# Fides: Reliable Trust-Region Optimization for Parameter Estimation of Ordinary Differential Equation Models

Fabian Fröhlich [1,*], Peter K. Sorger [1,*]

May 2021

[1]Department of Systems Biology, Harvard Medical School, Boston, MA 02115, USA,
[*]To whom correspondence should be addressed.

**Abstract**

**Motivation:** Because they effectively represent mass action kinetics, ordinary differential equation models are widely used to describe biochemical processes. Optimization-based calibration of these models on experimental data can be challenging, even for low-dimensional problems. However, reliable model calibration is a prerequisite for many subsequent analysis steps, including uncertainty analysis, model selection and biological interpretation. Although multiple hypothesis have been advanced to explain why optimization based calibration of biochemical models is challenging, there are few comprehensive studies that test these hypothesis and tools for performing such studies are also lacking.

**Results:** We implemented an established trust-region method as a modular python framework (fides) to enable structured comparison of different approaches to ODE model calibration involving Hessian approximation schemes and trust-region subproblem solvers. We evaluate fides on a set of benchmark problems that include experimental data. We find a high variability in optimizer performance among different implementations of the same algorithm, with fides performing more reliably that other implementations investigated. Our investigation of possible sources of poor optimizer performance identify shortcomings in the widely used Gauss-Newton approximation. We address these shortcomings by proposing a novel hybrid Hessian approximation scheme that enhances optimizer performance.

**Availability:** Fides is published under the permissive BSD-3-Clause license with source code publicly available at `https://github.com/fides-dev/fides`. Citeable releases are archived on Zenodo.

**Contact:** fabian_froehlich@hms.harvard.edu and peter_sorger@hms.harvard.edu

**Supplementary information:** Supplementary data are available at *Bioinformatics* online and at `https://github.com/fides-dev/fides-benchmark`.

## 1 Introduction

Mass action biochemical systems can be accurately described in the continuous (large number of molecules) approximation by ordinary differential equation (ODE) models. ODE modeling applies remarkably well to biochemical processes in animal as well as human cells (Klipp et al., 2005). To ensure these models can recapitulate and predict experimental observations, model parameters must be inferred from data. This inference problem can be formulated as optimization problem, where the objective function describes the discrepancy between a solution to the ODE and and experimental data. Finding and adequate solution to this optimization problem can be computationally demanding, since evaluation of the objective function and it's derivatives requires numerical integration (Fröhlich et al., 2019). Model analysis such as uncertainty quantification via profile likelihood or sampling is typically initialized at the solution to the this optimization problem (Ballnus et al., 2018; Raue et al., 2009). Moreover, model selection using established measure such as AIC, BIC or other methods for penalizing complexity (Loos et al., 2018; Steiert et al., 2016) depend on optimization results. Thus, reliably finding adequate solutions to the optimization problem is of upmost importance for many aspects of model analysis.

In general, the optimization problem for ODE models is non-convex, resulting in few theoretical convergence guarantees for numerical optimizations. Researchers must therefore rely on empirical evidence to select appropriate optimization algorithms (Fröhlich et al., 2019). For a broad set of biochemical problems, Trust-Region methods initialized from a large number of random initial parameter values have performed well (Hass et al., 2019; Raue et al., 2013). Trust-Region methods use local (quadratic) approximations of the objective function to propose parameter update and iteratively refine their trust-region, i.e., the local neighborhood in which the local approximation can expected to adequately recapitulate the shape of the true objective function (Nocedal and Wright, 2006). Popular implementations of trust region methods are available in the MATLAB optimization toolbox or the scipy optimization module. However, for a considerable subset of problems, including low dimensional problems with as few as  20 parameters, these optimizers do not consistently converge to the same final values for the objective function (Hass et al., 2019), strongly suggesting that the global optimum - and perhaps not even a local optimum - has been reached. Specifically, for the problem based on work of Fujita et al. (2010), the difference in negative log-likelihood between the best and second best parameter values exceeds 10, the statistical threshold for model rejection according to criteria such as AIC or BIC, and could thus erroneously lead to rejection of the model . Thus, the presence of inconsistent final objective function values effectively precludes accurate investigation of the model.

Inconsistent final objective function values (Fig 1A) can either indicate that optimization has converged on one of several critical points (local minima, saddle points) (Fig 1B right) or that optimization was terminated before convergence was achieved (Fig 1B left). Most problems of interest in cell biology feature multiple local minima, which can be a result of curvature of the model manifold (Transtrum et al., 2011). Thus, when repeated local optimization runs consistently converge to a small set of similar objective function values the model is may be non-identifiable, but there is not necessarily a problem with optimization itself. However, when no consistency is achieved despite a large number of runs, it is unclear whether optimization did not converge or the objective function is "rugged" (Fröhlich et al., 2019) with many local minima. Non-convergent optimization can be consequence of model simulation, in which lax integration tolerances result in inaccurate numerical evaluation of the objective function value and its gradient (Tönsing et al., 2019). Inaccurate gradients often result in poor parameter update proposals, slowing the search in parameter space, and

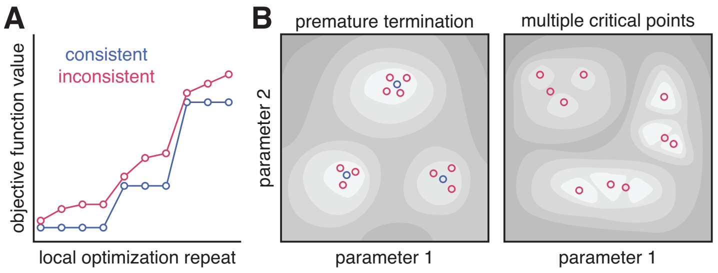

Figure 1: Illustration of final objective function values consistency and possible objective function landscapes. **A:** Waterfall plot with examples of consistent (blue) and inconsistent (red) final objective function values. **B:** Possible objective function landscapes that could explain the waterfall plots in A.

inaccurate objective function values may result in incorrect rejection of parameter updates, both of which can erroneously suggest convergence of optimization and thus lead to premature termination of optimization. Tönsing et al. (2019) found that prematurely terminated runs were often located in the neighborhood of local minima, corroborating the hypothesis of premature termination and suggested that a nudged elastic band method might by effective in improving optimization. Premature termination can also arise from poor performance of the optimization method. Dauphin et al. (2014) found that saddle points are prevalent in the objective function of neural network models and that optimization methods that do not account for directions of negative curvature may perform poorly in the vicinity of saddle points. However, neither the prevalence of saddle points, nor their impact on premature optimizer termination have been investigated in the case of biochemical ODE models. Similarly, Transtrum et al. (2011) have suggested that the use of Gauss-Newton Hessian approximations may not work well for sloppy problems, i.e., when the objective function Hessian has a broad eigenvalue spectrum, suggesting non-identifiability of parameters and ill-conditioning of the optimization problem. Sloppiness is believed to be a universal property of biochemical models (Gutenkunst et al., 2007). However, the method proposed by Transtrum et al. (2011) to address the limitations of Gauss-Newton Hessian approximation has not seen wide adoption, likely due to high computational cost and the complexity of current implementations. Overall, the findings described above show that early optimizer termination is a recurrent problem and may have a variety of causes, but a comprehensive evaluation of these causes as well as development and testing of methods to identify or resolve these issues are missing. In principle, this could be resolved by adapting optimization algorithms. For example, issues with the Gauss-Newton Hessian approximation could be resolved by using alternative approximation schemes such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). Issues with Saddle points could be resolved by employing symmetric rank-one (SR1) (Conn et al., 1991) approximations that account for negative curvature directions. However, many optimization algorithms were written decades ago and are difficult to customize or extend.

