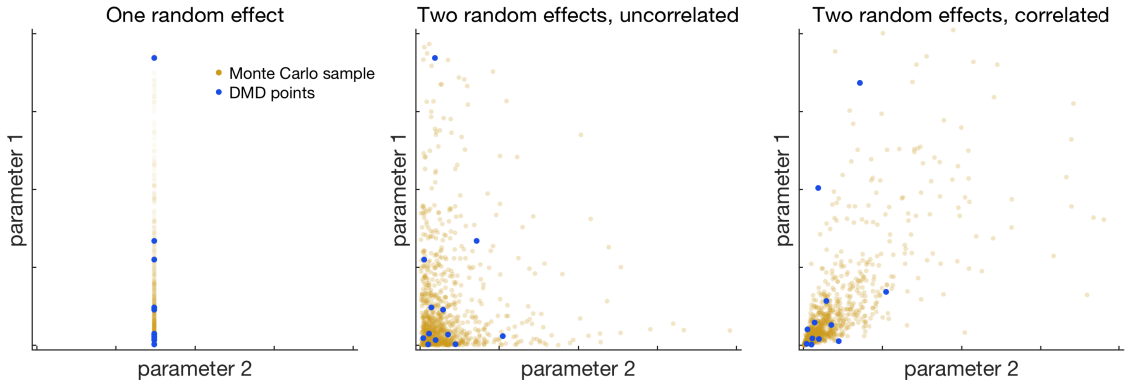

Figure 5.1: In-silico population of parameter vectors for different MEMs with two parameters. Monte Carlo samples with 1000 individuals are depicted in yellow, a Dirac mixture model (DMD) with 12 individuals is depicted in blue **A** MEM with two fixed effects and one log-normally distributed random effect. **B** MEM with two fixed effects and two uncorrelated, log-normally distributed random effects. **C** MEM with two fixed effects and two positively correlated, log-normally distributed random effects.

identify the best among a group of candidate models, the choice of the in-silico population may have a substantial impact on the outcome. As DMDs or sigma-points are approximative methods, the approximation quality can differ for different numbers of random effects or individuals [Wang et al., 2019]. This has to be taken into account when comparing log-likelihood values of the fitted candidate models, as done in model selection [Fröhlich et al., 2019].

5.1.2 Estimating covariance matrices in MEMs

In an ODE MEM, the parameters which describe the covariance structure Σ of the random effects, denoted by δ, have to be inferred and hence extend the estimation problem when compared to a classic ODE model. In many applications, Σ is often assumed to be diagonal for simplicity and therefore, possible correlations between the random effects are disregarded [Fröhlich et al., 2018b]. However, these correlations may be important, as they encode information about the modeled biological system, such as coregulations of certain proteins or coupled synthesis and degradation rates [Llamosi et al., 2016]. Hence, it is important to not restrict the model to diagonal covariance matrices, but to also take block-diagonal or other covariance structures into account. For this task, the following approaches have been presented in the literature [Pinheiro and Bates, 1996]:

- The **matrix logarithm** sets $\Sigma = \exp(\Psi)$ [Pinheiro and Bates, 1996]: A symmetric matrix Ψ is parametrized entry-wise and then the matrix exponential is applied to ensure positive definiteness through the spectral mapping theorem.

- The (upper) **Cholesky decomposition** L of Σ [Lindstrom and Bates, 1990] can be parametrized entry-wise. Then, the covariance matrix is obtained by $\Sigma = L^T L$, which again ensures positive definiteness – or semi-definiteness, depending on the choice of the parameter bounds.

- The **spherical parametrization** [Bates and Watts, 1988] is based on the Cholesky decomposition L: It parametrizes the vectors of the upper Cholesky decomposition by spherical coordinates (denoted by l), therefore allowing to put box-constraints on the variances of Σ. It sets $\Sigma = \varphi(l)^T \varphi(l)$ with $L = \varphi(l)$ and φ denoting the coordinate transformation from spherical to Cartesian coordinates.

- The **Givens parametrization** [Thisted, 1988] parametrizes the eigenvalue decomposition by using a diagonal matrix D of eigenvalues and an orthogonal matrix of eigenvectors U, which yields $\Sigma = U^T DU$. The parametrization of U is solved by a product of rotation matrices U_m, yielding $U = U_1 \cdot \ldots \cdot U_M$.

While the parametrizations via the matrix logarithm and the Cholesky decomposition are conceptionally simple, they do not allow to constrain the variances or the eigenvalues of Σ directly by box-constraints [Pinheiro and Bates, 1996]. This complicates the definition of meaningful lower and upper bounds for the corresponding parameters, which are important during parameter estimation or when sampling initial points for multi-start local optimization. Moreover, if random effects are used for parameters which describe initial values of the ODE, those two parametrizations are prone to integration failure of the ODE solver, as large eigenvalues of Σ may lead to unfavorable initial values of the ODE at the initial point of a local optimization. This is a substantial drawback, as frequent ODE integration failure may lead to poor results when performing parameter estimation.

The spherical parametrization allows to parametrize the variance terms directly and defines additional parameters for angles on a sphere, which are related to the correlations between random effects. This allows to constrain the eigenvalues of Σ to a certain degree, but if Σ is sufficiently high dimensional, the eigenvalues can still become large. Finally, the Givens parametrization allows to directly constrain the eigenvalues of Σ and therefore reduces the probability of ODE integration failure at the initial point of a local optimization, which makes it favorable for ODE MEMs. However, the parametrization of the orthogonal matrices is rather involved. This may be an explanation for the comparably slow convergence which was observed for this method [Pinheiro and Bates, 1996].

5.2 Contribution 1: Combining single-cell snapshot and population average data in ODE MEMs

ODE modeling of biological systems typically happens based on population average data first, as these data are cheaper and quicker to collect. For many cellular processes, such ODE models have already been established, curated and made publicly available on databases such as BioModels [Li et al., 2010] or JWS online [Olivier and Snoep, 2004]. Hence, when a system is modeled at the single-cell level, an ODE model and population average data are often already available. It would hence be eligible to also integrate these data and extend the model, instead of completely replacing it. Moreover, single-cell and population average measurements often cover different biochemical species, as not all measurement techniques are suitable for all species. Thus, it is usually impossible to replace all population average data by single-cell data, without losing information about the system. For these reasons, a framework which is able to combine multiple data types is highly desirable.

5.2.1 A common likelihood function for single-cell snapshot and population average data based on statistical moments

Assuming a biochemical species has been measured by single-cell snapshot data such as flow cytometry, the mean value of the measurement can be used just like population average data. However, single-cell snapshot measurements yield distributions of measurement values for each experiment and hence more information can be extracted than only the mean value: As the mean is the first moment of an observable which is distributed across a population of cells, the most

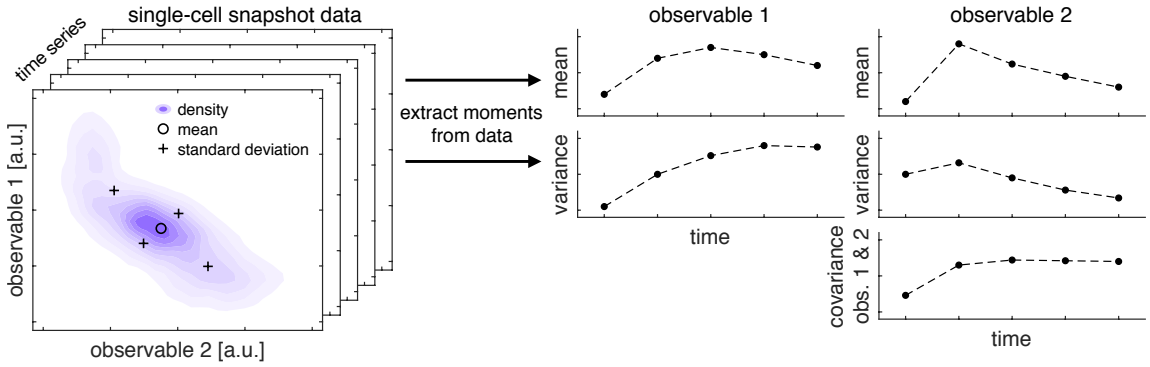

Figure 5.2: Visualization of moment extraction based on single-cell snapshot data. For a time-series of single-cell snapshot measurements, first and second order moments are computed based on Equations 5.3 and 5.4. A computational model can then be fitted to the time-series data of these extracted moments.

straight forward possibility to combine information about the measured distribution would be also considering further (central) moments of higher order. This would be firstly the variance of the measured observable and, in the case of multiplexed (simultaneous) measurements of different observables, the covariances of these observables.

Extracting statistical moments from single-cell snapshot data allows the combination with population average data

In the following, we denote data from population average measurements by $\bar{y}_{ij}^{k}$ and data from single-cell snapshot measurements by $\tilde{z}_{ij}^{k,\ell}$. As before, the index i enumerates the observables, j the measurement timepoints, k the experimental conditions, and ℓ over the cells. This yields the following two datasets:

$$D^{PA} = \left\{ \bar{y}_{i,j}^{k} \right\}_{\substack{i=1,\dots,n_y^k \\ j=1,\dots,n_t}}^{k=1,\dots,n_e} \quad \text{and} \quad D^{SC} = \left\{ \tilde{z}_{i,j}^{k,\ell} \right\}_{\substack{i=1,\dots,n_y^k \\ j=1,\dots,n_t}}^{\substack{k=1,\dots,n_e \\ \ell=1,\dots,n_c(i,j,k)}} \tag{5.2}$$

We propose to use the following quantities as observed quantities for single-cell snapshot data, which provide information about the distribution of the population:

$$\text{1}^{\text{st}} \text{ order moments:} \quad \bar{z}_{\mu(i,j)}^{k} = \mathbb{E}_\ell\left(\tilde{z}_{i,j}^{k,\ell}\right) = \frac{1}{n_c} \sum_{\ell=1}^{n_c(i,j,k)} \tilde{z}_{i,j}^{k,\ell} \tag{5.3}$$

$$\begin{aligned} \text{Central 2}^{\text{nd}} \text{ order moments:} \quad \bar{z}_{\Sigma(i_1,i_2,j)}^{k} &= \mathbb{E}_\ell\left(\left(\tilde{z}_{i_1,j}^{k,\ell} - \mathbb{E}_\ell\left(\tilde{z}_{i_1,j}^{k,\ell}\right)\right)\left(\tilde{z}_{i_2,j}^{k,\ell} - \mathbb{E}_\ell\left(\tilde{z}_{i_2,j}^{k,\ell}\right)\right)\right) \\ &= \frac{1}{n_c - 1} \sum_{\ell=1}^{n_c(i,j,k)} \left(\tilde{z}_{i_1,j}^{k,\ell} - \bar{z}_{\mu(i_1,j)}^{k,\ell}\right)\left(\tilde{z}_{i_2,j}^{k,\ell} - \bar{z}_{\mu(i_2,j)}^{k,\ell}\right) \end{aligned} \tag{5.4}$$

Hence, we use the sample mean and the sample variance over the single cells, which we denote by $\mathbb{E}_\ell$, rather than all data points in the sample. However, the proposed moment extraction technique reduces the information, which can be used for parameter estimation: Not the whole distribution of measured values is exploited, but only its statistical moments (Figure 5.2). Yet,

this approach has the advantage that information about the shape of the distribution (second order moments) can be treated in the same way as information about the average of the measured quantities (first order moments). Hence, single-cell snapshot data can be integrated with data from population average measurements, by interpreting the latter as first oder moments of an unknown distribution of the measured quantity.

Analogously to the simulations $y_i(t_j, \theta, u^k)$, which are linked to population average data points $\bar{y}^k_{i,j}$, the measured quantities $\bar{z}^k_{\mu(i,j)}$ and $\bar{z}^k_{\Sigma(i_1,i_2,j)}$ have to be simulated. This can be done by extending the ODE model to an ODE MEM and by creating an in-silico population (via a Monte Carlo sample, sigma points, or a DMD), which is propagated through the ODE and which yields a distribution of simulated individuals:

$$1^{\text{st}} \text{ order moments:} \quad z_{\mu(i)}(t_j, (\theta, \delta), u^k) = \mathbb{E}_\phi\Big[z_i(t_j, \phi, u^k)\Big] \tag{5.5}$$

$$\begin{aligned} \text{Central } 2^{\text{nd}} \text{ order moments:} \quad z_{\Sigma(i_1,i_2)}(t_j, (\theta, \delta), u^k) &= \mathbb{E}_\phi\Bigg[\Big(z_{i_1}(t_j, \phi, u^k) - \mathbb{E}_\phi\Big[z_{i_i}(t_j, \phi, u^k)\Big]\Big) \\ &\quad \cdot \Big(z_{i_2}(t_j, \phi, u^k) - \mathbb{E}_\phi\Big[z_{i_2}(t_j, \phi, u^k)\Big]\Big)\Bigg] \end{aligned} \tag{5.6}$$

In this notation, ϕ are the parameters of the simulated individuals, and $\mathbb{E}_\ell$ denotes again the sample mean and the sample variance taken over the distribution of simulated individuals. If the single-cell measurements are assumed to be subject to a substantial level of (single-cell) measurement noise, the simulated moments must be corrected. However, as the single-cell measurement noise is typically assumed to have mean 0, only the second order moments are affected. The noise term can be modeled using a multi-variate normal distribution with covariance matrix $\Lambda(\lambda)$, where λ is a parametrization of the noise which extends the parameter estimation problem. Assuming the single-cell measurement noise to be independent of the cell-to-cell variability, the noise-corrected central second order moments read:

$$\begin{aligned} z_{\Sigma(i_1,i_2)}(t_j, (\theta, \delta, \lambda), u^k) = \mathbb{E}_\phi\Bigg[&\Big(z_{i_1}(t_j, \phi, u^k) - \mathbb{E}_\phi\Big[z_{i_i}(t_j, \phi, u^k)\Big]\Big) \\ &\cdot \Big(z_{i_2}(t_j, \phi, u^k) - \mathbb{E}_\phi\Big[z_{i_2}(t_j, \phi, u^k)\Big]\Big)\Bigg] + \Lambda(\lambda) \end{aligned} \tag{5.7}$$

In order to use a consistent statistical model, the simulations of population average data points also have to be simulated by the MEM as well and are treated as measurements of first order moments:

$$y_i(t_j, (\theta, \delta), u^k) = \mathbb{E}_\phi\Big[y_i(t_j, \phi, u^k)\Big] \tag{5.8}$$

A common likelihood function enables the simultaneous fitting with heterogeneous data types

In order to use both data types for parameter estimation, a common likelihood function has to be set up. For this purpose, a noise model for the measurement noise of $\bar{z}^k_{\mu(i,j)}$ and $\bar{z}^k_{\Sigma(i_1,i_2,j)}$ is necessary. In principle, various noise models, such as discussed in, e.g., [Maier et al., 2017], provide reasonable choices. However, if the measurement error is not assumed to scale with the measured observable, an additive normal noise is the simplest choice, to which we want to restrict in the following. We hence model the statistical moments of the single-cell measurements

as random variables in the following sense:

$$\bar{z}^k_{\mu(i,j)} = z_{\mu(i)}(t_j, (\theta, \delta), u^k) + \varepsilon^k_{\mu(i,j)} \quad \text{with} \quad \varepsilon^k_{\mu(i,j)} \sim \mathcal{N}\left(0, \left(\sigma^k_{\mu(i,j)}(\theta)\right)^2\right) \tag{5.9}$$

and

$$\bar{z}^k_{\Sigma(i_1,i_2,j)} = z_{\Sigma(i_1,i_2)}(t_j, (\theta, \delta), u^k) + \varepsilon^k_{\Sigma(i_1,i_2,j)} \quad \text{with} \quad \varepsilon^k_{\Sigma(i_1,i_2,j)} \sim \mathcal{N}\left(0, \left(\sigma^k_{\Sigma(i_1,i_2,j)}(\theta)\right)^2\right) \tag{5.10}$$

Assuming independence of the measurement noise terms for the population average measurements and for the statistical moments of the single-cell measurements, we obtain:

$$\mathcal{L}\big(\theta, \delta \mid D\big) = \mathcal{L}_{PA}\big(\theta, \delta \mid D^{PA}\big) \cdot \mathcal{L}_{SC}\big(\theta, \delta \mid D^{SC}\big) \tag{5.11}$$

The population average likelihood function is given as usual by:

$$\mathcal{L}_{PA}\big(\theta, \delta \mid D^{PA}\big) = \prod_{k=1}^{n_e} \prod_{i=1}^{n_y^k} \prod_{j=1}^{n_t} \frac{1}{\sqrt{2\pi}\sigma^k_{ij}(\theta)} \exp\left\{-\frac{1}{2}\left(\frac{\bar{y}^k_{ij} - y_i(t_k, \theta, u^k)}{\sigma^k_{ij}(\theta)}\right)^2\right\} \tag{5.12}$$

This likelihood function is identical to the likelihood function (2.11) used in Chapter 2. The only difference consists in using the sample means over series of single-cell measurements instead of using population average measurements as data points. In full analogy, relying on the sample variance across single-cell measurements, the single-cell likelihood is given as:

$$\begin{aligned}\mathcal{L}_{SC}\big(\theta, \delta \mid D^{SC}\big) = &\prod_{k=1}^{n_e} \prod_{i=1}^{n_y^k} \prod_{j=1}^{n_t} \frac{1}{\sqrt{2\pi}\sigma^k_{\mu(i,j)}(\theta)} \exp\left\{-\frac{1}{2}\left(\frac{\bar{z}^k_{\mu(i,j)} - z_{\mu(i)}(t_j, (\theta, \delta), u^k)}{\sigma^k_{\mu(i,j)}(\theta)}\right)^2\right\} \cdot \\ &\prod_{k=1}^{n_e} \prod_{\substack{i_1, i_2 = 1 \\ i_1 \le i_2}}^{n_y^k} \prod_{j=1}^{n_t} \frac{1}{\sqrt{2\pi}^k_{\Sigma(i_1,i_2,j)}(\theta)} \exp\left\{-\frac{1}{2}\left(\frac{\bar{z}^k_{\Sigma(i_1,i_2,j)} - z_{\Sigma(i_1,i_2)}(t_j, (\theta, \delta), u^k)}{\sigma^k_{\Sigma(i_1,i_2,j)}(\theta)}\right)^2\right\}\end{aligned} \tag{5.13}$$

Importantly, this formulation allows to avoid the numerically highly problematic marginalization of random effect parameters, by using a large sample of data points and simulations instead. Based on these expressions, the corresponding negative log-likelihood functions can be derived and used as objective functions for parameter estimation. This enables maximum likelihood or maximum a posteriori estimation based on population average and single-cell snapshot data simultaneously.

