SUPPLEMENTARY INFORMATION

Data2Dynamics: a modeling environment tailored to parameter estimation in dynamical systems

A. Raue [1,], B. Steiert[2], M. Schelker[3], C. Kreutz[2], T. Maiwald[2], H. Hass[2], J. Vanlier[2], C. Tönsing[2], L. Adlung[4], R. Engesser[2], W. Mader[2], T. Heinemann[5], J. Hasenauer[6], M. Schilling[4], T. Höfer[5], E. Klipp[2], F. Theis[6], U. Klingmüller[4], B. Schöberl[1] and J. Timmer [2,7]

[1]Merrimack Pharmaceuticals Inc., 02139 Cambridge, MA, USA
[2]University of Freiburg, Institute for Physics, 79104 Freiburg, Germany
[3]Humboldt-Universität zu Berlin, Theoretical Biophysics, 10115 Berlin, Germany
[4]German Cancer Research Center, 69120 Heidelberg, Germany
[5]BioQuant, University of Heidelberg, 69120 Freiburg, Germany
[6]Helmholtz Center Munich, 85764 Neuherberg, Germany
[7]BIOSS Centre for Biological Signalling Studies, University of Freiburg, 79104 Germany

Implementation and System Requirements

The Data2Dynamics modeling environment is MATLAB-based, open source and freely available. MATLAB R2012 or later is recommended, the MATLAB Optimization Toolbox and the Symbolic Math Toolbox are required. Windows, Mac and Linux operating systems are supported. For MATLAB mex-compilation under Windows systems, the freely available Windows SDK 7.1 is required.

Availability, Documentation and Bug Reports

The Data2Dynamics modeling environment is available at **http://www.data2dynamics.org**. The website contains full a documentation and a description of the example applications. Code changes between versions can be tracked, bug reports can be issued and code improvement and additional functionality can be contributed. Participation is highly welcome!

Contact

andreas.raue@fdm.uni-freiburg.de
jeti@fdm.uni-freiburg.de

Contents

1 Efficient solution of the ODE system

The dynamics of biochemical reaction networks, i.e. the time evolution of the concentrations of the involved molecular compounds, can be modeled by a system of ordinary differential equations (ODE)

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t,\boldsymbol{\theta}) = \mathbf{f}_x(\mathbf{x}(t,\boldsymbol{\theta}),\mathbf{u}(t,\boldsymbol{\theta}),\boldsymbol{\theta}). \tag{1}$$

The variables $\mathbf{x}$ correspond to the dynamics of the concentration of n molecular compounds such as hormones, proteins in different phosphorylation states, mRNA or complexes thereof. A time-dependent experimental treatment that alters the dynamical behavior of the system can be incorporated by the function $\mathbf{u}(t,\boldsymbol{\theta})$. To be able to handle unknown quantities in these driving inputs, we consider the case where the function $\mathbf{u}$ can also be parameter dependent. For instance, $\mathbf{u}$ can represented by a smoothing spline as described in Schelker et al. (2012). The initial state of the system is described by

$$\mathbf{x}(0,\boldsymbol{\theta}) = \mathbf{f}_{x_0}(\boldsymbol{\theta}). \tag{2}$$

The set of parameters $\boldsymbol{\theta} = \{\theta_1 \dots \theta_l\}$ determines the dynamics. Except for very small systems the analytical solution to Equation (1) is not available any more. Therefore, the dynamics have to be computed by a numerical ODE solver. For ODE systems describing biochemical reactions, a reaction flux occurs multiple times in the right hand side of Equation (1). Therefore, the right hand side of Equation (1) can usually be decomposed into a stoichiometry matrix $\mathbf{N}$ and rate equations $\mathbf{v}$ of the reaction fluxes of the molecular interactions

$$\mathbf{f}_x(\mathbf{x}(t,\boldsymbol{\theta}),\mathbf{u}(t,\boldsymbol{\theta}),\boldsymbol{\theta}) = \mathbf{N}\cdot\mathbf{v}(\mathbf{x}(t,\boldsymbol{\theta}),\mathbf{u}(t,\boldsymbol{\theta}),\boldsymbol{\theta}). \tag{3}$$

The stoichiometry matrix $\mathbf{N}$ is sparse. For numerical efficiency it is advantageous to precompute the reaction fluxes $\mathbf{v}$ and use Equation (3) as right hand side of the ODE system.