To address these and other challenges, this paper re-implements a trust-region algorithm in python and uses it to investigate several hypotheses about causes and potential solutions for poor optimizer performance. We find that use of an inaccurate Hessian approximation

Table 1: Feature overview for different trust-region optimization implementations. The non least-squares column indicates whether the method is applicable to non least-squares problems. The free column indicates whether the implementation is freely available or proprietary software.

| **Optimizer** | Subspace | Non least-squares | BFGS/ SR1 | Programming Language | free |
|---|---|---|---|---|---|
| `lsqnonlin` | $\mathcal{S}_{2\text{D}}$ | ☐ | ☐ | MATLAB | ☐ |
| `fmincon` | $\mathcal{S}_{2\text{D}}$ | ✓ | ☐ | MATLAB | ☐ |
| `ls_trf` | $\mathbb{R}^{n_\theta}$, $\mathcal{S}_{2\text{D}}$ | ☐ | ☐ | python | ✓ |
| `fides` | $\mathbb{R}^{n_\theta}$, $\mathcal{S}_{2\text{D}}$ | ✓ | ✓ | python | ✓ |

is one important contributor to poor optimization performance and propose a novel hybrid Hessian approximation scheme and demonstrate that this approach outperforms existing approaches on a set of benchmark problems.

## 2 Materials and Methods

For the purpose of this study, we considered four different optimizers that all implement the interior-trust-region algorithm proposed by Coleman and Li (1992): `fmincon`, referring to the MATLAB function of the same name and with `trust-region-reflective` as algorithm and `ldl-factorize` as subproblem algorithm, `lsqnonlin`, referring to the MATLAB function of the same name, `ls_trf`, referring to the scipy function `least_squares` with `trf` algorithm, and `fides`, the novel implementation we provide with this manuscript. Below we describe algorithmic details as well as the benchmark problems we used to evaluate these algorithms.

### 2.1 Model Formulation

An ODE model describes the temporal evolution of abundances of $n_x$ different molecular species $x_i$. The temporal evolution of $\mathbf{x}$ is determined by the vector field $f$ and the initial condition $\mathbf{x}_0$:

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, \boldsymbol{\theta}), \quad \mathbf{x}(t_0) = \mathbf{x}_0(\boldsymbol{\theta}). \tag{1}$$

Both may depend on the unknown parameters $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^{n_\theta}$ such as catalytic rates or binding affinities. Restricting optimization to the parameter domain $\Theta$ can constrain the search space for parameter values to biologically reasonable values and prevent numerical integration failures. For most problems, $\Theta$ is the tensor product of scalar search domains $(l_i, u_i)$ with $l_i < u_i$ and $l_i, u_i \in \mathbb{R} \cup \{-\infty, \infty\}$ for every parameter $\theta_i$

Experiments usually provide information about observables $\mathbf{y}$ which depend abundances $\mathbf{x}$ and parameters $\boldsymbol{\theta}$. A direct measurement of $\mathbf{x}$ is usually not possible. The dependence of the observable on abundances and parameters is described by

$$\mathbf{y}(t, \boldsymbol{\theta}) = h(\mathbf{x}(t, \boldsymbol{\theta}), \boldsymbol{\theta}). \tag{2}$$

## 2.2 Optimization Problem

To build predictive models, the model parameters $\boldsymbol{\theta}$ have to be inferred from experimental data. Experimental data are subject to measurement noise. A common assumption is that the measurement noise for $n_t$ time-points $t_j$ and $n_y$ observables $y_i$ is additive and independent, normally distributed for all time-points:

$$\bar{y}_{ij} = y_i(t_j, \boldsymbol{\theta}) + \epsilon_{ij}, \quad \epsilon_{ij} \overset{id}{\sim} \mathcal{N}(0, \sigma_{ij}^2(\boldsymbol{\theta})). \tag{3}$$

The model can be inferred from experimental data by maximizing the likelihood, which yields the maximum likelihood estimate (MLE). However, the evaluation of the likelihood function, involves the computation of several products, which can be numerically unstable. Thus, the negative log-likelihood

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n_y} \sum_{j=1}^{n_t} \log\left(2\pi\sigma_{ij}^2(\boldsymbol{\theta})\right) + \left(\frac{\bar{y}_{ij} - y_i(t_j, \boldsymbol{\theta})}{\sigma_{ij}(\boldsymbol{\theta})}\right)^2 \tag{4}$$

is typically used as objective function for minimization. As the logarithm is a strictly monotonously increasing function, the minimization of $J(\boldsymbol{\theta})$ is equivalent to the maximization of the likelihood. Therefore, the corresponding minimization problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} J(\boldsymbol{\theta}), \tag{5}$$

will infer the MLE parameters. If the noise variance $\sigma_{ij}^2$ does not depend on the parameters $\boldsymbol{\theta}$, (4) is a weighted least-squares objective function. As we discuss later, this least-squares structure can be exploited by using several optimization methods. Optimizers that do not require least-squares structure can also work with other noise models.

For the MATLAB optimizers `fmincon` and `lsqnonlin`, the objective function and it's derivatives were evaluated using data2dynamics (Raue et al., 2015) (commit `b1e6acd`), which was also by Hass et al. (2019). For the python optimizers `ls_trf` and fides, objective function and it's derivates were evaluated using AMICI (Fröhlich et al., 2021) 0.11.16 and pyPESTO 0.2.5.

## 2.3 Trust-Region Optimization

Trust-region methods minimize the objective function $J$ by iteratively updating parameter values $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}_k$ according to the local minimum $\Delta\boldsymbol{\theta}_k = p^* = \operatorname{argmin}_p m_k(p)$ of an approximation $m_k$ to the objective function. In many applications, a local, quadratic approximation is used, which that means the trust-region subproblem

$$p^* = \operatorname*{argmin}_{p\in\mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \qquad s.t.\ ||p|| \leq \Delta_k \tag{6}$$

is solved in every step $k$, where $f_k$ is the value, $g_k$ is the gradient and $B_k$ is the Hessian of the objective function, all evaluated at $\boldsymbol{\theta}_k$. $\Delta_k$ is the trust-region radius that restricts the norm of parameter updates. $\Delta_k$ is updated in every iteration depending on the agreement

Table 2: Overview of properties of different Hessian approximation schemes. BFGS is the Broyden-Fletcher-Goldfarb-Shannon algorithm. SR1 is the symmetric rank-1 update. GN is the Gauss-Newton approximation. The construction column indicates whether pointwise evaluation is possible or whether iterative construction is necessary. The indefinite column indicates whether the approximation can yield indefinite approximations. The convergence column indicates whether convergence to the true Hessian is generally possible.

| **Scheme** | Construction | Indefinite | Convergence |
|---|---|---|---|
| **BFGS** | iterative | ☐ | ✓ |
| **SR1** | iterative | ✓ | ✓ |
| **GN** | pointwise | ☐ | ☐ |

between the predicted decrease and actual decrease in objective function value (Nocedal and Wright, 2006). While $g_k$ can be efficiently and accurately computed using forward or adjoint sensitivity analysis (Fröhlich et al., 2017), it is computationally more demanding to compute $B_k$ (Stapor et al., 2018).