Implementation in the mixed-effect modeling toolbox MEMOIR

The proposed methodology was implemented in the freely available MATLAB toolbox MEMOIR[1], which provides a framework for (ODE based) mixed effect modeling. MEMOIR already had an implementation for likelihood or posterior computation based on population average data. As a contribution of this thesis, this likelihood function was extended by the proposed approach for single-cell snapshot data based on statistical moments.

[1]Current version available at https://github.com/ICB-DCM/MEMOIR

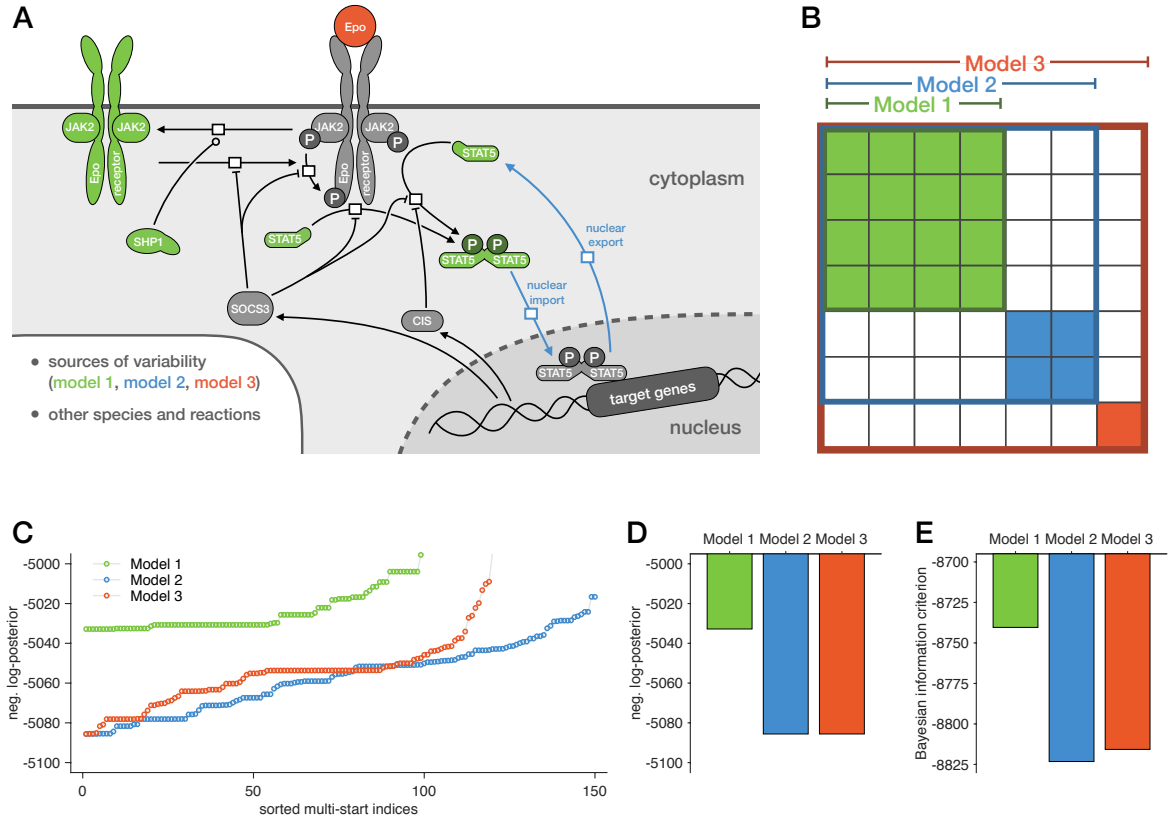

Figure 5.3: Model selection for ODE MEM of JAK2/STAT5 signaling. This figure was adapted from the author's publication [Adlung et al., 2019], Figure 4. **A** Visualization of the ODE model with its species and reactions. Colored components are assumed to be variable across cells. **B** Covariance structures of the different nested ODE MEMs. **C** Waterfall plot with final objective function values for the best 150 out of 400 local optimization runs for the three considered ODE MEMs. **D** Best found final objective function values for each of the three considered ODE MEMs. **E** Model selection via Bayesian information criterion (BIC) reveals that Model 2 provides the best trade-off between goodness of fit and model complexity.

5.2.2 Application: Single-cell ODE modeling of JAK2/STAT5 signaling

The motivating application example for this work was the development of an ODE MEM of JAK2/STAT5 signaling. The JAK2/STAT5 pathway regulates apoptosis decisions in colony forming unit-erythroids (CFU-Es), which are progenitors of red blood cells in the bone marrow [Swameye et al., 2003]. In this application, an ODE model, which was developed by a group of collaboration partners based on an existing model [Bachmann et al., 2011], was used as a starting point. This model describes the phosphorylation of the transcription factor STAT5 through the Janus Kinase JAK2 and the phosphorylated Epo receptor upon stimulation of the Epo receptor with Erythropoietin (Epo) and the migration of phosphorylated STAT5 from the cytoplasm to the nucleus (Figure 5.3 A). Phosphorylated STAT5 activates the transcription of the genes CISH and SOCS3, resulting in the synthesis of the proteins CIS and SOCS3, which regulate STAT5 activation by negative feedback.

The model was extended by the author of this thesis to an ODE MEM by including single-cell snapshot data, which was obtained from flow cytometry measurements carried out by the collaboration partners. Three time-series of flow cytometry data were measured under eight different treatment conditions for eleven timepoints. The measurements were multiplexed and quantified the total amount of STAT5 and the amount of phosphorylated STAT5 simultaneously. In order

to identify the main sources of cell-to-cell heterogeneity in the response upon stimulation with Epo, three candidate models with a nested structure of random effects were developed (Figure 5.3 B), which assume different quantities to be variable across cells: The first candidate model only considered variability in the initial amounts of proteins (the Epo receptor JAK2 complex, the phosphatase SHP1, the amount of total STAT5 and a cell-to-cell variable measurement offset of phosphorylated STAT5) and allowed those random effects to be correlated. The second model additionally allowed the nuclear shuttling rate constants (import and export) to vary across cells and assumes an additional correlation between these two random effects, but assumed those to be independent of the initial protein concentrations. The reason for this heterogeneity can be explained by varying volumes of the nucleus and the cytoplasm, which may lead to different import and export times of the transcription factor. The third model extended this by also including variability in the amount of Epo which is sensed by the cells, as the ligand might be unevenly distributed.

As the sigma point approximation resulted in inaccurate estimates of the covariance structure, a DMD was used to balance computation time and approximation accuracy more flexibly. For the largest candidate model, Model 3, which had seven random effects, the corresponding DMD had $6 \cdot 7 = 42$ Dirac points, consisting of three times as many individuals as a corresponding sigma point approximation, to ensure sufficient accuracy. The DMDs for the smaller models 1 and 2 were computed from the DMD of Model 3 by marginalizing the corresponding dimensions in the DMD. This ensured the behavior of the smaller DMDs to be consistent with the largest DMD, assuming the corresponding random effects were estimated to have no variability. Hence, not only the random effect structure but also the DMDs were nested and had the same number of Dirac points for all models.

Then, the three candidate models were fitted to the data by multi-start local optimization, using the MATLAB routine fmincon with the trust-region-reflective algorithm (Figure 5.3 C) and 400 initial points, which were randomly generated by latin hypercube sampling [Raue et al., 2013b]. The waterfall plot revealed plateaus indicating local optima that could be reproduced by different initial points. Interestingly, the two models 2 and 3, which accounted for cell-to-cell variable shuttling rates, performed clearly better than Model 1, which only allowed the initial protein concentrations to be variable (Figure 5.3 D). Moreover, the models 2 and 3 reached the same negative log-posterior values indicating that the additional variability in the amount of Epo, which is sensed by the cell, was not supported by model and data. For this reason, model selection based on the Bayesian information criterion (BIC) identified the second model as best candidate (Figure 5.3 E).

The nested DMDs turned out to be indeed crucial for model selection: For comparison, Model 2 was also fitted with a DMD consisting of $6 \cdot 6 = 36$ Dirac points, keeping the number of Dirac points per random effect fixed, rather than the DMD itself. Surprisingly, this 36-point-DMD allowed to achieve better negative log-posterior values than it was possible for Model 3 and for Model 2 with the 42-point-DMD. However, using these "better" parameter estimates from Model 2 with the 36-point-DMD with Model 2 and 3 with the 42-point-DMD could not reproduce these superior log-posterior values, but resulted in worse values. As moreover Model 3 is a superset of Model 2, Model 3 should be able to achieve at least the same log-posterior values as Model 2. From this, it can be concluded that in a model selection of nested ODE MEMs which are fitted using DMDs, also the DMDs must be nested to ensure proper model selection.

For the chosen Model 2, we compared the quality of the model simulation after parameter estimation with the measured single-cell data. As the model was only fitted based on statistical moments, which it could explain well (Figure 5.4), it was unclear whether it would also explain

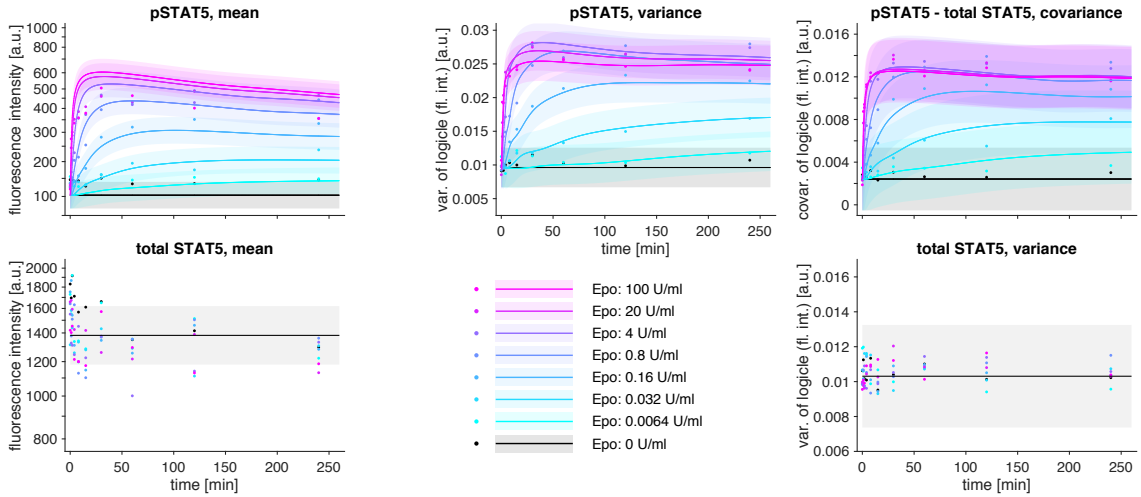

Figure 5.4: Moments extracted from a time series of multiplexed flow cytometry data for the JAK2/STAT5 model with fits of the selected ODE MEM. The model fits (solid lines) reproduce the behavior of the data (dots) for the mean, variance and covariance of phosphorylated and total STAT5, the transparent bands indicate the estimated standard deviation of the data. This figure was adapted from the author's publication [Adlung et al., 2019], Supplementary Figure 5.

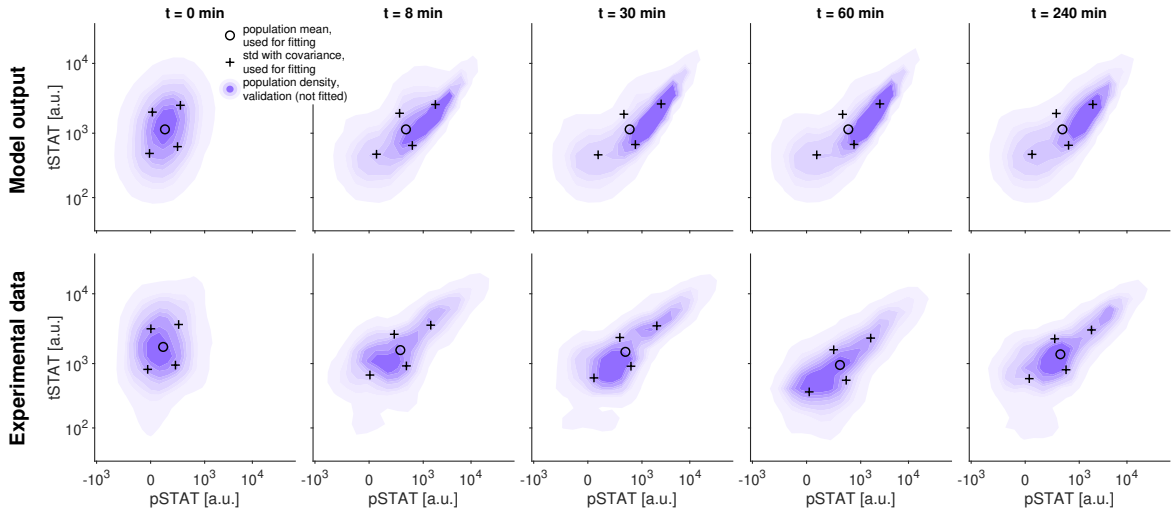

Figure 5.5: Kernel density plots of selected timepoints for multiplexed flow cytometry data and model simulation for the JAK2/STAT5 model. The upper panel shows the model simulation (shading), with mean (circle) and covariance structure (crosses), which were used for parameter estimation. The lower panel shows the corresponding quantities for the measured data. Beyond only explaining the statistical moments, the model also captures the overall shape of the distribution. This figure was adapted from the author's publication [Adlung et al., 2019], Figure 5 A.

the population distribution of the multiplexed measurements. We found that, despite not reproducing the measured distribution with high accuracy, the model indeed captured the overall shape of the distribution and its density to at least a satisfactory degree (Figure 5.5). Given the fact that the model had not been informed about the population density of the measured quantities during parameter estimation, the result was convincing. From this, we can conclude that fitting single-cell snapshot data based on statistical moments does indeed provide a meaningful way of integrating heterogeneous data types into a common framework.

The selected Model 2 was then used, together with data on cellular survival, to infer the mechanism which controls the survival of the CFU-E cells. A set of different candidate survival models was created based on hypotheses which were previously discussed in the literature [Bachmann et al., 2011]. The analysis suggested that the percentage of phosphorylated nuclear STAT5 has to exceed a certain threshold over a time span of about two hours in a single cell, in order to ensure a sufficient survival signal. Otherwise, the cell will undergo apoptosis [Adlung et al., 2019].

5.3 Contribution 2: A novel parametrization of covariance matrices

We have seen that covariance matrices have to be inferred when performing parameter estimation for an ODE MEM. Desirable properties of such parametrizations – especially in the context of an application to ODE MEMs – would be:

Favorable convergence properties during parameter estimation: Especially for ODE MEMs, fast convergence and the possibility to constrain the parameters to a reasonable region in parameter space is most important. Since the covariance matrix is estimated simultaneously with the remaining parameters of the ODE MEM and since parameter estimation for standard ODE models is already challenging [Kapfer et al., 2019, Raue et al., 2013b], good convergence properties of the covariance parametrization are of particular importance.

Meaningful parameter bounds for the parameter estimation problem: The possibility to constrain the model parameters to a plausible region in parameter space prior to optimization is crucial, as well. Especially for ODE MEMs, random effects may influence the stiffness of the underlying ODE. This is particularly important if the initial values of the ODE are modeled by random effects, as poorly chosen starting points of local optimizations may lead to frequent failure of ODE integration and can hence obstruct the optimization process.

Inferring structured covariance matrices without constraints: Ideally, also structured covariance matrices (e.g., block diagonal or more complex shapes) can be estimated, without using constraint optimization methods, as unconstrained optimization methods tend to enjoy better convergence properties than constrained optimization strategies. As an alternative, priors can be used for certain entries of the covariance matrix, in order to constrain them to small values, without using actual constraint-based methods. However, if the parametrization allows to encode a certain structure directly, this will be preferable.

Uniqueness of the parametrization for a given covariance matrix: Ideally, it would be favorable if there was a one-to-one correspondence between parameters and covariance matrices,

as this ensures structural identifiability. However, when working with biological models, in none of the available parametrization techniques, the parameters themselves carry any direct biological meaning, but only the variances and correlations have a biological interpretation. Hence, possible redundancies in the parametrization are likely to be less critical.