The timescales of cellular processes can differ by orders of magnitude. As a consequence, the resulting nonlinear ordinary differential equations can be stiff, for a general introduction into this topic see Lambert (1977). Roughly speaking, stiffness is characterized by the presence of at least one rapidly damped mode whose time constant is small compared to the timescale of the remaining dynamics. In the presented software package, the CVODES solver (Hindmarsh et al., 2005) that is implemented in C for efficiency is used. It solves both stiff and non-stiff systems. The methods used in CVODES are variable-order, variable-step multistep methods, based on the Adams-Moulton formulas for non-stiff and on the Backward Differentiation Formulas for stiff systems (Byrne and Hindmarsh, 1975). CVODES also allows for efficient simultaneous solution of the sensitivity equations, see Section 5. For convenience the solver was implemented in combination with a MATLAB mex-interface. For numerical efficiency it is favorable to provide the Jacobian matrix $\mathbf{J}$ of the right hand side of Equation (1) with respect to the dynamic variables $\mathbf{x}$

$$\mathbf{J} = \frac{\partial \mathbf{f}_x}{\partial \mathbf{x}} = \mathbf{N} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{x}} \tag{4}$$

to the ODE solver. Here, again, Equation (3) can be exploited to simplify computations.

2 Parallelization of computations

Experimental measurements are mathematically represented by functional mappings

$$\mathbf{y}(t, \boldsymbol{\theta}) = \mathbf{f}_y(\mathbf{x}(t, \boldsymbol{\theta}), \mathbf{u}(t, \boldsymbol{\theta}), \boldsymbol{\theta}). \tag{5}$$

For each experimental condition Equation (3) and (4) have to be modified according to the treatment applied. This can for example be a different network structure represented by $\mathbf{N}$ due to knock-out, knock-in, or inhibition experiments or different experimental treatments incorporated by the function $\mathbf{u}(t, \boldsymbol{\theta})$ or $\mathbf{f}_{x_0}(\boldsymbol{\theta})$. Consequently, there can be as many variants of the ODE systems as experimental conditions, each having an individual numerical solution for the dynamics. For comparing the whole model to the experimental data, given a specific set of candidate parameters, all ODE variants have to be solved. During parameter estimation many evaluations of the whole model are necessary. This can be a numerically intensive task. All ODE variants can be solved independently, therefore this problem is ideal for parallel computing. Here, a multithreading technique was employed that results in a significant acceleration on multi-core machines. For the model of Bachmann et al. (2011), 24 ODE variants have to be solved consecutively. As displayed in Figure 1, the speed up increases as expected with increasing number of threads used.