## 2.4 Hessian Approximation

Due to the high computational cost of exact Hessians $B_k$, many calibration tools resort to approximating the Hessian using a the Gauss-Newton (GN) method. The GN approximation $B^{(GN)}$ is based on a linearization of residuals $r_{ij}$

$$\mathbf{r}_{ij}(\boldsymbol{\theta}) = \frac{\bar{y}_{ij} - y_i(t_j, \boldsymbol{\theta})}{\sigma_{ij}(\boldsymbol{\theta})} \qquad B^{(GN)}_{kl} = \frac{1}{2}\sum_{i=1}^{n_y}\sum_{j=1}^{T} \frac{\partial r_{ij}}{\partial \theta_k}\frac{\partial r_{ij}}{\partial \theta_l}, \tag{7}$$

which yield a symmetric and positive semi-definite approximation, i.e., does not account for negative curvature. At the maximum likelihood estimate, the $B^{(GN)}$ is equal to the negative empirical Fisher Information Matrix, if $\sigma_{ij}$ does not depend on parameters $\boldsymbol{\theta}$. For parameter dependent $\sigma$, the $\log(\sigma)$ term in (4) cannot be assumed to be constant, which results in a non least-squares optimization problem. For non least-squares problems, the adequateness and formulation of the GN approximation is not well established. Raue (2013) proposes to transform the problem into least-squares form by introducing additional error residuals $\mathbf{r}^e_{ij}$ and adding a corresponding correction to the Gauss-Newton approximation $B^{(GN)}$ from (7), yielding $B^{(GNe)}$:

$$\begin{aligned} \mathbf{r}^e_{ij}(\boldsymbol{\theta}) &= \sqrt{2\log(\sigma_{ij}(\boldsymbol{\theta})) + C} \\ B^{(GNe)}_{kl} &= B^{(GN)}_{kl} + \frac{\frac{\partial \sigma_{ij}}{\partial \theta_l}\frac{\partial \sigma_{ij}}{\partial \theta_k}}{\sigma_{ij}(\boldsymbol{\theta})^2(2\log(\sigma_{ij}(\boldsymbol{\theta})) + C)}, \end{aligned} \tag{8}$$

where $C$ is some arbitrary, but sufficiently large constant that ensures $2\log(\sigma_{ij}(\boldsymbol{\theta})) + C > 0$. This condition ensures that residuals are real-valued and the approximation $B^{(GNe)}$ is positive semi-definite. Adding the constant $C$ to residuals adds a constant to the objective function value and, thus, neither influences its gradient and Hessian nor the location of minima.

However, $C$ does enter the GNe approximation, with unclear implications. In contrast, Stapor et al. (2018) suggest to ignore the second order derivative of the $\log(\sigma)$ term in (4), which corresponds to the limit $\lim_{C\to\infty} B^{(GNe)} = B^{(GN)}$.

As alternative to the GN approximation, BFGS or SR1 approximation schemes permit iterative construction of an approximate Hessian. The BFGS approximation guarantees a positive semi-definite approximation as long as a curvature condition is satisfied (Nocedal and Wright, 2006). Thus, it is usually only applied with line search methods that guarantee satisfaction of the curvature condition by selecting the step length according to (strong) Wolfe conditions (Nocedal and Wright, 2006). However, BFGS can also be used in trust-region methods by rejecting updates when the curvature condition is not satisfied, at the cost of losing some theoretical convergence guarantees (Nocedal and Wright, 2006). The SR1 approximation scheme can also yield indefinite approximations, incorporating negative curvature information, and has no step requirements.

`fmincon` and `lsqnonlin` were only evaluated using the GNe approximation as implemented in data2dynamics. `ls_trf` only implements the GN approximation. Fides was evaluated using BFGS and SR1 using respective native implementations as well as GN and GNe as implemented in AMICI. We used the default value of $C = 50$ for the computation of GNe in both data2dynamics and AMICI.

## 2.5 Subproblem Solution

More (1978) proposes an approach to solve (6) via eigenvalue decomposition of $B_k$. However, Byrd et al. (1988) note the high computational cost of this approach and suggest an approximate solution by solving the trust-region problem over a two dimensional subspace $\mathcal{S}_{2\mathrm{D}}$, spanned by gradient $g_k$ and Newton search directions $B_k^{-1}g_k$, instead of $\mathbb{R}^{n_\theta}$. Specifically for objective functions requiring numerical integration of ODE models, the cost of eigenvalue decomposition is generally negligible for problems involving fewer than $10^3$ parameters.

A crucial issue for the two-dimensional subspace approach are problems with indefinite (approximate) Hessians. For an indefinite $B_k$, the Newton search direction may not be a descent direction. This can be addressed through dampening of $B_k$ (Nocedal and Wright, 2006). For boundary constrained problems additional considerations are necessary (Coleman and Li, 1992) and require the identification of a direction of strong negative curvature. In fides, this negative curvature direction is computed using the the Lanczos method (Gould et al., 1999) implemented in `scipy.sparse.linalg.eigs`.

`fmincon` and `lsqnonlin` implement optimization only over $\mathcal{S}_{2\mathrm{D}}$, where the Newton search direction is computed using preconditioned direct factorization. Fides and `ls_trf` implement optimization over $\mathcal{S}_{2\mathrm{D}}$ (denoted by 2D in text and figures) and $\mathbb{R}^{n_\theta}$ (denoted by ND in text and figures). For `ls_trf`, we specified `tr_solver="lsmr"` for optimization over $\mathcal{S}_{2\mathrm{D}}$ and `tr_solver="exact"` for optimization over $\mathbb{R}^{n_\theta}$. `ls_trf` and fides both use (regularized) least-squares solvers to compute the Newton search direction, which is equivalent to using the Moore-Penrose pseudoinverse.