5.3.1 A Lie algebraic parametrization of covariance matrices

Among the four methods, which were mentioned in the background section – the matrix logarithm, the Cholesky decomposition, the spherical Cholesky approach, and the Givens parametrization – the Givens parametrization seems to most promising when it comes to constraining the spectrum of the covariance matrix. However, the parametrization of the rotation matrices as a product of elementary rotations is mathematically unsatisfactory, as it depends on the choice of an ordering of these elementary rotations. This may cause dependencies between the rotation angles. For this reason, we propose a novel parametrization approach, which relies on Lie theoretic ideas, in order to improve the Givens parametrization, and discuss it in the context of ODE MEMs.

Concept and properties of the Lie algebraic parametrization

The problematic part is how to best parametrize the rotation matrices. The n-dimensional rotation matrices are a representation of the group of special orthogonal transformations SO_n, which is a compact Lie group. As compact Lie groups have nontrivial geometries, parametrizing them directly is intricate. However, Lie groups enjoy a rich mathematical structure, which can be exploited. A possibility to parametrize the Lie group SO_n indirectly consists in going via the Lie algebra and using the exponential map. The Lie algebra is the tangent space of the Lie group at the unit element and can hence be parametrized easily by choosing a basis, as it is a vector space. For SO_n, the Lie algebra is denoted by $\mathfrak{so}_n$ [Duistermaat and Kolk, 2000], and its elements are often called the infinitesimal generators of SO_n. The exponential map provides then a correspondence between the Lie group and its Lie algebra. It maps the unit element of the vector space 0 on the unit element of the Lie group and is diffeomorphic in a neighborhood around these unit elements. When representing SO_n by the rotation matrices, the corresponding representation of $\mathfrak{so}_n$ are the antisymmetric matrices, and the representation of the exponential map is the matrix exponential.

Similar to the Givens parametrization, the proposed Lie algebraic parametrization δ describes a covariance matrix Σ by its eigenvalue decomposition $\lambda_1, \ldots, \lambda_n$ and its system of eigenvectors, which yield a rotation matrix (Figure 5.6 A). However, this rotation matrix is parametrized in the Lie algebra, i.e., via an antisymmetric matrix, which is then exponentiated to generate the actual rotation matrix (Figure 5.6 B). The parametrization of the antisymmetric matrix happens via coefficients $\alpha_1, \ldots, \alpha_{n_A}$, which are multiplied with a basis $A_1, \ldots, A_{n_A}$ of the antisymmetric matrices (with $n_A = n(n-1)/2$):

$$\delta_n \colon (\lambda_1, \ldots, \lambda_n, \alpha_1, \ldots, \alpha_{n_A}) \longmapsto \exp\left(\sum_{m=1}^{n_A} \alpha_m A_m\right) \exp\left(\operatorname{diag}\left(\lambda_1, \ldots, \lambda_n\right)\right) \exp\left(\sum_{m=1}^{n_A} -\alpha_m A_m\right) \tag{5.14}$$

Hence, the proposed approach describes a covariance matrix by a vector of eigenvalues, which are parametrized logarithmically, to ensure positivity, and a set of coefficients for the antisymmetric matrices (Figure 5.6 C).

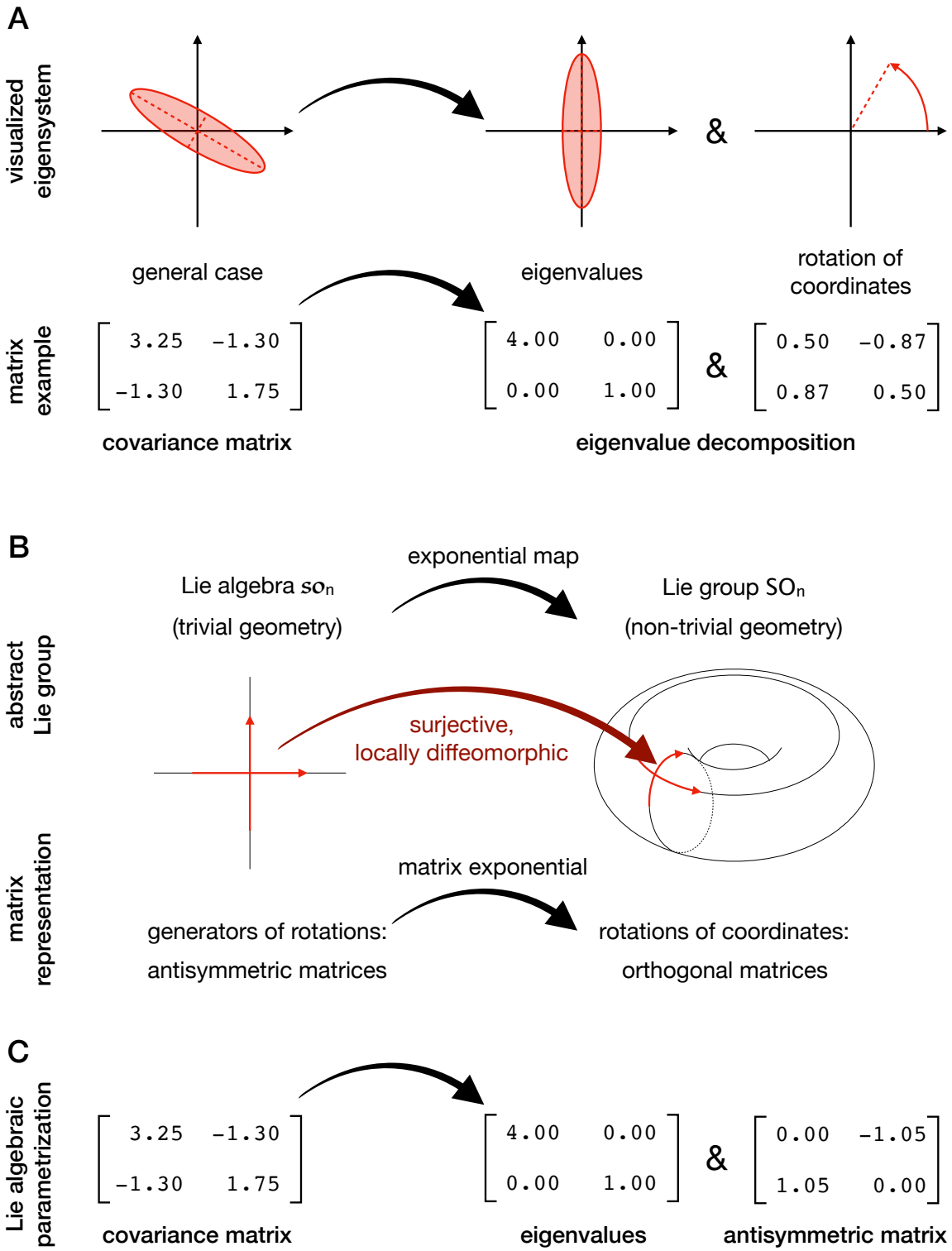

Figure 5.6: Concept of the Lie algebraic parametrization of covariance matrices. **A** A covariance matrix is decomposed into a set of eigenvalues and a special orthogonal (rotation) matrix of eigenvectors. **B** Special orthogonal matrices form the Lie group SO_n, for which the exponential map from the Lie algebra $\mathfrak{so}_n$ (the antisymmetrc matrices) to the Lie group is surjective. Hence, every rotation matrix can be described as the exponential of an antisymmetric matrix. **C** The correspondence between Lie algebra and Lie group can be used to parametrize the rotations in its Lie algebra. Hence, every covariance matrix can be described by a vector of eigenvalues and an antisymmetric matrix.

Practical considerations

In principle, every basis of the antisymmetric matrices will yield a parametrization of the covariance matrices. As only the final generating matrix, i.e., the linear combination of the basis matrices, is exponentiated, the impact of choosing another basis of the antisymmetric matrices is minimal: When changing to another basis, the new parameters δ are obtained by linear transformation between those two bases. As long as two orthonormal bases are chosen, the corresponding transformation matrix will be orthogonal and will thus have a condition number of 1. Hence, for orthonormal bases, the numerical properties of the parametrization will not depend on the choice of the basis.

For the implementation, we chose elementary antisymmetric matrices A_m: Each m translates into a row index m_1 and a column index m_2 by lexicographical ordering, i.e., $(m_1, m_2)(m)$, with $m_2 > m_1$. Hence, $(m_1, m_2)(1) = (1, 2), (m_1, m_2)(2) = (1, 3), \ldots, (m_1, m_2)(n-1) = (1, n), (m_1, m_2)(n) = (2, 3), \ldots, (m_1, m_2)(n_A) = (n-1, n)$. We set $A_m = A_{(m_1,m_2)(m)}$, having an entry of 1 in row m_1 and column m_2, an entry of -1 in row m_2 and column m_1 and being 0 elsewhere. This choice makes it possible to estimate also block diagonal covariance matrices, and hence allows to enforce certain structures on the estimated covariance matrix without using constraint optimization. For example, if the linear combination of the antisymmetric matrices is block diagonal, also its exponential, i.e., the orthogonal matrix, will be block-diagonal. Hence, all factors in Equation 5.14 share the same block-diagonal shape, and the covariance matrix will also be block-diagonal.

Surjectivity of the Lie algebraic parametrization

It remains to show that the map δ_n from Equation 5.14 is indeed a valid parametrization of the covariance matrices, i.e., that all covariance matrices can be described in this way. In other words, the aim is to prove the following theorem:

Theorem 1. *Let $n \in \mathbb{N}$ and δ_n the map from Equation 5.14 for a basis of the antisymmetric matrices $\{A_m\}_{m=1,\ldots,n_A}$. Then the map δ_n is surjective and locally injective around 0.*

In order to prove Theorem 1, we need some results from Lie theory about the exponential map. The first one concerns compact Lie groups and is given in, e.g., [Duistermaat and Kolk, 2000] as Corollary 3.1.4:

Lemma 1. *Let G be a finite-dimensional, connected, and compact Lie group and $\mathfrak{g}$ its Lie algebra. Then the exponential map $\exp\colon \mathfrak{g} \longrightarrow G$ is surjective.*

Proof. The proof is given in Chapter 3.1 of [Duistermaat and Kolk, 2000]. □

The second result we will need is that for a (finite-dimensional) Lie algebra $\mathfrak{g}$ of a Lie group G and every $X \in \mathfrak{g}$, the exponential map defines a one-parametric subgroup of G.

Lemma 2. *Let G be a finite-dimensional Lie group, $\mathfrak{g}$ its Lie algebra and $X \in \mathfrak{g}$. Then, the map*

$$\Phi_x\colon \mathbb{R} \longrightarrow G, \quad t \longmapsto \exp(tX) \tag{5.15}$$

defines a one-parametric subgroup of G. In particular, we have for $s, t \in \mathbb{R}$:

$$\Phi_x(s) \cdot \Phi_x(t) = \Phi_x(s+t) = \Phi_x(t) \cdot \Phi_x(s) \tag{5.16}$$

Proof. The proof follows directly from Theorem 1.3.2 and Definition 1.3.3 of the exponential map given in [Duistermaat and Kolk, 2000]. Alternatively, the statements can be found in Proposition 2.3 of [Hall, 2015]. □

Proof of Theorem 1. Let Σ be a symmetric positive definite matrix of size $n \times n$. Then, it has a unique eigenvalue decomposition (unique up to reordering) with eigenvalues $\Lambda_1, \ldots, \Lambda_n > 0$ and an orthogonal matrix U such that

$$\begin{aligned}\Sigma &= U \cdot \operatorname{diag}(\Lambda_1, \ldots, \Lambda_n) \cdot U^T \\ &= U \cdot \operatorname{diag}(\exp(\lambda_1), \ldots, \exp(\lambda_n)) \cdot U^T \\ &= U \cdot \exp\big(\operatorname{diag}(\lambda_1, \ldots, \lambda_n)\big) \cdot U^T\end{aligned}$$

As the exponential map $\exp\colon \mathfrak{so}_n \longrightarrow SO_n$ is surjective (Lemma 1) and $U \in SO_n$, there exists an antisymmetric matrix A such that $\exp(A) = U$. Due to Lemma 2, we have $\exp(-A) = \exp(A)^{-1}$. As $U^T = U^{-1}$, the map δ given in Equation 5.14 is indeed a surjective parametrization of the symmetric positive definite matrices. Local injectivity of δ_n is ensured be the local injectivity of the exponential map, the bijectivity of the parametrization of antisymmetric matrices and the uniqueness (up to reordering) of the eigenvalue decomposition. □

It is important to note that the proposed parametrization δ is not globally injective: Different sets of $(\alpha_1, \ldots, \alpha_{n_A})$ may yield the same rotation matrix after exponentiation. As discussed previously, uniqueness in the parametrization of Σ may be helpful in certain cases, but it is not crucial, when parametrizing covariance structures in ODE MEMs. There, the most interesting properties are the variances and the correlations of the random effects rather than the parameters themselves.

Implementation into the framework SPToolbox

The computational toolbox SPToolbox is a MATLAB framework that can be linked to the mixed-effect toolbox MEMOIR [Wang et al., 2019]. SPToolbox computes in-silico populations based on Monte Carlo sampling, the sigma point method [van der Merwe, 2004], or Dirac mixture distributions [Gilitschenski and Hanebeck, 2013]. It transforms these in-silico populations, which are created based on standard normal distributions, according to a given covariance matrix, which describes the random effects in the case of a MEM. Originally, SPToolbox was capable of computing these covariance matrices based on the matrix logarithm. As a contribution of this thesis, it was extended by the remaining parametrization approaches which are discussed in this chapter, in particular by the Lie algebraic parametrization.

5.3.2 Benchmarking convergence of covariance parametrizations

In order to assess whether the proposed Lie algebraic approach enjoys the desired convergence properties when estimating symmetric positive definite matrices, the previously discussed four methods for parametrization of covariance matrices and the Lie algebraic method were compared with each other. For this purpose, a benchmark study was created based on 5 benchmark test cases (B1 - B5, Figure 5.7 A) of increasing difficulty, with $d = 4, \ldots, 8$ and $d \times d$ describing the shape of the matrix. Each benchmark study was repeated five times. For each benchmark $B = B1, \ldots, B5$ and each repetition $R = 1, \ldots, 5$, multi-start local optimization was carried out in the following way: A covariance matrix Σ_B^R was randomly generated based on a previously defined correlation pattern (Figure 5.7 A). This covariance matrix was considered to be the

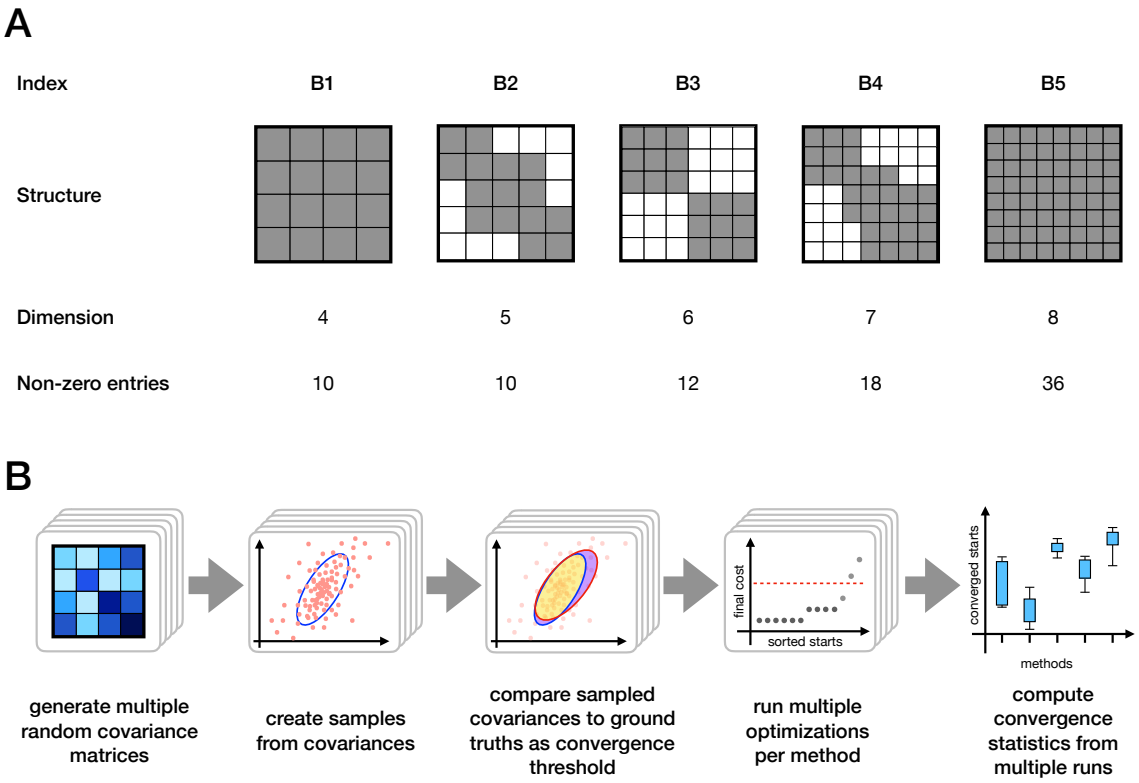

Figure 5.7: Concept of benchmark study for covariance parametrizations. **A** Correlation patterns of benchmark models. Grey field denote non-zero entries. **B** Work flow of benchmark study.