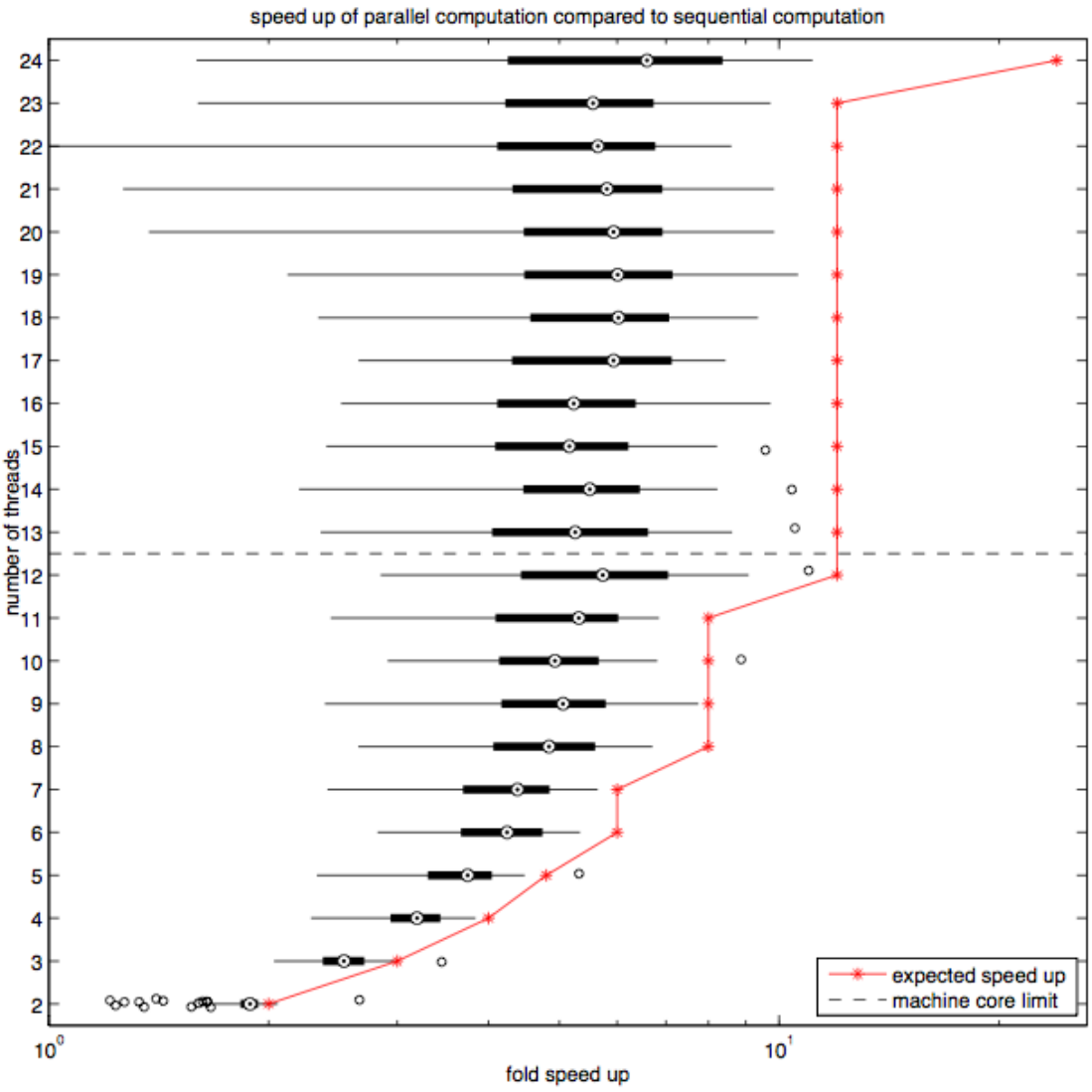

Figure 1: Acceleration of numerical computations for the solution of ODE systems by multithreading. For each bar, 24 variants of the ODE systems used for the model by Bachmann et al. (2011) were solved for 1000 randomly drawn sets of parameters using a Latin hypercube sampling strategy. The red line displays the theoretically possible acceleration. A 12-core processor was used in this benchmark.

3 Maximum Likelihood estimation and deterministic optimization algorithms

Numerical optimization algorithms try to find the argument $\hat{\boldsymbol{\theta}}$ that minimizes the value of an objective function $L(\boldsymbol{\theta})$. In the case of independently and normally distributed measurement noise the objective function for maximum likelihood estimation is given by

$$L(\boldsymbol{\theta}) = \sum_{k=1}^{m} \sum_{i=1}^{d_k} 2\log(\sqrt{2\pi}) + 2\log(\sigma_k(t_i, \boldsymbol{\theta})) + \left(\frac{y_{ki}^{\dagger} - y_k(t_i, \boldsymbol{\theta})}{\sigma_k(t_i, \boldsymbol{\theta})} \right)^2 \tag{6}$$