## 2.6 Handling of Boundary Constraints

The trust-region region subproblem (6) does not account for boundary constraints, which means that respectively $\boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}_k$ may not satisfy these constraints. For this purpose Coleman and Li (1992) introduced a rescaling of the opimization variables depending on how close current values are to parameter boundaries. Moreover, they propose a stepback strategy, where solutions to (6) are reflected at the parameter boundary. To ensure convergence, $\Delta\boldsymbol{\theta}_k$ is then selected based on the lowest $m_k(p)$ value along the reflection of $p^*$ at the parameter boundary and the cauchy point, which is the minimizer of (6) along the gradient (Nocedal and Wright, 2006). `fmincon`, `lsqnonlin` and `ls_trf` all implement rescaling, but only allow for a single reflection at the boundary. Fides implements the rescaling and allows for a single or arbitrarily many reflections until the first local minimum is encountered. As the reflection of $p^*$ at the parameter boundary defines a one-dimensional search space, the local minimum can be computed analytically at minimal computational cost.

## 2.7 Optimizer Performance Evaluation

To evaluate optimization performance, Hass et al. (2019) counted the number $n_{success}$ of optimization runs that reached a final objective function value sufficiently close to the "best value" and divided that by the time to complete all optimization runs $t_{total}$. The best objective function value was defined as the lowest value reach by any method we investigated. This implements the performance metric introduced by Villaverde et al. (2019) except that we replaced $t_{total}$ with the number of gradient evaluations $n_{grad}$. For the calculation of $n_{success}$ we used a threshold of 0.05, which is equivalent to the threshold 0.1 used in (Hass et al., 2019) since the objective function we used was scaled by a factor of 2. The performance metric $\phi = n_{success}/n_{grad}$ ignores differences in computation time for gradients at different parameter values and prevents computer or simulator performance from influencing results. This provides a fairer evaluation of the algorithm or method itself and is particularly relevant when optimization is performed on computing clusters having heterogeneous nodes. For all trust-region optimizers we employed, the number of gradient evaluations was equal to the number of iterations.

## 2.8 Implementation

Fides is implemented as modular, object-oriented python code. Specifically, the subproblem, subproblem solvers, stepback strategies and Hessian approximations have class-based implementations, making it easy to extend our methods or add new methods. Internally, fides uses scipy and numpy libraries to store vectors and matrices and perform linear algebra operations. To ensure accessibility to state of the art simulation and sensitivity analysis, we implemented an interface to fides in pyPESTO, which uses AMICI (Fröhlich et al., 2021) to perform simulation and sensitivty computation via CVODES (Hindmarsh et al., 2005). Moreover, this enables import of biological parameter estimation problems specified in the PEtab (Schmiester et al., 2021) format.

Table 3: Summary of problem characteristics for benchmark examples

| Problem | $n_\theta$ | $n_x$ | $n_y * n_t$ | sloppy |
|---|---|---|---|---|
| ***Fujita*** | 19 | 9 | 144 | ✓ |
| ***Boehm*** | 9 | 8 | 48 | ☐ |
| ***Fiedler*** | 19 | 6 | 72 | ✓ |
| ***Crauste*** | 12 | 5 | 21 | ✓ |
| ***Brannmark*** | 22 | 9 | 43 | ✓ |
| ***Weber*** | 36 | 7 | 135 | ✓ |
| ***Zheng*** | 46 | 15 | 60 | ✓ |

## 2.9 Benchmark Problems

To evaluate the performance of different optimizers, we considered a total of 7 different benchmark problems (Table 2.9) introduced by Hass et al. (2019) that were reformulated in the PEtab (Schmiester et al., 2021) format. The selection of benchmark problems was based on whether all functionality required for the import of respective PEtab problems was supported and validated in AMICI and whether poor optimization performance was observed in the original benchmark. All benchmarks problems were previously published and included experimental data for model calibration. In the following sections we highlight three of these benchmark problems and describe specific problem characteristics.

**Boehm Benchmark Characteristics:** The model by Boehm et al. (2014) describes homo- and heterodimerization of STAT5A and STAT5B transcription factor isoforms, which is required to form a functional complex. The model includes 8 biochemical species, 3 observables and 9 parameters. The calibration data includes 1 experimental conditions with 48 datapoints. The benchmark by Hass et al. (2019) found that the Hessian of the objective function of this problem has a narrow eigenvalue spectrum, indicating a non-sloppy model and the performance of optimizers we investigated was good insofar as the best objective function value was repeatedly reached. Nonetheless, only a small fraction of optimizer runs converged to the best identified local minimum.

**Fujita Benchmark Characteristics:** The model by Fujita et al. (2010) describes the activation of the Akt signaling pathway in PC12 cells in response to epidermal growth factor stimulation. The model includes 9 biochemical species, 3 observables and 19 parameters. The calibration data includes 3 experimental conditions with a total of 144 datapoints. Using the Hass et al. (2019) benchmark, this problem has a broad eigenvalue spectrum, indicating a sloppy model and overall poor optimizer performance, as the best objective function value is only reached once.

## 2.10 Simulation and Optimization Settings

We encountered difficulties reproducing results from (Hass et al., 2019) and decided to repeat the evaluation using the latest version of data2dynamics. We deactivated Bessel correction (Hass et al., 2019) and and increased the function evaluation limit to match the iteration limit (see supplementary materials).

For all problems, relative and absolute integration tolerances were set to $10^{-8}$. The maxi-

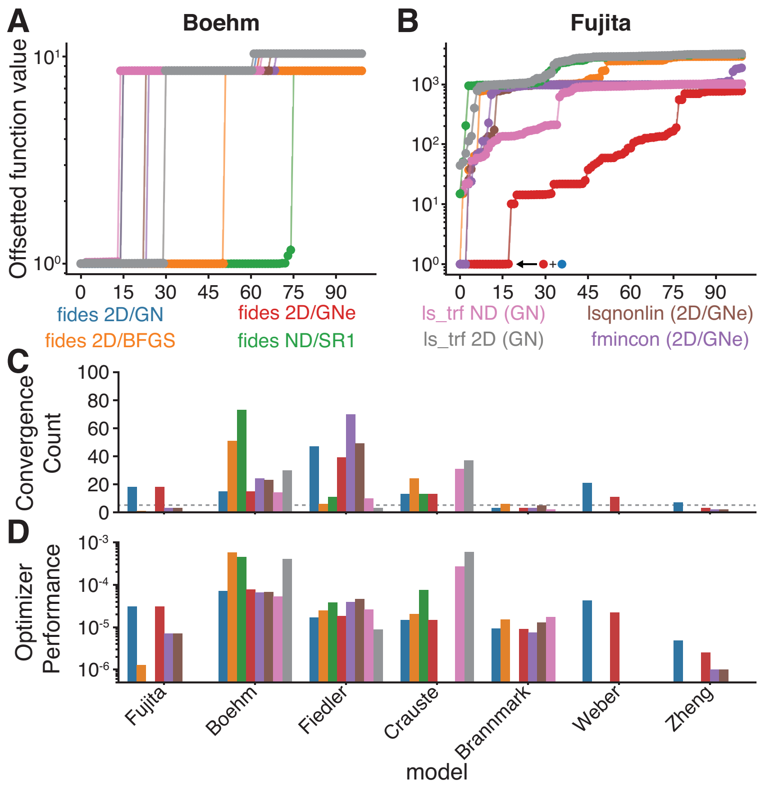

Figure 2: Comparison of MATLAB and python optimizers. Color scheme is the same in all subplots. **A,B:** Waterfall plot of top 100 optimizer runs for *Boehm* and *Fujita* benchmarks. Final objective function values are offsetted by substracting lowest value across all methods and runs and adding 1. **C,D:** Convergence count $n_{success}$ and optimizer performance $\phi$ for all evaluated methods and benchmark problems.

mum number of iterations for optimization was set to $10^5$. Convergence criteria were limited to step sizes with a tolerance of $10^{-6}$, where `ls_trf` code was modified such that the applied convergence criteria matched the implementation in other optimizers. For all problems, we performed $10^3$ optimizer runs. Initial parameter values were uniformly sampled in the log-transformed parameter domain and equal across all python optimizers. Optimization was performed in logarithmic coordinates for all optimizers.