ground truth, from which a population with $10000 \cdot d$ individuals was sampled. From this population, the empirical covariance $\hat{\Sigma}_B^R$ of the sample was computed to mimic a certain level of random measurement noise (Figure 5.7 B). Based on $\hat{\Sigma}_B^R$, the following objective function was defined:

$$J_{\hat{\Sigma}_B^R}(\delta) = \frac{1}{2}\left(\left\|\hat{\Sigma}_B^R - \Sigma_B^{\text{sim}}(\delta)\right\|_F^2 + \left\|\text{diag}\left(\hat{\Sigma}_B^R - \Sigma_B^{\text{sim}}(\delta)\right)\right\|_F^2\right) \tag{5.17}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm, $\text{diag}(\cdot)$ reduces a matrix to its diagonal, and Σ_B^{sim} is the simulated covariance matrix. Hence, the objective function is an adaptation of the Frobenius norm, which accounts for the symmetry of covariance matrices. With this objective function, optimization was carried out using the parameter estimation toolbox PESTO with default options, using 100 local optimization runs in multi-start local optimization with 100 optimization steps each and randomly generated initial guesses based on latin hypercube sampling. An optimization run was considered to be converged, if its final objective function value reached a threshold T_B^R defined by the ground truth:

$$J_{\hat{\Sigma}_B^R}(\delta) < T_B^R = \frac{1}{2}\left(\left\|\hat{\Sigma}_B^R - \Sigma_B^R\right\|_F^2 + \left\|\text{diag}\left(\hat{\Sigma}_B^R - \Sigma_B^R\right)\right\|_F^2\right) \tag{5.18}$$

When assessing the final objective function values for all repetitions and all benchmarks, we found that the overall convergence for each method was mostly conserved across repetitions (Figure 5.8 A). Moreover, the quality of optimization dropped substantially when the dimension of the benchmark problem increased: While for the smallest benchmark problem B1, at least some local optimization runs reached low objective function values in each repetition for all

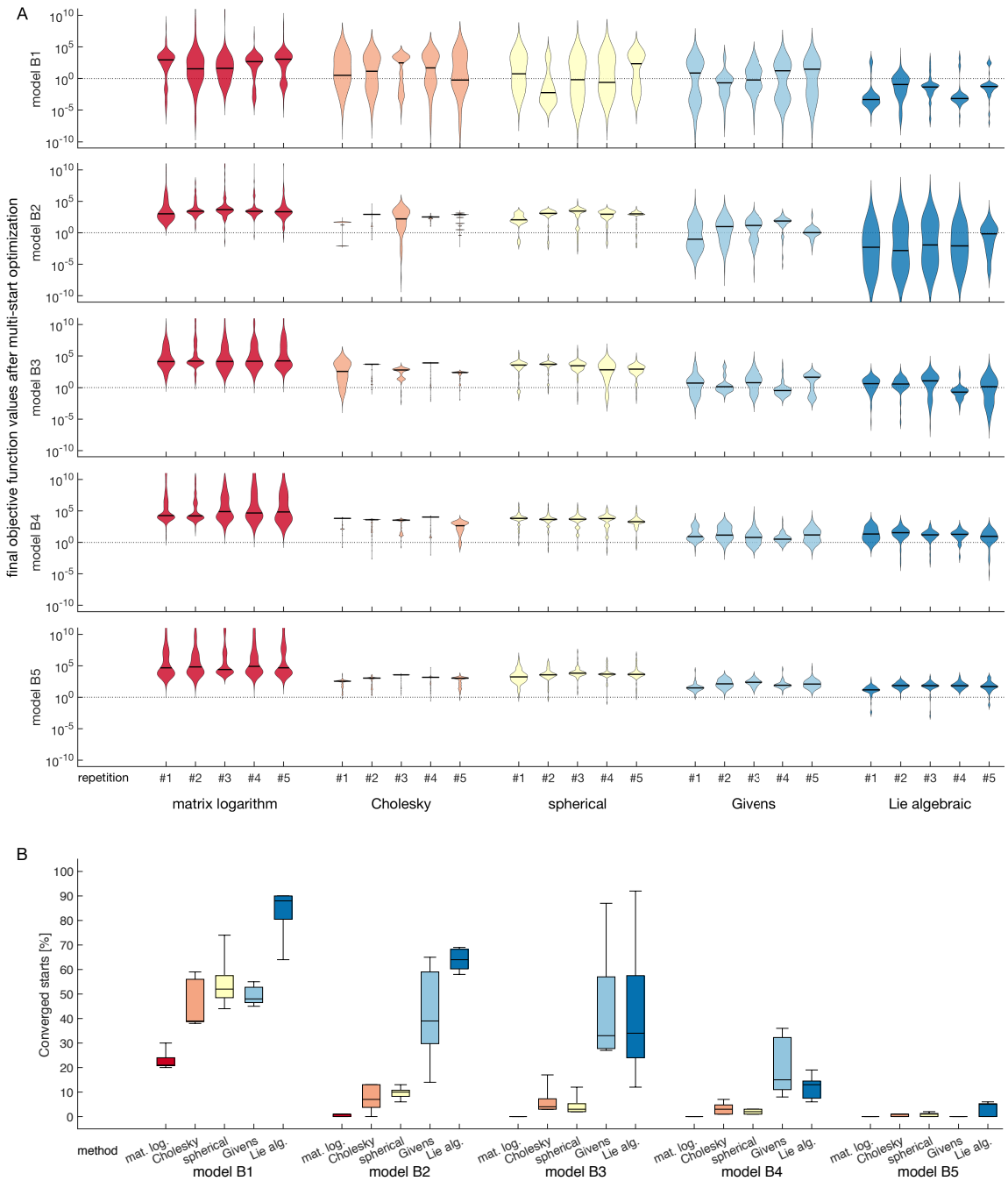

Figure 5.8: Comparison of convergence properties of different covariance parametrizations. **A** Violin plots over final objective function values for all benchmarks and all repetitions across all methods. The solid lines in the violin plots indicate the medians. **B** Converged starts per benchmark model and method. Boxes show 25 and 75 percentiles computed over repetitions, whiskers show minima and maxima, medians are depicted as solid black lines.

methods, convergence was substantially and consistently worse for the largest benchmark B5. When analyzing the total number of converged starts and the variability across repetitions, the matrix logarithm performed worst across all benchmarks (Figure 5.8 B). The Cholesky and the spherical Cholesky parametrization still yielded favorable results for the smallest benchmark B1, but performed considerably worse for the benchmarks B2 to B5. The Givens parametrization and the proposed Lie algebraic approach performed best overall. For B4, the Givens parametrization showed the highest number of converged starts, while for B1, B2, and B5 the Lie algebraic parametrization performed clearly best. For B3, both approaches achieved similar convergence. For the largest benchmark B5, the Lie algebraic parametrization was the only method which was able to find the global optimum at all, in at least three out of the five repetitions.

Overall, the Lie algebraic approach outperformed the other parametrization methods: When comparing the median number of converged starts across repetitions, it was the best method for four of the five benchmarks and the second method for the remaining one. This indicates that the proposed approach may indeed be helpful when inferring covariance structures for ODE MEMs.

5.3.3 Application: Inferring the covariance structure in an ODE MEM of JAK2/STAT5 signaling

We then studied the properties of the Lie algebraic parametrization for the previously discussed models of JAK2/STAT5 signaling in terms of optimization convergence and integration failure more closely and investigated the optimization results for the estimated covariance matrices. The models consisted of 21 parameters which described reaction rate and hence influence the model dynamics. In the finally selected model, six random effects (hereafter called RE_1 to RE_6) were considered. RE_1 and RE_2 described export and import rates of the phosphorylated transcription factor STAT5 from and into the nucleus, which were allowed to be correlated. The random effects RE_3, RE_4, and RE_5 described initial concentrations of the Epo receptor, the protein SHP1, and the total amount of STAT5. RE_6 described a cell dependent offset in the measurement of phosphorylated STAT5, which could also be interpreted to cover a basal activation rate of STAT5. As previously discussed, also RE_3 to RE_6 are were assumed to be correlated between each other, but independent of RE_1 and RE_2. Hence, six eigenvalue parameters and seven parameters for the correlations had to be estimated, yielding in total 33 parameters which influence the model dynamics. Together with observation parameters, such as scaling factors and measurement noise levels, the model comprised 178 parameters and was fitted to a total of 1945 data points.

For this application example, the possibility to control the eigenvalues of the covariance matrices at the initial point of optimization turned out to be crucial. To test the robustness of initial guesses for local optimization runs, 400 initial points were randomly generated by latin hypercube sampling for the matrix logarithm approach, the Cholesky, the spherical, the Givens parametrization, and the Lie algebraic method. For those initial guesses, the computation of the objective function value and its gradients was evaluated. When using the matrix logarithm, ODE integration failure at the initial point of optimization occurred in roughly 64% of the cases, despite a careful adaptation of parameter bounds for the sampling of starting points. For the Cholesky parametrization, ODE integration even failed for about 96% of the initial points. In contrast, the spherical and the Givens parametrization resulted in failure for only 35% and 16% of the cases, respectively. With the Lie algebraic parametrization, this evaluation succeeded for 88% of the starting points and caused ODE integration failure only in 12% of the cases, making it a very robust method for the parametrization of covariance structures in this ODE MEM (Figure 5.9 A). This suggests that parametrization methods, which allow to control the

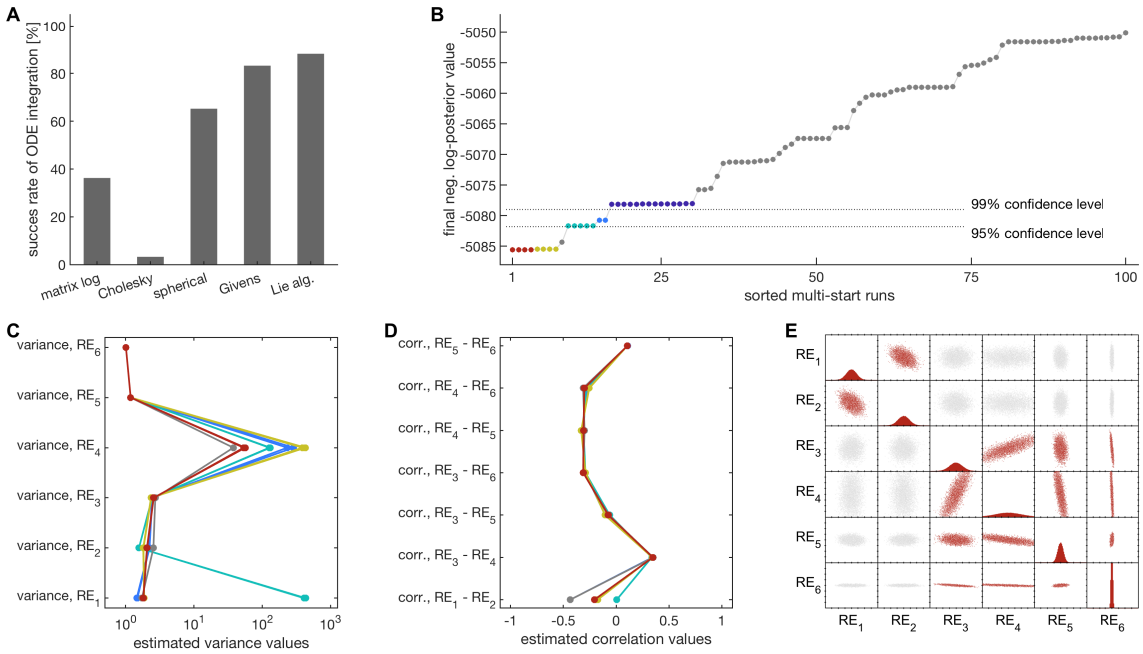

Figure 5.9: Results for parameter estimation of an ODE MEM with non-diagonal covariance matrix. **A** Success rate of ODE integration at initial point of local optimization for different methods of covariance parametrization. **B** Waterfall plot for the ODE MEM of JAK2/STAT5 signaling, using the Lie algebraic parametrization. The best 100 out of 400 starts are depicted, plateaus indicate convergence of optimization to the same local optimum. The dashed lines indicate thresholds for drawing confidence intervals to confidence levels 95% and 99%, respectively, when using the profile likelihood approach. **C** Coordinate plot of estimated variance terms of the covariance matrix of random effects for all optimization results below the threshold of 99% confidence. Coordinate plots which overlay each other prove that the same local optimum was found. **D** Coordinate plot of estimated correlations in the covariance matrix of random effects for all optimization results below the threshold of 99% confidence. **E** Scatter plot for the global optimum of estimated covariance structure. This subfigure was adapted from the author's publication [Adlung et al., 2019], Figure 5 B.

eigenvalues of the covariance matrix or at least the variance terms, are less prone to numerical errors when applied in ODE MEMs.

For the subsequent parameter estimation, multi-start local optimization was used together with the Lie algebraic parametrization method and the trust-region-reflective algorithm of the optimization routine fmincon from the MATLAB optimization toolbox [MathWorks, 2016]. Despite the comparably high number of model parameters (for an ODE MEM) and the non-trivial covariance matrix, parameter estimation showed a favorable convergence behavior in the waterfall plot, indicating clear plateaus for different local minima (Figure 5.9 B). When computing cutoff values for the negative log-posterior for confidence levels of 95% and 99%, which are used to infer practical identifiability in the profile likelihood method [Raue et al., 2009], the final objective function values for all local optimization runs but one indicated reproducible local optima. (Figure 5.9 B). Furthermore, not only the negative log-posterior values were reproducible, but also the entries of the estimated covariance matrices: The variance terms of the different minima overlayed each other in the coordinate plot (Figure 5.9 C), and the estimated correlations as well (Figure 5.9 D). The coincidence of the estimated covariance matrices proved that indeed the same local optima were found, when the same objective function value was obtained, although the parameters which described those covariance matrices did not coincide across local

optimizations. This suggested, that most of the covariance terms (except the variance of random effect 4, which described the initial amount of SHP1) were likely to be practically identifiable. However, as profile likelihood computation was not carried out, due to the high computation times involved, this was only an indication for and not a proof of practical identifiability.

Concerning the biological interpretation of the inferred results, the estimated covariance matrix of the ODE MEM showed pronounced correlation patterns (Figure 5.9 E). Biologically speaking, this argues for strong co-regulations of protein abundances. This was especially the case for the Epo recpector-JAK2 complex and the phosphatase SHP1, for which a strong positive correlation ($\rho = 0.80$) was detected, and for the total abundance of STAT5 and SHP1, for which a negative correlation ($\rho = -0.71$) was predicted. This would indeed be plausible, as proteins which commonly form complexes tend to have correlated degradation rates or may be co-expressed. These results indicate that it is important to not only infer variances of model parameters in an ODE MEM, but to also consider off-diagonal elements, as those may be important to describe the actual biological system.

5.4 Discussion

5.4.1 Summary and conclusion

Mathematical modeling of single-cell dynamics is a young research field with miscellaneous challenges and concepts [Loos and Hasenauer, 2019]. When restricting to single-cell ODE modeling, two of the main conceptional challenges are firstly, how to integrate heterogeneous data types in a consistent statistical framework [Loos and Hasenauer, 2019], and secondly, how to efficiently infer the main aspects of cell-to-cell variability of a population of cells [Pinheiro and Bates, 1996]. In this chapter, two contributions were presented which each address one of these issues.

In a first section, single-cell snapshot data was combined with population average data, as population average data is still the most common data type in dynamical modeling (such as used in [Hass et al., 2019]). This was achieved by reducing single-cell snapshot data to its statistical moments, which allowed a consistent integration with population average data. The approach was implemented into a computational framework and applied to an ODE MEM, which extended a previously published ODE model of JAK2/STAT5 signaling [Bachmann et al., 2011]. The presented framework allowed to identify and better understand the sources of cell-to-cell variability in erythroid progenitor cell, which regulate the response of the erythroid system at the population level, i.e., the production of red blood cells upon stimulation with Erythropoietin.

The combination of population average and single-cell snapshot data in one common statistical framework and parameter estimation of a model to these data types simultaneously had not been possible so far [Loos and Hasenauer, 2019]. Single-cell snapshot data can be collected in a highly multiplexed fashion for many species when compared with other single-cell data types [Giesen et al., 2014, Perfetto et al., 2004]. Hence, this contribution enables the extension of many ODE models, which so far captured only the population average behavior, to single-cell ODE models. In this way, it will be possible to infer more knowledge about the dynamics of many cellular processes at the single-cell level, which can differ substantially and also qualitatively from the behavior at the population level for some systems [Gaudet and Miller-Jensen, 2016, Tay et al., 2010]. This might help to better understand the differences between the response of biological systems at the population and the single-cell level. As the behavior at the single-cell level is more representative for the actual biological process than the dynamics at the population level [Llamosi et al., 2016], removing obstacles in single-cell modeling may leverage our understanding

of many biological systems.

In a second section, a novel Lie algebraic approach to parametrize covariance structures, i.e., symmetric positive definite matrices, was proposed. These matrices are necessary to describe the distribution of cell-to-cell variable components in a cell population [Karlsson et al., 2015]. The proposed method was benchmarked against other existing approaches [Pinheiro and Bates, 1996] and outperformed them for the majority of test cases. Subsequently, the different methods were compared on the model of JAK2/STAT5 signaling, which underlined the robustness of the Lie algebraic method. Finally, the inference results for the covariance structure of cell-to-cell variability in the JAK2/STAT5 model were investigated, which demonstrated that the proposed method also performs well in a challenging practical application.

In ODE modeling of biological systems, reliable and efficient estimation of the model parameters is still one of the key challenges [Kapfer et al., 2019, Kreutz, 2019]. This is even more the case when working with ODE MEMs, as not only the parameters describing unknown reaction rate constants and unknown initial concentrations have to be inferred, but also additional parameters, which describe the heterogeneity of the cell population. These population distribution parameters are typically used to model the covariance structure of the random effects in ODE MEMs [Karlsson et al., 2015]. As their estimation interferes with the estimation of the population average model parameters, such as reaction rate constants, parameter estimation becomes overall substantially more challenging for ODE MEMs [Fröhlich et al., 2018b]. Hence, methods which allow the efficient estimation of covariance structures of random effects are indispensable for efficient ODE mixed-effect modeling. As the presented Lie algebraic approach tends to enjoy better convergence properties than other currently existing methods and also enabled a successful parameter optimization for the ODE MEM of JAK2/STAT5 signaling, we hope that it will be beneficial also in future applications of single-cell ODE modeling.

5.4.2 Open problems for ODE MEMs

Improving the assessment of the population distribution

As already pointed out in Section 5.2, the reduction of single-cell snapshot data to its first and second order moments allows to capture main aspects of cell-to-cell variability, but it does not suffice to infer the shape of the distribution of single-cell measurements accurately. In order to address this shortcoming, further enhancements are necessary. A straight forward approach would be to include higher, e.g., third order, moments into the concept. Additionally estimating the skewness of the distribution of a measured observable might allow to reproduce the data even more accurately [Loos and Hasenauer, 2020, Sahu et al., 2003]. At the same time, including more moments in the proposed approach may lead to predominance of the single-cell data points in the parameter estimation process. As the number of moments to include grows rapidly with the order of the moments considered, it may be desirable to balance their statistical weight with an additional weighting factor, which could be derived from enumerating the extracted moments. A complementary idea would be to consider different population models than multi-variate normal distributions for either the random effects or the measured quantities, such as considered in [Loos and Hasenauer, 2020].

Alternatively, if restricting to first and second order moments and multivariate normal distributions, standard deviations or coefficients of variation and correlations in the measurements could be used to inform the parameter estimation process, instead of the variances and covariances [Toni and Tidor, 2013]. A priori, it is unclear whether one or the other approach reflects the behavior of a cell population more appropriately, and hence this may be subject of a further

study.

Correlated noise model for single-cell snapshot data

In the application example of JAK2/STAT5 signaling, measurements of total and phosphorylated STAT5 were carried out by multiplexed flow cytometry. Yet, the measurement noise of those data points was considered to be independent although correlated models of measurement noise might have been appropriate (such as proposed in [Sommerlade et al., 2015]): In multiplexed flow cytometry, different biochemical species are measured simultaneously and hence, measurement noise may show dependencies for those species [Herzenberg et al., 2006]. However, correlated noise models require the estimation of covariance matrices and their inversion, which makes them numerically challenging. For this task, the Lie algebraic parametrization approach proposed in this thesis may be beneficial. Hence, combining correlated noise models with the Lie algebraic parametrization might be an interesting enhancement for future applications.

Furthermore, it is a known problem that the data obtained from multiple single-cell measurements tends to be more clustered within each experiment than across experiments, which is known as the batch effect [Hicks et al., 2015]. Different methods have been developed to cope with this problem, especially when analyzing single-cell sequencing data [Büttner et al., 2019]. A hierarchical structure for the noise model, which considers batch effects of measurements may hence also be appropriate in ODE MEMs, especially when dealing with single-cell time-lapse or single-cell time-lapse statistics data.

Extension of the moment-based approach to single-cell time-lapse statistics data

In principle, the idea of moment extraction from single cell data would also be applicable for the data type of single-cell time-lapse statistics, such as present in [Filippi et al., 2016]. In addition to the covariance structure of measurements for each time point, also the correlation across different time points has to be inferred in this data type. However, the extraction of these moments for the time correlation yields a substantial increase in the number of data points, which makes this data type substantially more challenging to work with. Hence, concepts for weighting these data points against those coming from other measurement techniques are likely to be necessary [Loos and Hasenauer, 2019]. Yet, schemes based on the enumeration of statistical moments and an additional decay of the weights over the time series might allow the integration of even single-cell time-lapse statistics with single-cell snapshot and population average data.

5.4.3 Outlook and further research

Combination of the moment-based approach with single-cell time-lapse data

Although the moment based approach allows to combine heterogeneous data types in one parameter estimation problem, it may fail to allow a straight forward integration of single-cell time-lapse data. Single-cell time-lapse data differs substantially from the other data types, as each single cell can be identified over time, and hence any approach, that restricts to only relying on the distribution of the cell population, will miss important information in these data. For this reason, a specific likelihood (as shown in Equation 5.1 and discussed in [Karlsson et al., 2015]) is used for ODE MEMs of single-cell time-lapse data.

Unfortunately, this likelihood for single-cell time-lapse data in ODE MEMs is problematic by itself, as it relies on numerically hardly tractable integrals, which require approximations [Pinheiro, 1994]. For this reason, maximum likelihood estimation for ODE MEMs is an active

field of research itself. Different approaches have been proposed to cure this problem, which rely either on a modified expectation-maximization algorithm [Chan et al., 2011, Kuhn and Lavielle, 2005] or try to circumvent MEMs completely [Dharmarajan et al., 2019]. Hence, even combining single-cell time-lapse data with other data types in a common statistical framework is highly challenging and a currently unresolved problem.

Establishing a collection of benchmark models for ODE MEMs

It was already mentioned that various approaches and frameworks for deterministic single-cell modeling exist [Filippi et al., 2016, Fröhlich et al., 2018b, Llamosi et al., 2016, Loos et al., 2018b]. However, these different methods have, to the best of the author's knowledge, never been thoroughly compared against each other. Moreover, no computational toolbox, which would be able to apply all of these concepts is available, and currently the best competitors are software packages under proprietary licenses [Antony, 2019, ICON Development Solutions, 2020]. This makes it almost impossible to properly benchmark computational methods against each other, which in turn make it challenging for a modeler to identify the best computational approach to address a specific scientific question. At the same time, the aspect of reproducibility, reusability, and comparability of computational models are becoming increasingly important in systems biology [Kapfer et al., 2019, Kreutz, 2019].

For classic ODE models, a collection of benchmark models has recently been proposed [Hass et al., 2019]. This collection allows to address many methodological questions by thoroughly testing them on a large sample of real applications. A similar benchmark collection of single-cell ODE models, comprising a set of published models and datasets [Almquist et al., 2015, Dharmarajan et al., 2019, Filippi et al., 2016, Fröhlich et al., 2018b, Hasenauer et al., 2014, Karlsson et al., 2015, Llamosi et al., 2016, Loos et al., 2018b] would be an important next step for the scientific community. Such a benchmark collection would also necessitate a common standard for the formulation of single-cell ODE models, the corresponding data, and the parameter estimation problem, similar to the PEtab format, which has recently been proposed for population average ODE models [Schmiester et al., 2020]. Ideally, this format would come together with a flexible and freely available computational toolbox for single-cell model inference, which could cope with at least most of the common model types, which have been used in recent publications. Similar to PEtab, which is based upon the SBML standard [Hucka et al., 2003], a standard for the definition of parameter estimation problems of ODE models could be based upon the pharmML standard [Swat et al., 2015], which would be flexible enough to cover most of the model types. However, also a simple and efficient standard format for the measurement data would have be defined, which is non-trivial.

Given such a benchmark collection, a common format, and a toolbox, a set of highly interesting questions could possibly be answered:

$Q1$ How do the different modeling approaches for single-cell time-lapse data [Dharmarajan et al., 2019, Karlsson et al., 2015] relate to each other and is one of these approaches more appropriate or computationally less expensive than the rest? An answer to this question which is based on more than one or two models might allow a more efficient model development and more reliable conclusions about biological systems at the single-cell level in the future.

$Q2$ How does the type of single-cell data influence the parameter inference and the biological conclusions which can be drawn from a fitted model? Or, more precisely: If a model was originally informed with single-cell time-lapse data, what is the effect of reducing this

data to single-cell time-lapse statistics or even single-cell snapshot data? Answering this question might allow a better interpretation of single-cell models.

*Q*3 How does the chosen model for the covariance structures, i.e., a diagonal, full, or structured covariance matrix, in single-cell models influence the result of parameter estimation and hence biological conclusions?

*Q*4 How do the different approaches for covariance matrix parametrization perform in a more exhaustive benchmark study based on realistic application examples?

As pointed out, properly addressing these questions is hardly possible without a sufficiently large set of realistic benchmarks. For this reason, a community effort to create a common model and data collection and to allow for interoperability between the currently exiting frameworks via a common format is likely to be the one of the most important challenges in single-cell ODE modeling at the moment [Loos and Hasenauer, 2019].

Chapter 6

Discussion

6.1 Summary and Conclusion

Mathematical modeling of biological processes is an important tool to describe and understand cellular biology at a systems level [Kitano, 2002]. This thesis was concerned with the development of efficient computational methods, which is necessary due to the steady increase of size and complexity of such mathematical models based on ordinary differential equations (ODEs) [Fröhlich et al., 2018a, Kapfer et al., 2019]. In Chapter 1, five current challenges were mentioned, which arise when dealing with ODE based models and the inference of unknown parameters. After giving an overview about currently existing computational methods in Chapter 2, these five challenges were addressed by the methodological contributions presented in Chapter 3, 4, and 5.

In Chapter 3, the concept of mini-batch optimization was transferred from the field of deep learning to parameter optimization in ODE models. Adaptations of common mini-batch optimization algorithms were presented, which allowed to efficiently employ these algorithms also in ODE constrained optimization with its specific challenges. These adapted algorithms were implemented into a parallelized parameter estimation framework and tested on a set of benchmark problems, from which conclusions about the most important hyperparameters of mini-batch optimization were drawn. This allowed to scale up parameter estimation for a large-scale ODE model of cancer signaling to a dataset from a public database of unprecedented scale. At the same time, mini-batch optimization allowed to employ ensemble methods for generating model predictions, which were compared to predictions based on point estimates. In this comparison, ensemble methods turned out to be substantially more reliable for at least this particular large-scale ODE model.

In Chapter 4, adjoint sensitivity analysis was extended to second order. Adjoint sensitivity analysis is currently the most efficient method to compute the gradient of an objective function which depends on the solution of an ODE. Analogously, second order adjoint sensitivity analysis allows to compute the Hessian of such an objective function at substantially lower cost than this was possible before [Stapor et al., 2018a]. The corresponding equations for time-discrete measurements were derived and implemented into a freely available ODE solver toolbox. Then, second order adjoint sensitivity analysis was applied in the context of parameter optimization and profile likelihood computation, where it was shown to improve the robustness and efficiency of existing algorithms. Moreover, a hybrid approach for profile likelihood computation, which combines the advantages of the two currently existing methods, was proposed, implemented into the freely available parameter estimation toolbox PESTO [Stapor et al., 2018b] and tested on

two published models, which showed it to outperform the current alternatives [Stapor et al., 2018a].

In Chapter 5, two challenges in the field of ODE mixed-effect modeling of biological processes at the single-cell level were addressed. Firstly, a combined likelihood based on population average data and first and second order moments of single-cell snapshot data was proposed, which allowed to integrate these two heterogeneous data types into a statistical model and to estimate model parameters based on both data types simultaneously. Secondly, a novel Lie algebraic method to parametrize symmetric positive definite matrices, which are used to describe the covariance structure of cell populations, was introduced, which enjoys good convergence properties during parameter estimation. These two contributions allowed to carry out parameter estimation and model selection for an of ODE MEM of JAK2/STAT5 signaling, which described the survival decisions of erythroid progenitor cells at the single-cell level and which allowed to identify sources to cell-to-cell heterogeneity and their impact on the survival mechanism [Adlung et al., 2019].

Overall, the contributions in this thesis substantially improved the scalability of a series of mathematical approaches in parameter estimation and uncertainty analysis for large-scale and medium-scale ODE models, as well as for single-cell ODE MEMs. It enabled especially parameter estimation for some model classes, where this was computationally extremely challenging before. By this means, it enabled to further push the boundaries of what is feasible in deterministic mathematical modeling of biologically systems. Hopefully, these advances will allow the development of larger, more complex and more realistic models of cellular processes and will contribute to a better understanding of some of the problems, which are currently faced in medicine, pharmacology, and healthcare.

6.2 Outlook and future directions

In the individual chapters of this thesis, we outlined further research questions and possible extensions of the proposed contributions, such as next steps concerning mini-batch optimization of ODE models, extension of adjoint sensitivity analysis to discrete events, or the integration of additional types of single-cell data in ODE mixed-effect models. However, it will also be critical to tackle complementary challenges, which were not addressed in this thesis but are yet linked to it, as they either relate to multiple of its chapters or match the same goals as its contributions. Here, a broader outlook on the most important of these further challenges shall be given.

In some applications, large-scale ODE models are trained on data which was collected when the modeled system is considered to be in steady-state [Fröhlich et al., 2018a, Gopalakrishnan et al., 2020, Khodayari and Maranas, 2016, Schmiester et al., 2019a]. As parameter estimation for steady-state data is a special case, it is possible to apply a set of mathematical simplifications and to employ additional methods, some of which may be more powerful than the approaches which were discussed in this thesis. For some models, integration of the underlying ODE can be circumvented and direct root finding methods, auch as Newton's method, can be applied [Fiedler et al., 2016, Gopalakrishnan et al., 2020]. It was shown that for large-scale ODE models, inferring the steady-state via Newton's method can accelerate computations by up to two orders of magnitude [Lines et al., 2019]. However, it may be that a model possesses conserved quantities, which have to be removed prior to applying Newton's method [Vallabhajosyula et al., 2006]. For this reason, some computational toolboxes provide implementations of such methods [Fröhlich et al., 2019, Hoops et al., 2006], if only the steady-state of a system is needed rather than the full time-course. Yet, when inferring the objective function gradient based on steady-state data, a part of the speed-up may be lost, as the computation of state sensitivities requires solving a

large linear system of equations. This steady-state sensitivity approach is still substantially faster than a full integration of forward sensitivities, but not necessarily faster than adjoint sensitivity analysis [Lines et al., 2019]. Hence, combining adjoint sensitivity analysis with steady-state sensitivity analysis may be highly beneficial when working with large-scale systems.

Another problem concerning large-scale models is how to collect enough measurement data to properly constrain model parameters. Large public databases exist [Barretina et al., 2012, Cancer Genome Atlas Network, 2012, Yang et al., 2013], which can be exploited for this task. However, those databases are mostly restricted to cancer cell lines and often only comprise steady-state data. Hence, even for cancer models, these datasets may be insufficient to render parameters and model predictions identifiable, as information about the temporal evolution is missing. Thus, it is an important challenge to collect and exploit more sources of information: One possibility might be including prior information for parameters from existing databases, such as SABIO-RK [Wittig et al., 2012] or BRENDA [Schomburg et al., 2013]. For large-scale models, it is unlikely that this can be done by hand. Tools which collect these priors automatically and map them to an annotated model would be highly beneficial. Another option would be including qualitative data, which does not compare model outputs with quantitative measurements, but rather with findings which can be retrieved from literature screenings from publication texts. Similar tools, which can be used for semi-automated model assembly have recently been developed [Gyori et al., 2017]. Such qualitative data may concern inequality constraints on models outputs or the ordering of observation rather than their quantitative values. In recent years, some approaches have been presented which can exploit such qualitative data [Mitra et al., 2018, Schmiester et al., 2019b] and which may turn out promising in the near future. However, also for this topic, many problems persist, as it is, e.g., not yet possible to combine qualitative and quantitative data into one statistical framework and also the formulation of a noise model for qualitative data is currently problematic.

Generally, when developing mathematical methods for ODE models, a thorough benchmarking of existing approaches and their implementations would be extremely beneficial [Kreutz, 2016, 2019]. With the recent advances concerning a common standard for parameter estimation problems [Schmiester et al., 2020] and the establishment of a benchmark collection of published ODE models [Hass et al., 2019], it is finally possible to properly compare computational toolboxes against each other. This might also lead to a substantial boost in method and model development, as modelers will be able to choose the best framework to analyze their system of study.

Scalability of models and methods will also be an important aspect in single-cell modeling. Here, method and toolbox development seems not yet to be as mature as for classic ODE models. Making models and tools interoperable and analyzing which are the methods of choice to address specific problems would be an important achievement [Loos and Hasenauer, 2019]. For example, the efficiency of existing approaches for parameter optimization of ODE MEMs has – to the best of the author's knowledge – never been benchmarked in systems biology, and similar problems exist for methods for uncertainty analysis. Hence, the earlier mentioned benchmark collection of single-cell ODE models and a common standard for the definition of a parameter estimation problem, which might, e.g., rely on the model standard pharmML [Swat et al., 2015], could provide substantial novel insights and boost the field.

A further question in single-cell modeling may be how to best combine adjoint sensitivity analysis [Fröhlich et al., 2017a, Sengupta et al., 2014] with models of cell populations, such as sigma point methods [van der Merwe, 2004] or Dirac mixture models [Gilitschenski and Hanebeck, 2013]. In principle, this would be possible, but so far, no implementation of this concept exists. When

scaling up ODE MEMs to larger systems, the development of adjoint sensitivity analysis for cell population models might become an enabling step in parameter estimation of these models. In the future, this might allow a holistic understanding of large-scale systems at the single-cell level.

Seen from a higher level, the challenges which were discussed so far can and will be addressed in the near and midterm future. They are either concerned with the question, how computational efficiency in large-scale modeling can be improved in order to circumvent the need for large-scale computing clusters, which have been used in, e.g., [Penas et al., 2015, Schmiester et al., 2019a]. Or they relate to progresses in single-cell ODE modeling, in order to better understand the behavior of biological systems at the single-cell level, which is often argued to be biologically more fundamental than the population level [Llamosi et al., 2016, Tay et al., 2010]. In the long run however, it will be crucial to improve our capability of combining models which reflect different layers of biological processes: Those which describe the fine-grained behavior inside of cells [Kholodenko, 2007] and those which describe the spatial or tissue specific context and capture the coarse-grained behavior [Starruß et al., 2014]. This area of multi-scale or multi-level modeling is currently a highly active field of research [Hasenauer et al., 2015, Imle et al., 2019, Jagiella et al., 2017, Yu and Bagheri, 2020] and will gain importance, as using different layers of models is the only way to reflect complex systems in detail and sufficiently close to reality for drawing conclusions about the functioning of biology. And a thorough understanding of complex biological systems is urgently needed to tackle the pressing questions, which we face in biology, pharmacology, and medicine, and which we need to answer in order to improve healthcare and to save lives.