## 3 Results

### 3.1 Validation and Optimizer Comparison

The implementation of trust-region optimization involves complex mathematical operations that can result in error-prone implementations. To validate the trust-region methods implemented in fides, we compared the performance of different combinations of subspace choices and Hesssian approximations against implementations of the same algorithm in MATLAB (`fmincon`, `lsqnonlin`) and python (`ls_trf`). In the following section, the application of fides with different subspace solvers and Hessian approximation methods is denoted by `fides` *subspace/hessian*.

For the *Boehm* example (Fig 2A), we observed consistent final objective function values, but `fides 2D/BFGS` (orange) and `fides ND/SR1` (green) reached the best objective function value about twice as often (`fides 2D/BFGS`: 51, `fides ND/SR1`: 73) compared to other python (blue, red, pink, grey) or MATLAB methods (brown, purple) using the Gauss-Newton approach (`fides 2D/GN`: 15, `fides 2D/GNe`: 15, `fmincon`: 24, `lsqnonlin`: 23, `ls_trf 2D/GN`: 30, `ls_trf ND/GN`: 14). For the *Fujita* example (Fig 2B), `fides 2D/GN` (blue) and `fides 2D/GNe` (red) are the only method that obtains consistent objective function values with 40 starts consistently converging to 3 distinct objective function values and 18 converging to the best objective function value. MATLAB optimizers (brown/purple) and `fides 2D/BFGS` (orange) reached the best objective function value, but don't do so repeatedly (`fides 2D/BFGS`: 1, `fmincon`: 3, `lsqnonlin`: 3). `fides ND/SR1` (green), `ls_trf 2D/GN` (pink), `ls_trf ND/GN` (grey) did not reach the best local minimum.

Overall, we found that `fides 2D/GN` and `fides 2D/GNe` were the only methods that reached the best objective function value for all 7 benchmark problems at least once and more than 5 times (dashed line) for 6 benchmark problems (Fig 2C). For problems where `fmincon` and/or `lsqnonlin` reached the best objective function value (*Fujita*, *Boehm*, *Fiedler*, *Brannmark*, *Zheng*), optimizer performances $\phi$, was similar (0.79 to 2.78 fold change, *Boehm*, *Fiedler*, *Brannmark*) or worse (0.06 to 0.22 fold change, *Fujita*, *Zheng*) for , `fmincon` (purple) and `lsqnonlin` (brown) compared to `fides 2D/GN` (blue) (Fig 2D). We observed similar (0.46 to 1.08 fold change) performance $\phi$ for `fides 2D/GN` (blue) and `fides 2D/GNe` (red) for all examples, indicating that difference in Hessian approximation was not responsible for differences in performance between `fides`, `lsqnonlin` and `fmincon`. While `ls_trf 2D` and `ls_trf ND` outperform `fides 2D/GN` (18.89 to 41.35 fold change) and all other methods on the *Crauste* example, they perform similar to `fides 2D/GN` (0.75 to 1.56 fold change) on three examples (*Boehm*, *Fiedler*, *Brannmark*) with the exception of `ls_trf 2D` on the *Boehm* example, which performs better (5.70 fold change), and did not reach the best local minimum the remaining examples (*Fujita*, *Zheng*, *Weber*). This identifies `fides 2D/GN` as good reference method that has comparable or better performance than `fmincon`, `lsqnonlin` and `ls_trf` on a majority of examples (*Fujita*, *Fiedler*, *Brannmark*, *Weber*, *Zheng*).

`Fides ND/SR1` and `fides 2D/BFGS` perform similar or better (1.41 to 8.11 fold change) than `fides 2D/GN` on three examples (*Boehm*, *Fiedler*, *Crauste*), even though they achieve substantially lower (*Fiedler*, 11/6 vs. 45) or substantially higher (*Boehm*, 71/51 vs. 15) convergence counts. However they fail to reach the best objective function value attained by `fides 2D/GN` in four (*Fujita*, *Crauste*, *Weber*, *Zheng*) and three examples (*Crauste*, *Weber*, *Zheng*) respectively.

Overall, these findings demonstrate that trust-region optimization implemented in fides is competitive with the MATLAB optimizers `fmincon` and `lsqnonlin` and the python optimizer `ls_trf`, outperforming them on a majority of examples. At the same time, this also demonstrates a surprisingly high variability in optimizer performance among methods that all implement the same algorithm.