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

L. Adlung, S. Kar, M.-C. Wagner, B. She, S. Chakraborty, J. Bao, S. Lattermann, M. Boerries, H. Busch, P. Wuchter, A. D. Ho, J. Timmer, M. Schilling, T. Höfer, and U. Klingmüller. Protein abundance of AKT and ERK pathway components governs cell type-specific regulation of proliferation. *Mol. Syst. Biol.*, 13(1):904, 2017.

L. Adlung, P. Stapor, C. Tönsing, L. Schmiester, L. E. Schwarzmüller, D. Wang, J. Timmer, U. Klingmüller, J. Hasenauer, and M. Schilling. Cell-to-cell heterogeneity in JAK2/STAT5 pathway components and cytoplasmic volumes decide about life or death of erythroid progenitor cells. *bioRxiv preprint, doi.org/10.1101/866871*, November 2019.

B. B. Aldridge, J. M. Burke, D. A. Lauffenburger, and P. K. Sorger. Physicochemical modelling of cell signalling pathways. *Nat. Cell Biol.*, 8(11):1195–1203, November 2006.

J. Almquist, L. Bendrioua, C. B. Adiels, M. Goksör, S. Hohmann, and M. Jirstrand. A nonlinear mixed effects approach for modeling the cell-to-cell variability of Mig1 dynamics in yeast. *PLoS ONE*, 10(4):1–32, April 2015.

G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceeding AFIPS 67*, volume Proceedings of the April 18–20, 1967, Spring Joint Computer Conference, pages 483–485, April 1967.

N. Andrei. Accelerated conjugate gradient algorithm with finite difference hessian/vector product approximation for unconstrained optimization. *J. Comput. Appl. Math.*, 230(2):570–582, 2009.

F. L. S. Antony. Monolix version 2019r1. antony, france: Lixoft sas, 2019. http://lixoft.com/products/monolix/, 2019.

L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pac. J. Math.*, 16(1):1–3, 1966.

A. C. Babtie and M. P. H. Stumpf. How to deal with parameters for whole-cell modelling. *J. R. Soc. Interface*, 14(133), 2017.

J. Bachmann, A. Raue, M. Schilling, M. E. Böhm, C. Kreutz, D. Kaschek, H. Busch, N. Gretz, W. D. Lehmann, J. Timmer, and U. Klingmüller. Division of labor by dual feedback regulators controls JAK2/STAT5 signaling over broad ligand range. *Mol. Syst. Biol.*, 7(1):516, July 2011.

B. Ballnus, S. Hug, K. Hatz, L. Görlitz, J. Hasenauer, and F. J. Theis. Comprehensive benchmarking of Markov chain Monte Carlo methods for dynamical systems. *BMC Syst. Biol.*, 11 (63), June 2017.

B. Ballnus, S. Schaper, F. J. Theis, and J. Hasenauer. Bayesian parameter estimation for biochemical reaction networks using region-based adaptive parallel tempering. *Bioinf.*, 34(13): i494–i501, July 2018.

E. Balsa-Canto and J. R. Banga. AMIGO, a toolbox for advanced model identification in systems biology using global optimization. *Bioinf.*, 27(16):2311–2313, Aug. 2011.

E. Balsa-Canto, J. R. Banga, A. A. Alonso, and V. S. Vassiliadis. Dynamic optimization of chemical and biochemical processes using restricted second-order information. *Comput. Chem. Eng.*, 25(4):539–546, 2001.

E. Balsa-Canto, J. R. Banga, A. A. Alonso, and V. S. Vassiliadis. Dynamic optimization of distributed parameter systems using second-order directional derivatives. *Ind. Eng. Chem. Res.*, 43:6756 – 6765, 2004.

J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, A. A. Margolin, S. Kim, C. J. Wilson, J. Lehár, G. V. Kryukov, D. Sonkin, A. Reddy, M. Liu, L. Murray, M. F. Berger, J. E. Monahan, P. Morais, J. Meltzer, A. Korejwa, J. Jané-Valbuena, F. A. Mapa, J. Thibault, E. Bric-Furlong, P. Raman, A. Shipway, I. H. Engels, J. Cheng, G. K. Yu, J. Yu, P. Aspesi, Jr, M. de Silva, K. Jagtap, M. D. Jones, L. Wang, C. Hatton, E. Palescandolo, S. Gupta, S. Mahan, C. Sougnez, R. C. Onofrio, T. Liefeld, L. MacConaill, W. Winckler, M. Reich, N. Li, J. P. Mesirov, S. B. Gabriel, G. Getz, K. Ardlie, V. Chan, V. E. Myer, B. L. Weber, J. Porter, M. Warmuth, P. Finan, J. L. Harris, M. Meyerson, T. R. Golub, M. P. Morrissey, W. R. Sellers, R. Schlegel, and L. A. Garraway. The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, Mar. 2012.

D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and Its Applications*. Wiley Series in Probability and Statistics. John Wiley & Sons, 1988.

M. J. Bayarri and J. Berger. The interplay of bayesian and frequentist analysis. *Stat. Sci.*, 19: 58–80, 2004.

S. Beal and L. Sheiner. The nonmem system. *Am. Stat.*, 34(2):118–119, 1980.

V. Becker, M. Schilling, J. Bachmann, U. Baumann, A. Raue, T. Maiwald, J. Timmer, and U. Klingmüller. Covering a broad dynamic range: information processing at the erythropoietin receptor. *Science*, 328(5984):1404–1408, June 2010.

H. H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proc. of the 9th IFAC World Congress, Budapest, Hungary*, pages 242–247, July 1984.

R. Boiger, J. Hasenauer, S. Hross, and B. Kaltenbacher. Integration based profile likelihood calculation for PDE constrained parameter estimation problems. *Inverse Prob.*, 32(12):125009, Dec. 2016.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311, 2017.

M. Bouhaddou, A. M. Barrette, A. D. Stern, R. J. Koch, M. S. DiStefano, E. A. Riesel, L. C. Santos, A. L. Tan, A. E. Mertz, and M. R. Birtwistle. A mechanistic pan-cancer pathway model informed by multi-omics data interprets stochastic cell fate responses to drugs and mitogens. *PLoS Comput. Biol.*, 14(3):e1005985, March 2018.

S. Boyd and L. Vandenberghe. *Convex Optimisation*. Cambridge University Press, UK, 2004.

M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM J. Sci. Comput.*, 21(1):1–23, 1999.

C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Math. Comput.*, 19(92):577–593, October 1965.

A. Butler, P. Hoffman, P. Smibert, E. Papalexi, and R. Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nat. Biotechnol.*, 36(5): 411–420, June 2018.

M. Büttner, Z. Miao, F. A. Wolf, S. A. Teichmann, and F. J. Theis. A test metric for assessing single-cell RNA-seq batch correction. *Nat. Methods*, 16(1):43, 2019.

R. H. Byrd, R. B. Schnabel, and G. A. Shultz. Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Math. Program.*, 40(1):247–263, 1988.

R. H. Byrd, H. F. Khalfan, and R. B. Schnabel. Analysis of a symmetric rank-one trust region method. *SIAM J. Optim.*, 6(4):1025–1039, 1996.

R. H. Byrd, J. C. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Math. Program.*, 89(1):149–185, Nov 2000.

Cancer Genome Atlas Network. Comprehensive molecular portraits of human breast tumours. *Nature*, 490(7418):61–70, October 2012.

C. Cartis, N. I. M. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Math. Program. Ser. A*, 127(2):245–295, 2011.

P. L. Chan, P. Jacqmin, M. Lavielle, L. McFadyen, and B. Weatherley. The use of the SAEM algorithm in MONOLIX software for estimation of population pharmacokinetic-pharmacodynamic-viral dynamics parameters of maraviroc in asymptomatic HIV subjects. *J. Pharmacokinet. Pharmacodyn.*, 38(1):41–61, Feb. 2011.

C. Chassagnole, N. Noisommit-Rizzi, J. W. Schmid, K. Mauch, and M. Reuss. Dynamic modeling of the central carbon metabolism of escherichia coli. *Biotechnol Bioeng*, 79(1):53–73, 2002.

J.-S. Chen and R. I. Jennrich. The signed root deviance profile and confidence intervals in maximum likelihood analysis. *J. Am. Stat. Assoc.*, 91(435):993–998, September 1996.

J.-S. Chen and R. I. Jennrich. Simple accurate approximation of likelihood profiles. *J. Comput. Graphical Statist.*, 11(3):714–732, 2002.

W. W. Chen, B. Schoeberl, P. J. Jasper, M. Niepel, U. B. Nielsen, D. A. Lauffenburger, and P. K. Sorger. Input–output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data. *Mol. Syst. Biol.*, 5(1):239, January 2009.

O.-T. Chis, J. R. Banga, and E. Balsa-Canto. Structural identifiability of systems biology models: A critical comparison of methods. *PLoS ONE*, 6(11):e27755, November 2011.

K. Choi, J. K. Medley, M. König, K. Stocking, L. Smith, S. Gu, and H. M. Sauro. Tellurium: An extensible python-based modeling environment for systems and synthetic biology. *Biosystems*, 171:74–79, 2018.

J. Chung, M. Chung, and D. P. O'Leary. Optimal regularized low rank inverse approximation. *Linear Algebra Appl.*, 468:260–269, 2015.

J. Chung, M. Chung, J. T. Slagel, and L. Tenorio. Stochastic newton and quasi-newton methods for large linear least-squares problems. *arXiv preprint, arXiv:1702.07367*, February 2017a.

M. Chung, J. Krueger, and M. Pop. Identification of microbiota dynamics using robust parameter estimation methods. *Math. Biosci.*, 294:71–84, December 2017b.

T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM J. Optim.*, 6:418–445, 1996.

J. C. Costello, L. M. Heiser, E. Georgii, M. Gonen, M. P. Menden, N. J. Wang, M. Bansal, M. Ammad-ud din, P. Hintsanen, S. A. Khan, J.-P. Mpindi, O. Kallioniemi, A. Honkela, T. Aittokallio, K. Wennerberg, N. D. Community, J. J. Collins, D. Gallahan, D. Singer, J. Saez-Rodriguez, S. Kaski, J. W. Gray, and G. Stolovitzky. A community effort to assess and improve drug sensitivity prediction algorithms. *Nat. Biotech.*, 32(12):1202–1212, 12 2014.

Y. N. Dauphin, R. Pascanu, C. Gulcehre, and K. Cho. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems 26 (NIPS 2014)*, pages 2933–2941, 2014.

M. De La Maza and D. Yuret. Dynamic hill climbing. *AI expert*, 9:26–26, 1994.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 27, pages 1646–1654, 2014.

J. E. Dennis, Jr., D. M. Gay, and R. E. Welsch. Algorithm 573: Nl2sol—an adaptive nonlinear least-squares algorithm [E4]. *ACM T. Math. Software.*, 7(3):369–383, September 1981.

L. Dharmarajan, H.-M. Kaltenbach, F. Rudolf, and J. Stelling. A simple and flexible computational framework for inferring sources of heterogeneity from single-cell dynamics. *Cell Syst.*, 2019.

A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Stat. Comp.*, 10(3):197–208, July 2000.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(12):2121–2159, 2011.

J. J. Duistermaat and J. A. C. Kolk. *Lie Groups*. Springer-Verlag Berlin Heidelberg, 2000.

F. Eduati, V. Doldàn-Martelli, B. Klinger, T. Cokelaer, A. Sieber, F. Kogera, M. Dorel, M. J. Garnett, N. Blüthgen, and J. Saez-Rodriguez. Drug resistance mechanisms in colorectal cancer dissected with cell type-specific dynamic logic models. *Cancer Res.*, 77(12):3364–3375, 2017.

J. A. Egea, M. Rodriguez-Fernandez, J. R. Banga, and R. Marti. Scatter search for chemical and bio-process optimization. *J. Global Optim.*, 37(3):481–503, October 2007.

J. A. Egea, D. Henriques, T. Cokelaer, A. F. Villaverde, A. MacNamara, D. P. Danciu, J. R. Banga, and J. Saez-Rodriguez. MEIGO: An open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. *BMC Bioinf.*, 15(136), 2014.

M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, August 2002.

O. Eriksson, A. Jauhiainen, S. M. Sasane, A. Kramer, A. G. Nair, C. Sartorius, and J. H. Kotaleski. Uncertainty quantification, propagation and characterization by bayesian analysis combined with global sensitivity analysis applied to dynamical intracellular pathway models. *Bioinf.*, 35(2):284–292, 2018.

A. Fiedler, S. Raeth, F. J. Theis, A. Hausser, and J. Hasenauer. Tailored parameter optimization methods for ordinary differential equation models with steady-state constraints. *BMC Syst. Biol.*, 10(80), August 2016.

S. Filippi, C. P. Barnes, P. D. W. Kirk, T. Kudo, K. Kunida, S. S. McMahon, T. Tsuchiya, T. Wada, S. Kuroda, and M. P. Stumpf. Robustness of MEK-ERK dynamics and origins of cell-to-cell variability in MAPK signaling. *Cell Reports*, 15(11):2524–2535, June 2016.

D. S. Fischer, A. K. Fiedler, E. Kernfeld, R. M. J. Genga, J. Hasenauer, R. Maehr, and F. J. Theis. Inferring population dynamics from single-cell rna-sequencing time series data. *Nat. Biotechnol.*, 37:461–468, 2019.

R. A. Fisher. The correlation between relatives on the supposition of mendelian inheritance. *Trans. R. Soc. Edinb.*, 52:399 – 433, 1918.

R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philos. Trans. R. Soc. London, Ser. A*, 222:309–368, 1922.

R. Fletcher and M. J. Powell. A rapidly convergent descent method for minimization. *Comput. J.*, 6(2):163–168, 1963.

F. Fröhlich, F. J. Theis, and J. Hasenauer. Uncertainty analysis for non-identifiable dynamical systems: Profile likelihoods, bootstrapping and more. In P. Mendes, J. O. Dada, and K. O. Smallbone, editors, *Proc. 12th Int. Conf. Comp. Meth. Syst. Biol.*, Lecture Notes in Bioinformatics, pages 61–72. Springer International Publishing Switzerland, November 2014.

F. Fröhlich, P. Thomas, A. Kazeroonian, F. J. Theis, R. Grima, and J. Hasenauer. Inference for stochastic chemical kinetics using moment equations and system size expansion. *PLoS Comput. Biol.*, 12(7):e1005030, July 2016.

F. Fröhlich, B. Kaltenbacher, F. J. Theis, and J. Hasenauer. Scalable parameter estimation for genome-scale biochemical reaction networks. *PLoS Comput. Biol.*, 13(1):e1005331, January 2017a.

F. Fröhlich, F. J. Theis, J. O. Rädler, and J. Hasenauer. Parameter estimation for dynamical systems with discrete events and logical operations. *Bioinf.*, 33(7):1049–1056, April 2017b.

F. Fröhlich, T. Kessler, D. Weindl, A. Shadrin, L. Schmiester, H. Hache, A. Muradyan, M. Schütte, J.-H. Lim, M. Heinig, F. J. Theis, H. Lehrach, C. Wierling, B. Lange, and J. Hasenauer. Efficient parameter estimation enables the prediction of drug response using a mechanistic pan-cancer pathway model. *Cell Syst.*, 7(6):567–579.e6, December 2018a.

F. Fröhlich, A. Reiser, L. Fink, D. Woschée, T. Ligon, F. J. Theis, J. O. Rädler, and J. Hasenauer. Multi-experiment nonlinear mixed effect modeling of single-cell translation kinetics after transfection. *npj Syst. Biol. Appl.*, 5(1):1, 2018b.

F. Fröhlich, C. Loos, and J. Hasenauer. Scalable inference of ordinary differential equation models of biochemical processes. In G. Sanguinetti and V. A. Huynh-Thu, editors, *Gene Regulatory Networks: Methods and Protocols*, volume 1883 of *Methods in Molecular Biology*, chapter 16, pages 385–422. Humana Press, 1 edition, 2019.

F. Fröhlich, D. Weindl, P. Stapor, Y. Schälte, L. Schmiester, L. Paszkowski, S. Merkt, and J. Hasenauer. Icb-dcm/amici: Amici 0.10.13 (version v0.10.13). Zenodo, October 2019. https://doi.org/10.5281/zenodo.3478595.

K. A. Fujita, Y. Toyoshima, S. Uda, Y.-i. Ozaki, H. Kubota, and S. Kuroda. Decoupling of receptor and downstream signals in the akt pathway by its low-pass filter characteristics. *Sci Signal*, 3(132):ra56, July 2010.

A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano. CellDesigner 3.5: A versatile modeling tool for biochemical networks. *Proceedings of the IEEE*, 96(8):1254–1265, August 2008.

S. Gaudet and K. Miller-Jensen. Redefining signaling pathways with an expanding single-cell toolbox. *Trends Biotechnol.*, 34(6):458–469, 2016.

N. Gazagnadou, R. M. Gower, and J. Salmon. Optimal mini-batch and step sizes for saga. *arXiv preprint, arXiv:1902.00071*, September 2019.

U. E. Gibson, C. A. Heid, and P. M. Williams. A novel method for real time quantitative RT-PCR. *Genome Res.*, 6:995–1001, 1996.