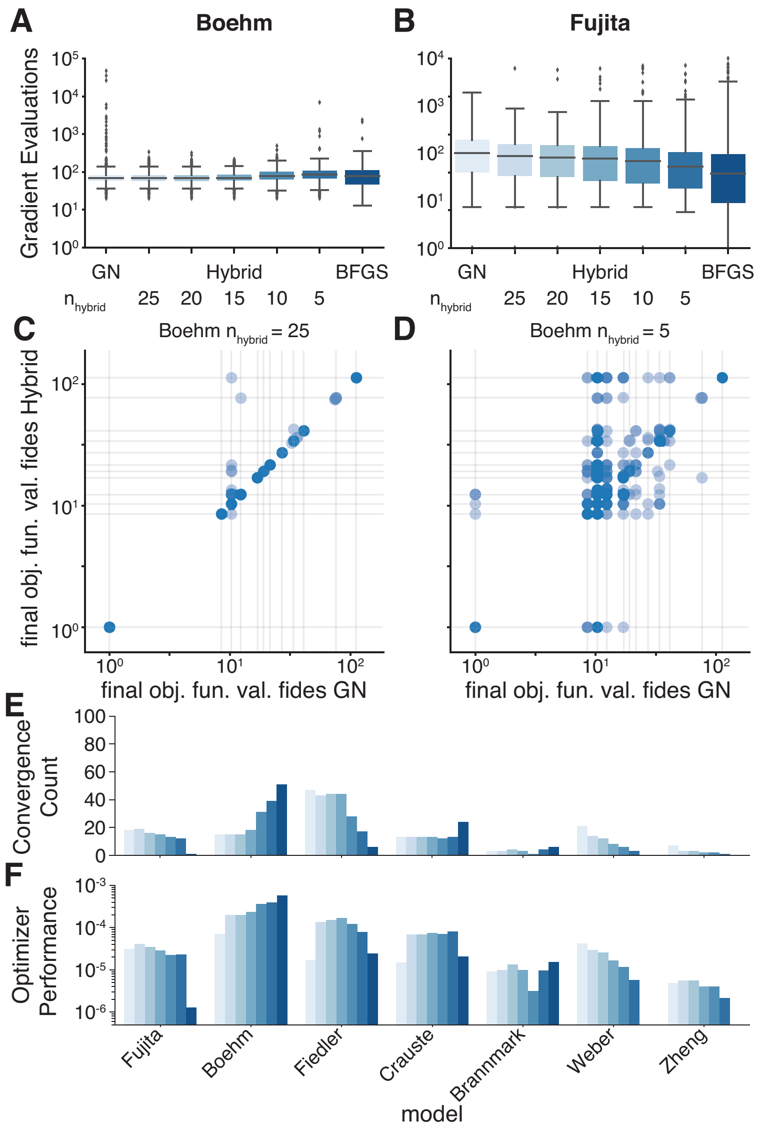

Figure 3: Evaluation of hybrid Hessian approximation scheme. **A,B:** Iteration statistics across all $10^3$ optimizer runs for *Boehm* and *Fujita* benchmarks. Box with line indicate median and first and third quantiles. Whiskers indicate minimum and maximum excluding outliers, which are shown as diamonds. **C,D:** Comparison of final objective function values between hybrid and GN Hessian approximation. Each dot corresponds to optimizer runs with the same initialization for both methods. Values are offsetted by subtracting lowest value across all methods and runs and adding 1. **E,F:** Convergence count $n_{success}$ and optimizer performance $\phi$ for all evaluated methods and benchmark problems.

## 3.2 Hybrid Approximation Scheme

We were surprised that `fides 2D/BFGS` outperformed `fides 2D/GN` for the *Boehm* problem but failed on *Fujita*, *Weber* and *Zheng* examples. The BFGS approximations is guaranteed to converge to the exact Hessian under mild assumptions (Nocedal and Wright, 2006), but requires at least $n_\theta/2$ gradient evaluations to build a full rank approximation. In contrast, the GN approximation can immediately construct a full rank approximation, but generally does not converge to the exact Hessian unless all residuals become zero. We observed that some optimization runs using the GN approximation required high number of iterations with long periods with no updates to the trust region radius. We hypothesize that this might be a result of the poor approximation quality of GN, which motivated us to develop a novel hybrid approach that initially uses the GN approach, constructs a BFGS approximation from the first iteration and switches to this BFGS approach when the trust region radius was not updated for at least $n_{hybrid}$ iterations.

We compared the hybrid approach using different values of $n_{hybrid}$ (5, 10, 15, 20, 25) to `fides 2D/GN` and `fides 2D/BFGS`. For the *Boehm* and *Fujita* benchmarks, we found that while the hybrid approach does not have a strong impact on the median number of gradient evaluations (Fig 3A,B), it reduces the number of optimization runs requiring a high number of gradient evaluations. This improves, optimization performance by reducing the total number of necessary gradient evaluations.

To analyse the impact of the hybrid approach on global convergence properties, we compared paired final objective function values for all optimizer runs between `fides 2D/GN` and the `fides 2D/hybrid` for late ($n_{hybrid} = 25$, Fig 3C) and early switching ($n_{hybrid} = 5$, Fig 3D). We found that late switching has only minimal impact on global convergence properties, as both methods consistently converge to the same final objective function value. The opposite is true for early switching, suggesting that BFGS and GN approximations have substantial impact on the region of attraction of local minima, thereby influencing global convergence properties.

We evaluated the convergence count $n_{success}$ (Fig 3E) and optimizer performance $\phi$ (Fig 3F) of GN (light blue), BFGS (dark blue) and hybrid (light to dark shadings of blue). We found that, while $n_{success}$ of the hybrid approach varied between the values obtained for `fides 2D/GN` and `fides 2D/BFGS`, the hybrid approach outperforms both GN and BFGS in terms of $\phi$ on a majority of benchmark problems (*Fujita*, *Fiedler*, *Crauste*, *Brannmark*) and achieves performance between `fides 2D/GN` and `fides 2D/BFGS` on the remaining examples. Overall, we found that the hybrid approach with $n_{hybrid} = 20$ exhibited comparable or better performance than `fides 2D/GN` on all problems, providing an average increase of 2.04 fold and maximal increase of 9.1 fold (`Fiedler`) across all models. This suggests that local convergence of `fides 2D/GN` is slowed by the poor approximation quality of GN. Thus, superior performance of the hybrid approach over both GN and BFGS approximations is most consistently achieved by a late switching by improves local convergence through BFGS and retaining good global convergence of GN.

## 3.3 Negative Curvature and Saddle Points

The good performance of the fides ND/SR1 approach for the *Boehm* benchmark prompted us to further investigate whether this was the result of employing a Hessian approximation that can account for negative curvature, which is important for problems featuring saddle points, or whether this was the result of using the exact solution to the trust-region subproblem, which should theoretically improve local convergence rates.

We compared fides 2D/GN (dark blue), fides ND/GN (light blue), fides 2D/SR1 (dark orange) and fides ND/SR1 (light orange) (Fig 4). Overall we did not find any consistent

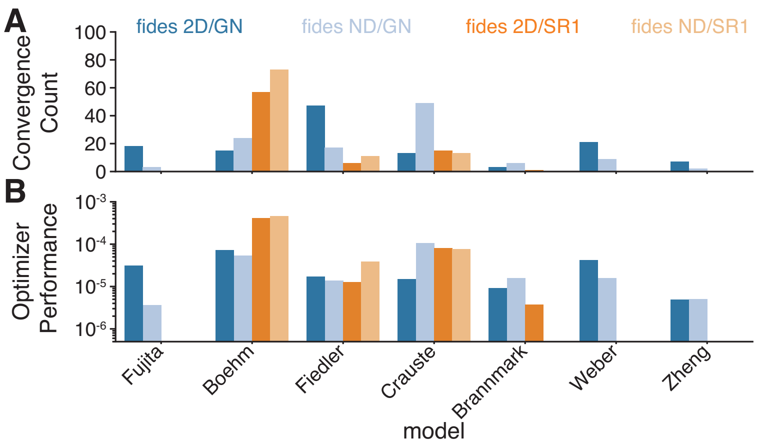

Figure 4: Analysis of saddle point influence on optimization result. **A,B**: Convergence count $n_{success}$ and optimizer performance $\phi$ for all evluated methods and benchmark problems.

trend across all benchmark examples. While the SR1 approximation improved convergence count on one example (*Boehm*), noteably the only non-sloppy example, it decreased the convergence count on all other examples. In three examples (*Fujita*, *Fiedler*, *Crauste*), this is at least partially compensated for by fewer necessary gradient evaluations, suggesting better local convergence, but for three benchmarks, the best objective function value is not reached at all (*Brannmark*, *Weber*, *Zheng*). This suggest that while saddle points may be present for some problems, they do not seem to pose a major issue for the examples we have investigated or at least not one that can be adequately resolved by the use of the SR1 approximation.

## 3.4 Parameter Boundaries and Stepback Strategies

The poor performance of all methods on *Brannmark*, *Weber* and *Zheng* examples was not expected. Only `fides 2D/GN` was able to consistently reach the best objective function value (Fig 5A), and decided to investigate possible sources. Analysis of parameter estimates revealed that for these problems, a large number of parameter estimates were close to the boundary $\partial\Theta$ of the search domain $\Theta$ (distance $< 10^{-2}$ in $\log_{10}$ space). Further investigation revealed that a large number of unique parameter combinations close to the boundaries $\partial\Theta$ were characteristic for these three benchmarks (152 to 865 combinations compared to 11 to 81 combinations, Fig 5B). We hypothesized that this might be the result of local minima on $\partial\Theta$, and accordingly quantified the ratio between the squared norm of the gradient with respect to those parameters close to the boundary $\nabla_b J(\theta)$ and the norm of the full gradient $\nabla J(\theta)$:

$$\frac{||\nabla_b J(\theta)||^2}{||\nabla J(\theta)||^2}, \ (\nabla_b J(\theta))_i = \begin{cases} \frac{\partial J}{\partial \theta_i} & \text{if } \min(\theta_i - l_i, u_i - \theta_i) < 10^{-2} \\ 0 & \text{else.} \end{cases} \tag{9}$$

For all problems except the *Crauste* benchmark, we found that only a small number of optimization runs yielded parameter estimates for which the fraction in (9) was smaller than 0.9 (0 to 77 runs compared to 274 runs), suggesting that most of the previously identified termination points close to parameter boundaries where not proximate to local minima and instead

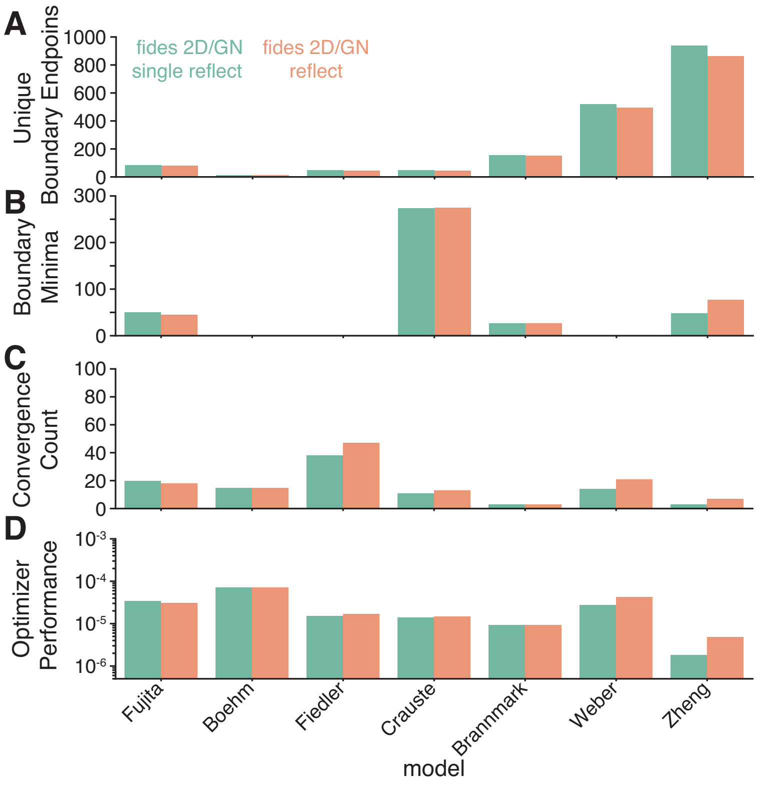

Figure 5: Evaluation of stepback strategies. **A:** Unique combinations of parameters close that are close to boundaries at the end of an optimizer run. **B:** Number of optimizer runs that terminated in a local minimum at the boundary. **C:** Convergence count $n_{success}$. **D:** Optimizer performance $\phi$.

the resulted of premature optimizer termination. The results for *Fujita* and *Crauste*, which feature many minima at the boundary, but few unique parameter combinations, suggests that minima at or near corners of $\Theta$ pose particularly challenging optimization problems.

We hypothesize that the good performance of fides as compared to other optimizers have resulted from a fully reflective strategy, which can reflect at all boundaries when close to multiple parameter boundaries. In contrast, the single reflective method, as employed by `fmincon`, `lsqnonlin` and `ls_trf`, needs to reduce step size when a second boundary is encountered. Accordingly, we compared the fully reflective method (orange) to the single reflective method (green) on all benchmarks (Fig 5B-E). The fully reflective approach substantially improved $n_{success}$ for *Weber* (14 to 21, 1.5 fold change) and *Zheng* (3 to 7, 2.30 fold change) examples, which resulted in a similar increase in performance by 1.56 and 3.28 fold respectively. This further corroborates the hypothesis that tight parameter boundaries can cause poor optimizer performance and shows that performance can be improved by using a fully reflective approach.

## 4 Discussion

In this paper we evaluated trust-region methods implemented in fides against python and MATLAB implementations of the same algorithm on seven benchmark examples and we proposed a new hybrid Hessian approximation scheme. The evaluation showed that fides can resolve issues with inconsistent final objective function values for all examples and that optimization performance could be further enhanced by using a novel hybrid Hessian approximation scheme. This suggests that previously encountered issues with `fmincon` and `lsqnonlin` are due to to poor optimizer performance that leads to premature optimizer termination. Specifically, we identified tight parameter boundaries as a possible reason for premature termination. This could be partially resolved by using a fully reflective versus a single reflective approach. Moreover, our results corroborate previous findings showing that the use of Gauss-Newton approximations can be problematic for optimization problems featuring sloppy models but that saddle points do not pose a major concern. Nonetheless, the inconsistent and sometimes poor performance of BFGS and SR1 approaches and exact subproblem solvers was unexpected, since they should theoretically have better local convergence properties than the Gauss-Newton approximation. However, our findings suggest worse global convergence properties, as indicated by lower convergence counts $n_{success}$. Global convergence properties depend on the shape of the objective function landscape and are therefore expected to be problem specific. Accordingly, BFGS and SR1 may perform better when combined with hybrid global-local methods such as scatter search (Egea et al., 2007), which substantially benefit from good local convergence (Villaverde et al., 2019), but rely less on global convergence properties of local optimizers. Moreover, SR1 and BFGS approaches enable the use of trust-region optimization for problems where the GN approximation is not applicable, e.g., when a non-Gaussian error model is used (Maier et al., 2017) or when gradients are computed using adjoint sensitivities (Fröhlich et al., 2017).