C. Giesen, H. A. O. Wang, D. Schapiro, N. Zivanovic, A. Jacobs, B. Hattendorf, P. J. Schüffler, D. Grolimund, J. M. Buhmann, S. Brandt, Z. Varga, P. J. Wild, D. Günther, and B. Bodenmiller. Highly multiplexed imaging of tumor tissues with subcellular resolution by mass cytometry. *Nat. Methods*, 11:417–422, Mar. 2014.

I. Gilitschenski and U. D. Hanebeck. Efficient deterministic dirac mixture approximation of gaussian distributions. In *American Control Conference*. IEEE, June 2013.

D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188(1): 404–425, September 1992.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J. R. Statist. Soc. B*, 73(2):123–214, March 2011.

D. Goldfarb. A family of variable-metric methods derived by variational means. *Math. Comp.*, 24(109):23–26, 1970.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

S. Gopalakrishnan, S. Dash, and C. Maranas. K-fit: An accelerated kinetic parameterization algorithm using steady-state fluxomic data. *Metabolic Eng.*, in press, March 2020.

R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtarik. Sgd: General analysis and improved rates. *arXiv preprint, arXiv:1901.09401*, 2019.

M. M. Graham and A. J. Storkey. Continuously tempered hamiltonian monte carlo. *arXiv preprint, arXiv:1704.03338*, 2017.

J. L. Gustafson. Reevaluating amdahl's law. *Commun. ACM*, 31(5):532–533, May 1988.

B. M. Gyori, J. A. Bachman, K. Subramanian, J. L. Muhlich, L. Galescu, and P. K. Sorger. From word models to executable models of signaling networks using automated assembly. *Mol. Syst. Biol.*, 13(11), 2017.

H. Haario, E. Saksman, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2): 223–242, 2001.

L. Haghverdi, M. Büttner, F. A. Wolf, F. Buettner, and F. J. Theis. Diffusion pseudotime robustly reconstructs lineage branching. *Nat. Methods*, 13:845–848, 2016.

B. C. Hall. *Lie Groups, Lie Algebras, and Representations - An elementary Introduction*, volume 222 of *Graduate Texts in Mathematics*. Springer International Publishing Switzerland, 2 edition, 2015.

U. D. Hanebeck and V. Klumpp. Localized cumulative distributions and a multivariate generalization of the cramer-von mises distance. In *International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, August 2008.

M. Hanke and O. Scherzer. Inverse problems light: Numerical differentiation. *Am. Math. Mon.*, 108(6):512–521, June 2001.

N. Hansen and A. Ostermaier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. *Evol. Comput.*, pages 312–317, 1996.

J. Hasenauer. *Modeling and parameter estimation for heterogeneous cell populations.* Logos Verlag, Berlin, February 2013.

J. Hasenauer, C. Hasenauer, T. Hucho, and F. J. Theis. ODE constrained mixture modelling: A method for unraveling subpopulation structures and dynamics. *PLoS Comput. Biol.*, 10(7): e1003686, July 2014.

J. Hasenauer, N. Jagiella, S. Hross, and F. J. Theis. Data-driven modelling of biological multi-scale processes. *Journal of Coupled Systems and Multiscale Dynamics*, 3(2):101–121, September 2015.

H. Hass, K. Masson, S. Wohlgemuth, V. Paragas, J. E. Allen, M. Sevecka, E. Pace, J. Timmer, J. Stelling, G. MacBeath, B. Schoeberl, and A. Raue. Predicting ligand-dependent tumors from multi-dimensional signaling features. *npj Syst. Biol. Appl.*, 3(1):27, 2017.

H. Hass, C. Loos, E. Raimúndez-Álvarez, J. Timmer, J. Hasenauer, and C. Kreutz. Benchmark problems for dynamic modeling of intracellular processes. *Bioinf.*, 35(17):3073–3082, 2019.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: data mining, inference, and prediction.* Springer Series in Statistics. Springer-Verlag, New York, 2 edition, 2005.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 51(1):97–109, April 1970.

C. R. Henderson, O. Kempthorne, S. R. Searle, and C. M. von Krosigk. The estimation of environmental and genetic trends from records subject to culling. *Biometrics*, 15(2):192–218, 1959.

D. Henriques, A. F. Villaverde, M. Rocha, J. Saez-Rodriguez, and J. R. Banga. Data-driven reverse engineering of signaling pathways using ensembles of dynamic models. *PLoS Comput. Biol.*, 13(2):e1005379, February 2017.

B. Hepp, A. Gupta, and M. Khammash. Adaptive hybrid simulations for multiscale stochastic reaction networks. *J. Chem. Phys.*, 142(3), 2015.

L. A. Herzenberg, J. Tung, W. A. Moore, L. A. Herzenberg, and D. R. Parks. Interpreting flow cytometry data: A guide for the perplexed. *Nat. Immunol.*, 7(7):681–685, July 2006.

S. C. Hicks, M. Teng, and R. A. Irizarry. On the widespread and critical impact of systematic bias and batch effects in single-cell RNA-seq data. *bioRxiv preprint, doi.org/10.1101/025528*, 2015.

A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM T. Math. Software.*, 31(3):363–396, September 2005.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.*, 117(4):500–544, August 1952.

M. D. Hoffman and A. Gelman. The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI – a COmplex PAthway SImulator. *Bioinf.*, 22(24):3067–3074, 2006.

M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinf.*, 19(4):524–531, March 2003.

S. Hug, A. Raue, J. Hasenauer, J. Bachmann, U. Klingmüller, J. Timmer, and F. J. Theis. High-dimensional Bayesian parameter estimation: Case study for a model of JAK2/STAT5 signaling. *Math. Biosci.*, 246(2):293–304, November 2013.

ICON Development Solutions. Nonmem 7.4. Software package, Proprietary License, 2020.

T. Ideker, T. Galitski, and L. Hood. A new approach to decoding life: systems biology. *Annu. Rev. Genomics Hum. Genet.*, 2(1):343–372, 2001.

A. Imle, P. Kumberger, N. D. Schnellbächer, J. Fehr, P. Carrillo-Bustamante, J. Ales, P. Schmidt, C. Ritter, W. J. Godinez, B. Müller, K. Rohr, F. A. Hamprecht, U. S. Schwarz, F. Graw, and O. T. Fackler. Experimental and computational analyses reveal that environmental restrictions shape hiv-1 spread in 3d cultures. *Nat. Commun.*, 10(2144), 2019.

E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14 (6):1569–1572, November 2003.

N. Jagiella, D. Rickert, F. J. Theis, and J. Hasenauer. Parallelization and high-performance computing enables automated statistical inference of multi-scale models. *Cell Syst.*, 4(2):194–206, February 2017.

A. Janowczyk and A. Madabhushi. Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. *J. Pathol. Inf.*, 7(29), 2016.

M. Joshi, A. Seidel-Morgenstern, and A. Kremling. Exploiting the bootstrap method for quantifying parameter confidence intervals in dynamical systems. *Metabolic Eng.*, 8:447–455, May 2006.

E.-M. Kapfer, P. Stapor, and J. Hasenauer. Challenges in the calibration of large-scale ordinary differential equation models. *IFAC-PapersOnLine*, 52(26):58–64, 2019.

M. Karlsson, D. L. I. Janzén, L. Durrieu, A. Colman-Lerner, M. C. Kjellsson, and G. Cedersund. Nonlinear mixed-effects modelling for single cell estimation: When, why, and how to use it. *BMC Syst. Biol.*, 9(1):52, September 2015.

J. R. Karr, J. C. Sanghvi, D. N. Macklin, M. V. Gutschow, J. M. Jacobs, B. Bolival Jr, N. Assad-Garcia, J. I. Glass, and M. W. Covert. A whole-cell computational model predicts phenotype from genotype. *Cell*, 150(2):389–401, July 2012.

D. Kaschek, W. Mader, M. Fehling-Kaschek, M. Rosenblatt, and J. Timmer. Dynamic modeling, parameter estimation, and uncertainty analysis in r. *J. Stat. Softw.*, 88(10), 2019.

J. D. Kearns and A. Hoffmann. Integrating computational and biochemical studies to explore mechanisms in nf-κb signaling. *J. Biol. Chem.*, 284(9):5439–5443, Dec. 2009.

J. Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

M. Khammash. Stochastic gene expression: modeling, analysis, and identification. In *Proc. of the 15th IFAC Symp. Syst. Ident.*, pages 1022–1028, Saint-Malo, France, 2009.

A. Khodayari and C. Maranas. A genome-scale escherichia coli kinetic metabolic model k-ecoli457 satisfying flux data for multiple mutant strains. *Nat. Commun.*, 7(13806), 2016.

B. N. Kholodenko. Untangling the signalling wires. *Nat. Cell Biol.*, 9(3):247–249, Mar. 2007.

D. P. Kingma and L. J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR) 2015 - accepted papers*. Ithaca, 2015.

S. Kirkpatrick, C. D. Gelatt Jr, and M. P. M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, November 2002.

E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach. *Systems biology in practice*. Wiley-VCH, Weinheim, 2005.

A. A. Kolodziejczyk, J. K. Kim, V. Svensson, J. C. Marioni, and S. A. Teichmann. The technology and biology of single-cell RNA sequencing. *Molecular cell*, 58(4):610–620, 2015.

A. Korkut, W. Wang, E. Demir, B. A. Aksoy, X. Jing, E. J. Molinelli, Ö. Babur, D. L. Bemis, S. O. Sumer, D. B. Solit, et al. Perturbation biology nominates upstream–downstream drug combinations in raf inhibitor resistant melanoma cells. *Elife*, 4:e04640, 2015.

A. Kramer, J. Hasenauer, F. Allgöwer, and N. Radde. Computation of the posterior entropy in a Bayesian framework for parameter estimation in biological networks. In *Proc. IEEE Multi-Conf. Syst. Contr.*, pages 493–498, Yokohama, Japan, September 2010.

C. Kreutz. New concepts for evaluating the performance of computational methods. *IFAC-PapersOnLine*, 49(26):63–70, 2016.

C. Kreutz. An easy and efficient approach for testing identifiability. *Bioinf.*, 34(11), 2018.

C. Kreutz. Guidelines for benchmarking of optimization-based approaches for fitting mathematical models. *Genome Biol.*, 20(281), December 2019.

C. Kreutz, A. Raue, and J. Timmer. Likelihood based observability analysis and confidence intervals for predictions of dynamic models. *BMC Syst. Biol.*, 6(120), September 2012.

C. Kreutz, A. Raue, D. Kaschek, and J. Timmer. Profile likelihood in systems biology. *FEBS J.*, 280(11):2564–2571, June 2013.

C. Kühn, C. Wierling, A. Kühn, E. Klipp, G. Panopoulou, H. Lehrach, and A. J. Poustka. Monte Carlo analysis of an ODE model of the sea urchin endomesoderm network. *BMC Syst Biol*, 3 (1):1, 2009.

E. Kuhn and M. Lavielle. Maximum likelihood estimation in nonlinear mixed effects models. *Comput. Stat. Data. Anal.*, 49(4):1020 – 1038, 2005.

M. K. Lacki and B. Miasojedow. State-dependent swap strategies and automatic reduction of number of temperatures in adaptive parallel tempering algorithm. *Stat. Comput.*, 26(5): 951–964, June 2015.

N. M. Laird and J. H. Ware. Random-effects models for longitudinal data. *Biometrics*, 38: 963–974, 1982.

S. Lan, T. Bui-Thanh, M. Christie, and M. Girolami. Emulation of higher-order tensors in manifold Monte Carlo methods for Bayesian inverse problems. *arXiv preprint, arXiv:1507.06244v2*, July 2015.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *P. IEEE*, 86(11):2278–2323, 1998.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, chapter Efficient BackProp, pages 9–50. Springer Berlin Heidelberg, 2002.

L. Lei and M. I. Jordan. On the adaptivity of stochastic gradient-based optimization. *arXiv preprint, arXiv:1904.04480v2*, April 2019.

J. Leis and M. Kramer. The simultaneous solution and sensitivity analysis of systems described by ordinary differential equations. *ACM T. Math. Software.*, 14:45–60, 1988.

C. Li, M. Donizelli, N. Rodriguez, H. Dharuri, L. Endler, V. Chelliah, L. Li, E. He, A. Henry, M. I. Stefan, J. L. Snoep, M. Hucka, N. Le Novère, and C. Laibe. BioModels database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Syst. Biol.*, 4:92, 2010.

J. Li, W. Zhao, R. Akbani, W. Liu, Z. Ju, S. Ling, C. P. Vellano, P. Roebuck, Q. Yu, A. K. Eterovic, L. A. Byers, M. A. Davies, W. Deng, Y. N. V. Gopal, G. Chen, E. M. von Euw, D. Slamon, D. Conklin, J. V. Heymach, A. F. Gazdar, J. D. Minna, J. N. Myers, Y. Lu, G. B. Mills, and H. Liang. Characterization of human cancer cell lines by reverse-phase protein arrays. *Cancer Cell*, 31(2):225–239, May 2017.

T. S. Ligon, F. Fröhlich, O. T. Chi, J. R. Banga, E. Balsa-Canto, and J. Hasenauer. Genssi 2.0: Multi-experiment structural identifiability analysis of sbml models. *Bioinf.*, 34(8):1421–1423, 2018.

D. Lill, J. Timmer, and D. Kaschek. Local riemannian geometry of model manifolds and its implications for practical parameter identifiability. *PLoS ONE*, 14(6), 2019.

M. Lindstrom and D. Bates. Nonlinear mixed effects models for repeated measures data, biometrics. *Biometrics*, September 1990.

G. Lines, L. Paszkowski, L. Schmiester, D. Weindl, P. Stapor, and J. Hasenauer. Efficient computation of steady states in large-scale ode models of biochemical reaction networks. *IFAC-PapersOnLine*, 52(26):32–37, 2019.

A. Llamosi, A. M. Gonzalez-Vargas, C. Versari, E. Cinquemani, G. Ferrari-Trecate, P. Hersen, and G. Batt. What population reveals about individual cell identity: Single-cell parameter estimation of models of gene expression in yeast. *PLoS Comput. Biol.*, 12(2):1–18, February 2016.

A. C. Lloyd. The regulation of cell size. *Cell*, 154(6):1194–1205, 2013.

C. Loos and J. Hasenauer. Mathematical modeling of variability in intracellular signaling. *Current Opinion in Systems Biology*, pages 17–24, 2019.

C. Loos and J. Hasenauer. Robust calibration of hierarchical population models for heterogeneous cell populations. *J. Theor. Biol.*, 488, March 2020.

C. Loos, S. Krause, and J. Hasenauer. Hierarchical optimization for the efficient parametrization of ODE models. *Bioinf.*, 34(24):4266–4273, July 2018a.

C. Loos, K. Moeller, F. Fröhlich, T. Hucho, and J. Hasenauer. A hierarchical, data-driven approach to modeling single-cell populations predicts latent causes of cell-to-cell variability. *Cell Syst.*, 6(5):593–603, 2018b.

P. Lucarelli, M. Schilling, C. Kreutz, A. Vlasov, M. E. Boehm, N. Iwamoto, B. Steiert, S. Lattermann, M. Wšch, M. Stepath, M. S. Matter, M. Heikenwälder, K. Hoffmann, D. Deharde, G. Damm, D. Seehofer, M. Muciek, W. D. Gretz, Norbert Lehmann, J. Timmer, and U. Klingmüller. Resolving the combinatorial complexity of smad protein complex formation and its link to gene expression. *Cell Syst.*, 6(1):75–89, 2018.

A. MacNamara, C. Terfve, D. Henriques, B. P. Bernabé, and J. Saez-Rodriguez. State–time spectrum of signal transduction logic models. *Phys. Biol.*, 9(4):045003, 2012.

M. Mahsereci, L. Balles, C. Lassner, and P. Hennig. Early stopping without a validation set. *arXiv preprint, arXiv:1703.09580*, April 2017.

C. Maier, C. Loos, and J. Hasenauer. Robust parameter estimation for dynamical systems from outlier-corrupted data. *Bioinf.*, 33(5):718–725, March 2017.

C. Maier, N. Hartung, J. de Wiljes, C. Kloft, and W. Huisinga. Bayesian data assimilation to support informed decision making in individualized chemotherapy. *CPT Pharmacometrics Syst. Pharmacol.*, 9(3), 2020.

T. Maiwald and J. Timmer. Dynamical modeling and multi-experiment fitting with potterswheel. *Bioinf.*, 24(18):2037–2043, July 2008.

T. Maiwald, H. Hass, B. Steiert, J. Vanlier, R. Engesser, A. Raue, F. Kipkeew, H. H. Bock, D. Kaschek, C. Kreutz, and J. Timmer. Driving the model to its limit: Profile likelihood based model reduction. *PLoS ONE*, 11(9), September 2016.

S. Mandt, M. D. Hoffman, and D. M. Blei. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint, arXiv:1704.04289v2*, January 2018.

J. Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning*, pages 735–742, 2010.

J. M. Martinez and M. Raydan. Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization. *J. Global. Optim.*, 68(2):367–385, 2017.

MathWorks. Matlab optimization toolbox (r2016a). Documentation available at https://de.mathworks.com/help/optim, 2016.

J. Melin, Z. P. Parra-Guillen, R. Michelet, T. Truong, W. Huisinga, N. Hartung, P. Hindmarsh, and C. Kloft. Pharmacokinetic/Pharmacodynamic evaluation of hydrocortisone therapy in pediatric patients with congenital adrenal hyperplasia. *J. Clin. Endocr. Metab.*, 105(4):e1729–e1740, April 2020.