Despite our having implemented the same optimization algorithm, we observed pronounced differences in optimizer performance across benchmark problems, where fides performed most reliably among all evaluated implementations. We believe the most likely reason for the difference in reliability is handling of numerical edge cases in each implementation. For example, fides uses a Moore-Penrose pseudoinverse to compute the Newton search direction for the 2D subproblem solver, while `fmincon` uses conjugate gradients and warns the user in cases of ill-conditioning. Another possible source of difference is the use of different simulation and sensitivity computation routines. While both data2dynamics and AMICI employ CVODES (Hindmarsh et al., 2005) for simulation and computation of parameter sensitivity, there may be slight differences in implementation of advanced features such as handling of events or pre-equilibration. Lastly, optimizers differ in their treatment of integration failures, which `fmincon`, `lsqnonlin` and `fides` handle by decreasing the trust region radius, while `ls_trf` terminates optimization immediately. Overall, these findings demonstrate the complexity of implementing trust-region methods and the importance of numerical subtleties for optimizer performance. For many examples, observed differences in performance among theoretically identical algorithms were similar in magnitude as differences observed with different algorithms. Thus, the evaluation of algorithms requires consistent implementations. We anticipate that the modular implementation of fides will ease some of this burden and drive methodological innovation in the future.

Overall, our results demonstrate that fides not only finds better solutions to objective function optimization when state of the art algorithms fail, but also performs on a par or better on problems where established methods find good solutions. We expect that good performance will generalize to other optimization problems beyond ODE models. Thus, we expect that the modular and flexible implementation of fides will drive widespread adoption within and outside the field of systems biology.

## Acknowledgements

This work was supported by the Human Frontier Science Program (Grant no. LT000259/2019-L1; F.F.), and the National Institute of Health (Grant no. U54-CA225088; F.F.). We thank the O2 High Performance Compute Cluster at Harvard Medical School for computing support. We thank Carolin Loos, Daniel Weindl, Dilan Pathirana, Elba Raimundez, Erika Dudkin, Jan Hasenauer and Leonard Schmiester for providing Benchmark examples in PEtab format. We thank Jan Hasenaur for feedback on the manuscript and Daniel Weindl for feedback on the implementation of fides.

## References

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

Boehm, M. E., Adlung, L., Schilling, M., Roth, S., Klingmüller, U., and Lehmann, W. D. (2014). Identification of Isoform-Specific Dynamics in Phosphorylation-Dependent STAT5 Dimerization by Quantitative Mass Spectrometry and Mathematical Modeling. *Journal of Proteome Research*, 13(12):5685–5694. Publisher: American Chemical Society.

Broyden, C. G. (1970). The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90.

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

Coleman, T. F. and Li, Y. (1992). On the convergence of reflective Newton Methods for large-scale nonlinear minimization subject to bounds. *Mathetmatical Programming*, pages 1–36. bibtex: ColemanLi1992.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (1991). Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50(1):177–195.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., and Cho, K. (2014). 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.

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

Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322.

Fröhlich, F., Kaltenbacher, B., Theis, F. J., and Hasenauer, J. (2017). Scalable parameter estimation for genome-scale biochemical reaction networks. *PLoS Computational Biology*, 13(1):1–18. bibtex: FroehlichKal2017.

Fröhlich, F., Loos, C., and Hasenauer, J. (2019). Scalable Inference of Ordinary Differential Equation Models of Biochemical Processes. In Sanguinetti, G. and Huynh-Thu, V. A., editors, *Gene Regulatory Networks: Methods and Protocols*, Methods in Molecular Biology, pages 385–422. Springer, New York, NY.

Fröhlich, F., Weindl, D., Schälte, Y., Pathirana, D., Paszkowski, L., Lines, G. T., Stapor, P., and Hasenauer, J. (2021). AMICI: High-Performance Sensitivity Analysis for Large Ordinary Differential Equation Models. *Bioinformatics*, (btab227).

Fujita, K. A., Toyoshima, Y., Uda, S., Ozaki, Y.-i., Kubota, H., and Kuroda, S. (2010). Decoupling of Receptor and Downstream Signals in the Akt Pathway by Its Low-Pass Filter Characteristics. *Science Signaling*, 3(132):ra56–ra56. Publisher: American Association for the Advancement of Science Section: Research Article.

Goldfarb, D. (1970). A Family of Variable-Metric Methods Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26.

Gould, N. I. M., Lucidi, S., Roma, M., and Toint, P. L. (1999). Solving the Trust-Region Subproblem using the Lanczos Method. *SIAM Journal on Optimization*, 9(2):504–525. Publisher: Society for Industrial and Applied Mathematics.

Gutenkunst, R. N., Waterfall, J. J., Casey, F. P., Brown, K. S., Myers, C. R., and Sethna, J. P. (2007). Universally sloppy parameter sensitivities in systems biology models. *PLOS Computational Biology*, 3(10):1871–1878. bibtex: GutenkunstWat2007.

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

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

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

Loos, C., Moeller, K., Fröhlich, F., Hucho, T., and Hasenauer, J. (2018). A Hierarchical, Data-Driven Approach to Modeling Single-Cell Populations Predicts Latent Causes of Cell-To-Cell Variability. *Cell Systems*, 6(5):593–603.e13.

Maier, C., Loos, C., and Hasenauer, J. (2017). Robust parameter estimation for dynamical systems from outlier-corrupted data. *Bioinformatics*, 33(5):718–725. bibtex: MaierLoo2017.

More, J. J. (1978). The Levenberg-Marquardt algorithm: Implementation and theory. *Lecture Notes in Mathematics*, 630:105–116.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Raue, A. (2013). Quantitative Dynamic Modeling: Theory and Application to Signal Transduction in the Erythropoietic System.

Raue, A. et al. (2013). Lessons learned from quantitative dynamical modeling in systems biology. *PLoS ONE*, 8(9):e74335.

Raue, A. et al. (2015). Data2Dynamics: a modeling environment tailored to parameter estimation in dynamical systems. *Bioinformatics*, 31(21):3558–3560.

Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., and Timmer, J. (2009). Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(25):1923–1929.

Schmiester, L. et al. (2021). PEtab—Interoperable specification of parameter estimation problems in systems biology. *PLOS Computational Biology*, 17(1):e1008646. Publisher: Public Library of Science.

Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656.

Stapor, P., Fröhlich, F., and Hasenauer, J. (2018). Optimization and profile calculation of ODE models using second order adjoint sensitivity analysis. *Bioinformatics*, 34(13):i151–i159.

Steiert, B., Timmer, J., and Kreutz, C. (2016). L1 regularization facilitates detection of cell type-specific parameters in dynamical systems. *Bioinformatics*, 32(17):i718–i726.

Transtrum, M. K., Machta, B. B., and Sethna, J. P. (2011). Geometry of nonlinear least squares with applications to sloppy models and optimization. *Physical Review E*, 83(3):036701. Publisher: American Physical Society.

Tönsing, C., Timmer, J., and Kreutz, C. (2019). Optimal Paths Between Parameter Estimates in Non-linear ODE Systems Using the Nudged Elastic Band Method. *Frontiers in Physics*, 7. Publisher: Frontiers.

Villaverde, A. F., Fröhlich, F., Weindl, D., Hasenauer, J., and Banga, J. R. (2019). Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics*, 35(5):830–838.