P. Mendes, S. Hoops, S. Sahle, R. Gauges, J. Dada, and U. Kummer. *Computational Modeling of Biochemical Networks Using COPASI*, chapter 2. Part of the Methods in Molecular Biology. Humana Press, 2009.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

B. Miasojedow, E. Moulines, and M. Vihola. An adaptive parallel tempering algorithm. *J. Comput. Graph. Stat.*, 22(3):649–664, 2013.

M. Mitchell. *An introduction to genetic algorithms.* MIT press, 1998.

S. Mitchell and A. Hoffmann. Identifying noise sources governing cell-to-cell variability. *Current opinion in systems biology*, 8:39–45, 2018.

E. D. Mitra, R. Dias, R. G. Posner, and W. S. Hlavacek. Using both qualitative and quantitative data in parameter identification for systems biology models. *Nat. Commun.*, 9(1):3901, 2018.

U. Münzner, E. Klipp, and M. Krantz. A comprehensive, mechanistically detailed, and executable model of the cell division cycle in saccharomyces cerevisiae. *Nat. Commun.*, 10, 2019.

K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

S. G. Nash. Newton-type minimization via the Lanczos method. *SIAM J. Numer. Anal.*, 21(4): 770–788, 1984.

R. M. Neal. *Handbook of Markov Chain Monte Carlo*, chapter MCMC using Hamiltonian dynamics. Chapman & Hall / CRC Press, London, United Kingdom, 2011.

J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7(4): 308–313, 1965.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Y. Nesterov and B. Polyak. Cubic regularization of newton method and its global performance. *Math. Program.*, 108(1):177–205, August 2006.

J. Nocedal. Updating quasi-newton matrices with limited storage. *Math. Comput.*, 35(151): 773–782, 1980.

J. Nocedal and S. Wright. *Numerical Optimization.* Springer Science & Business Media, 2006.

B. G. Olivier and J. L. Snoep. Web-based kinetic modelling using JWS Online. *Bioinf.*, 20(13): 2143–4, Sep 2004.

D. B. Özyurt and P. I. Barton. Cheap second order directional derivatives of stiff ODE embedded functionals. *SIAM J. Sci. Comput.*, 26(5):1725–1743, 2005.

M. Peifer and J. Timmer. Parameter estimation in ordinary differential equations for biochemical processes using the method of multiple shooting. *IET Syst. Biol.*, 1(2):78–88, Mar 2007.

D. R. Penas, P. González, J. A. Egea, J. R. Banga, and R. Doallo. Parallel metaheuristics in computational biology: An asynchronous cooperative enhanced scatter search method. *Procedia Comput. Sci.*, 51:630–639, 2015.

S. P. Perfetto, P. K. Chattopadhyay, and M. Roederer. Seventeen-colour flow cytometry: unravelling the immune system. *Nat. Rev. Immunol.*, 4(8):648–55, 08 2004.

J. C. Pinheiro. *Topics in mixed effects models.* Ph.d. thesis, University of Wisconsin, Madison, Madison, USA, 1994.

J. C. Pinheiro and D. M. Bates. Unconstrained parameterizations for variance-covariance matrices. *Stat. Comput.*, 6(3):289–296, 1996.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Comp. Math. Math. Phys.*, 4(5):1–17, 1964.

M. J. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. Technical report, Cambridge NA Report NA2009/06, University of Cambridge, Cambridge, 2009.

A. Raue. *Quantitative Dynamic Modeling: Theory and Application to Signal Transduction in the Erythropoietic System.* Phd. thesis, Albert-Ludwigs-Universität Freiburg im Breisgau, 2013.

A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinf.*, 25(25):1923–1929, May 2009.

A. Raue, V. Becker, U. Klingmüller, and J. Timmer. Identifiability and observability analysis for experimental design in nonlinear dynamical models. *Chaos*, 20(045105), December 2010.

A. Raue, C. Kreutz, F. J. Theis, and J. Timmer. Joining forces of Bayesian and frequentist methodology: A study for inference in the presence of non-identifiability. *Philos. T. Roy. Soc. A*, 371(1984), January 2013a.

A. Raue, M. Schilling, J. Bachmann, A. Matteson, M. Schelke, D. Kaschek, S. Hug, C. Kreutz, B. D. Harms, F. J. Theis, U. Klingmüller, and J. Timmer. Lessons learned from quantitative dynamical modeling in systems biology. *PLoS ONE*, 8(9):e74335, Sept. 2013b.

A. Raue, B. Steiert, M. Schelker, C. Kreutz, T. Maiwald, H. Hass, J. Vanlier, C. Tönsing, L. Adlung, R. Engesser, W. Mader, T. Heinemann, J. Hasenauer, M. Schilling, T. Höfer, E. Klipp, F. J. Theis, U. Klingmüller, B. Schöberl, and J.Timmer. Data2Dynamics: a modeling environment tailored to parameter estimation in dynamical systems. *Bioinf.*, 31(21):3558–3560, November 2015.

J. Renart, J. Reiser, and G. R. Stark. Transfer of proteins from gels to diazobenzyloxymethyl-paper and detection with antisera: a method for studying antibody specificity and antigen structure. *Proc. Natl. Acad. Sci. USA*, 76(7):3116–3120, July 1979.

F. Rigat and A. Mira. Parallel hierarchical sampling: a general-purpose class of multiple-chains MCMC algorithms. *Comp. Stat. Data Anal.*, 56(6):1450–1467, June 2012.

H. Robbins and S. Monroe. A stoachstic approximation method. *Ann. Math. Stat.*, 22(3):400–407, 1951.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer, 2004.

N. Rosenfeld, J. W. Young, U. Alon, P. S. Swain, and M. B. Elowitz. Gene regulation at the single-cell level. *Science*, 307(5717):1962–1965, March 2005.

S. Ruder. An overview of gradient descent optimisation algorithms. *arXiv preprint, arXiv:1609.04747*, 2016.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

S. K. Sahu, D. K. Dey, and M. D. Branco. A new class of multivariate skew distributions with applications to Bayesian regression models. *Can. J. Stat.*, 31(2):129–150, 2003.

Y. Schälte, P. Stapor, and J. Hasenauer. Evaluation of derivative-free optimizers for parameter estimation in systems biology. *IFAC-PapersOnLine*, 51(19):98–101, 2018.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Math. Program. Ser. A*, 162:83–112, 2017.

L. Schmiester, Y. Schälte, F. Fröhlich, J. Hasenauer, and D. Weindl. Efficient parameterization of large-scale dynamic models based on relative measurements. *Bioinf.*, 36(2), July 2019a.

L. Schmiester, D. Weindl, and J. Hasenauer. Statistical inference of mechanistic models from qualitative data using an efficient optimal scaling approach. *bioRxiv preprint, doi.org/10.1101/848648*, November 2019b.

L. Schmiester, Y. Schälte, F. T. Bergmann, T. Camba, E. Dudkin, J. Egert, F. Fröhlich, L. Fuhrmann, A. L. Hauber, S. Kemmer, P. Lakrisenko, C. Loos, S. Merkt, D. Pathirana, E. Raimúndez-Álvarez, L. Refisch, M. Rosenblatt, P. L. Stapor, P. Städter, D. Wang, F.-G. Wieland, J. R. Banga, J. Timmer, A. F. Villaverde, S. Sahle, C. Kreutz, J. Hasenauer, and D. Weindl. Petab – interoperable specification of parameter estimation problems in systems biology. *arXiv preprint, arXiv:2004.01154*, 2020.

I. Schomburg, A. Chang, S. Placzek, C. Söhngen, M. Rother, and M. Lang. Brenda in 2013: integrated reactions, kinetic data, enzyme function data, improved disease classification: new options and contents in brenda. *Nucleic Acids Res.*, 41(D1):D764–D772, 2013.

D. Seita, X. Pan, H. Chen, and J. Canny. An efficient minibatch acceptance test for metropolis-hastings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*. International Joint Conferences on Artificial Intelligence, 2018.

B. Sengupta, K. J. Friston, and W. D. Penny. Efficient gradient computation for dynamical models. *NeuroImage*, 98:521–527, 2014.

R. Serban and A. C. Hindmarsh. CVODES: The sensitivity-enabled ODE solver in SUNDIALS. In *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 257–269. ASME, 2005.

L. F. Shampine and M. W. Reichelt. The MATLAB ODE Suite. *SIAM J. Sci. Comput.*, 18(1): 1–22, 1997.

L. B. Sheiner and S. L. Beal. Evaluation of methods for estimating population pharmacokinetic parameters. III. Monoexponential model: Routine clinical pharmacokinetic data. *J. Pharmacokinet. Biop.*, 11(3):303–319, June 1983.

K. Smallbone and P. Mendes. Large-scale metabolic models: From reconstruction to differential equations. *Ind. Biotechnol.*, 9(4):179–184, 2013.

L. Sommerlade, M. Thiel, M. Mader, W. Mader, J. Timmer, B. Platt, and B. Schelter. Assessing the strength of directed influences among neural signals: An approach to noisy data. *J. Neurosci.*, 239:47–64, 2015.

E. T. Somogyi, J.-M. Bouteiller, J. A. Glazier, M. König, J. K. Medley, M. H. Swat, and H. M. Sauro. libRoadRunner: A high performance SBML simulation and analysis library. *Bioinf.*, 31(20):3315–3321, 2015.

D. C. Sorensen. Newton's method with a model trust region modification. *SIAM J. Numer. Anal.*, 19(2):409–426, 1982.

S. L. Spencer and P. K. Sorger. Measuring and modeling apoptosis in single cells. *Cell*, 144(6): 926–939, March 2011.

S. L. Spencer, S. Gaudet, J. G. Albeck, J. M. Burke, and P. K. Sorger. Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. *Nature*, 459(7245):428–433, May 2009.

P. Stapor, F. Fröhlich, and J. Hasenauer. Optimization and profile calculation of ODE models using second order adjoint sensitivity analysis. *Bioinf.*, 34(13):i151–i159, 2018a.

P. Stapor, D. Weindl, B. Ballnus, S. Hug, C. Loos, A. Fiedler, S. Krause, S. Hross, F. Fröhlich, and J. Hasenauer. PESTO: Parameter EStimation TOolbox. *Bioinf.*, 34(4):705–707, February 2018b.

P. Stapor, L. Schmiester, C. Wierling, B. M. H. Lange, D. Weindl, and J. Hasenauer. Mini-batch optimization enables training of ode models on large-scale datasets. *bioRxiv preprint, doi.org/10.1101/859884*, 2019.

J. Starruß, W. de Back, L. Brusch, and A. Deutsch. Morpheus: A user-friendly modeling environment for multiscale and multicellular systems biology. *Bioinf.*, 30(9):1331–1332, January 2014.

I. Sutskever. *Training recurrent neural networks*. PhD thesis, University of Toronto, Department of Computer Science, 2013.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proc. Int. Conf. Machine Learning*, pages 1139–1147, 2013.

P. S. Swain, M. B. Elowitz, and E. D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proc. Natl. Acad. Sci. USA*, 99(20):12795–12800, October 2002.

I. Swameye, T. G. Müller, J. Timmer, O. Sandra, and U. Klingmüller. Identification of nucleocytoplasmic cycling as a remote sensor in cellular signaling by databased modeling. *Proc. Natl. Acad. Sci. USA*, 100(3):1028–1033, February 2003.

M. J. Swat, S. Moodie, S. M. Wimalaratne, N. R. Kristensen, M. Lavielle, A. Mari, P. Magni, M. K. Smith, R. Bizzotto, L. Pasotti, E. Mezzalana, E. Comets, C. Sarr, N. Terranova, E. Blaudez, P. Chan, J. Chard, K. Chatel, M. Chenel, D. Edwards, C. Franklin, T. Giorgino, M. Glont, P. Girard, P. Grenon, K. Harling, A. C. Hooker, R. Kaye, R. Keizer, C. Kloft, J. N. Kok, N. Kokash, C. Laibe, C. Laveille, G. Lestini, F. Mentre, A. Munafo, R. Nordgren, H. B. Nyberg, Z. P. Parra-Guillen, E. Plan, B. Ribba, G. Smith, I. F. Trocóniz, F. Yvon, P. A. Milligan, L. Harnisch, M. Karlsson, H. Hermjakob, and N. Le Novére. Pharmacometrics markup language (PharmML): Opening new perspectives for model exchange in drug development. *CPT Pharmacometrics Syst Pharmacol*, 2015.

S. Tay, J. J. Hughey, T. K. Lee, T. Lipniacki, S. R. Quake, and M. W. Covert. Single-cell NF-κB dynamics reveal digital activation and analogue information processing. *Nature*, 466:267–271, July 2010.

R. A. Thisted. *Elements of Statistical Computing*. Chapman & Hall / CRC Press, 1988.

T. Tieleman and G. Hinton. Lecture 6.5 – rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

T. Toni and B. Tidor. Combined model of intrinsic and extrinsic variability for computational network design with application to synthetic biology. *PLoS Comput. Biol.*, 9(3):e1002960, March 2013.

C. W. Tornoe, H. Agerso, E. N. Jonsson, H. Madsen, and H. A. Nielsen. Non-linear mixed-effects pharmacokinetic/pharmacodynamic modelling in nlme using differential equations. *Comput. Meth. Prog. Bio.*, 76(1):31–40, October 2004.

R. Tsien. The green fluorescent protein. *Annu. Rev. Biochem.*, 67:509–544, 1998.

R. R. Vallabhajosyula, V. Chickarmane, and H. M. Sauro. Conservation analysis of large biochemical networks. *Bioinf.*, 22(3):346–353, February 2006.

R. van der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. Ph.d. thesis, Oregon Health & Science University, April 2004.

V. S. Vassiliadis, E. B. Canto, and J. R. Banga. Second-order sensitivities of general dynamic systems with application to optimal control problems. *Chem. Eng. Sci.*, 54(17):3851–3860, 1999.

A. Vaz and L. Vicente. A particle swarm pattern search method for bound constrained global optimization. *J. Global Optim.*, 39(2):197–219, 2007.

A. F. Villaverde, F. Froehlich, D. Weindl, J. Hasenauer, and J. R. Banga. Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinf.*, 35(5):830–838, March 2018.

A. F. Villaverde, E. Raimúndez-Álvarez, J. Hasenauer, and J. R. Banga. A comparison of methods for quantifying prediction uncertainty in systems biology. *IFAC-PapersOnLine*, 52 (26):45–51, 2019.

W. Vousden, W. M. Farr, and I. Mandel. Dynamic temperature selection for parallel tempering in Markov chain Monte Carlo simulations. *Mon. Not. R. Astron. Soc.*, 455(2):1919–1937, 2016.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, 2006.

D. Wang, P. Stapor, and J. Hasenauer. Dirac mixture distributions for the approximation of mixed effects models. *IFAC-PapersOnLine*, 52(26):200–206, 2019.

X. Wang, S. Ma, D. Goldfarb, and W. Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM J. Optim.*, 27(2):927–956, 2017.

P. Weber, J. Hasenauer, F. Allgöwer, and N. Radde. Parameter estimation and identifiability of biological networks using relative data. In S. Bittanti, A. Cenedese, and S. Zampieri, editors, *Proc. of the 18th IFAC World Congress*, volume 18, pages 11648–11653, Milano, Italy, August 2011.

A. Y. Weiße and W. Huisinga. Error-controlled global sensitivity analysis of ordinary differential equations. *J. Comput. Phys.*, 230:6824–6842, 2011.

A. Y. Weiße, R. H. Middleton, and W. Huisinga. Quantifying uncertainty, variability and likelihood for ordinary differential equation models. *BMC Syst. Biol.*, 4(144), Nov. 2010.

D. R. Wilson and T. R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16:1429–1451, April 2003.

U. Wittig, R. Kania, M. Golebiewski, M. Rey, L. Shi, L. Jong, E. Algaa, A. Weidemann, H. Sauer-Danzwith, S. Mir, O. Krebs, M. Bittkowski, E. Wetsch, I. Rojas, and W. Müller. SABIO-RK–database for biochemical reaction kinetics. *Nucleic Acids Res.*, 40(D1):D790–D796, 2012.

W. Yang, J. Soares, P. Greninger, E. J. Edelman, H. Lightfoot, S. Forbes, N. Bindal, D. Beare, J. A. Smith, I. R. Thompson, S. Ramaswamy, P. A. Futreal, D. A. Haber, M. R. Stratton, C. Benes, U. McDermott, and M. J. Garnett. Genomics of Drug Sensitivity in Cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells. *Nucleic Acids Res.*, 41:D955–D961, January 2013.

J. S. Yu and N. Bagheri. Agent-based models predict emergent behavior of heterogeneous cell populations in dynamic microenvironments. *Front. Bioeng. Biotechnol.*, 2020.

B. Yuan, C. Shen, A. Luna, A. Korkut, D. S. Marks, J. Ingraham, and C. Sander. Interpretable machine learning for perturbation biology. *bioRxiv preprint, doi.org/10.1101/746842*, August 2019.

C. Zechner, J. Ruess, P. Krenn, S. Pelet, M. Peter, J. Lygeros, and H. Koeppl. Moment-based inference predicts bimodality in transient gene expression. *Proc. Natl. Acad. Sci. USA*, 109 (21):8340–8345, May 2012.

Y. Zheng, S. M. M. Sweet, R. Popovic, E. Martinez-Garcia, J. D. Tipton, P. M. Thomas, J. D. Licht, and N. L. Kelleher. Total kinetic analysis reveals how combinatorial methylation patterns are established on lysines 27 and 36 of histone H3. *Proc. Natl. Acad. Sci. USA*, 109(34): 13549–13554, August 2012.