where $L(\boldsymbol{\theta}) = -2 \cdot \log(\mathcal{L}(\mathbf{y}^{\dagger} | \boldsymbol{\theta}))$ of the likelihood $\mathcal{L}$, $\mathbf{y}^{\dagger}$ is the experimental data and normally distributed measurement noise is assumed. The variance can be a parameterized function $\sigma_k(t_i, \boldsymbol{\theta})^2$, and the corresponding parameter can be estimated together with the other parameters. In the following we will refer to both $\mathcal{L}$ and L simply as likelihood. The minimum of $L(\boldsymbol{\theta})$ is called the best fit of the model to the experimental data and the corresponding parameters $\hat{\boldsymbol{\theta}}$ are the maximum likelihood estimates since they maximize $\mathcal{L}$. Numerical algorithms have to be used to estimate the parameters because of the non-linearity of the optimization problem. Deterministic optimization algorithms try to take steps of $\Delta\boldsymbol{\theta}$ that successively decrease the value of $L(\boldsymbol{\theta})$ beginning from an initial guess $\boldsymbol{\theta}_0$ (Press et al., 1990). The parameters are updated by $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \Delta\boldsymbol{\theta}$. To generate steps, deterministic optimization algorithms evaluate derivatives of the likelihood. The Data2Dynamics software also implements stochastic optimization algorithms, such as particle swarm optimization, differential evolution, simulated annealing, genetic algorithm and others, see Kronfeld et al. (2010) for a detailed description of methods an implementation, for a benchmark comparison, see Raue et al. (2013). The most efficient and reliable algorithm for parameter estimation in our hands is a deterministic trust region approach combined with multi-start strategy to map out local minima. Parameters can and should be estimated on a logarithmic scale.

Gradient descent method. The *gradient descent* method takes steps

$$\Delta\boldsymbol{\theta} = -\gamma \cdot \nabla L(\boldsymbol{\theta}) \tag{7}$$

down the gradient of L. γ is chosen sufficiently small such that a decrease in the objective function $L(\boldsymbol{\theta}_{i+1}) < L(\boldsymbol{\theta}_i)$ can be guaranteed. It is well known that this simple method exhibits only linear convergence rate (Stoer and Bulirsch, 2005). Furthermore, for non-linear problems the method has convergence problems if the objective function contains long and flat valleys (Rosenbrock, 1960). These structures are typical for likelihood functions of ODE models describing biochemical reaction networks, sometimes referred to as *sloppiness* (Gutenkunst et al., 2007).

Newton and quasi-Newton methods. To circumvent convergence problems of the gradient descent method, the *Newton* method applies a second order Taylor expansion of the objective function

$$\tilde{L}(\Delta\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \mathbf{g} \cdot \Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^\top \cdot \mathbf{H} \cdot \Delta\boldsymbol{\theta}, \tag{8}$$

where $\mathbf{g} = \nabla L(\boldsymbol{\theta})$ is the gradient and $\mathbf{H} = \nabla^\top \nabla L(\boldsymbol{\theta})$ is the Hessian matrix of $L(\boldsymbol{\theta})$. For the calculation of $\mathbf{g}$ and $\mathbf{H}$ for ODE models, see in the Section 4. The Newton step is given by

$$\Delta\boldsymbol{\theta} = -\mathbf{H}^{-1} \cdot \mathbf{g} \tag{9}$$

and leads, if the approximation is good, to the optimal solution in a single step. Compared to the gradient descent method, the Newton method exhibits quadratic convergence rate (Stoer and Bulirsch, 2005). The performance decreases considerably as the goodness of the approximation decreases. In this case and also for non-positive definite $\mathbf{H}$, the method can become unstable, i.e. decrease in the objective function is not guaranteed for each step.

Instead of computing $\mathbf{H}$ directly from the model, *quasi-Newton* methods build up information about $\mathbf{H}$ iteratively from sequential evaluations of $\mathbf{g}$, see for example the *Broyden-Fletcher-Goldfarb-Shanno* update (Shanno, 1970). Iterative approaches are especially useful if computation of second order derivatives is not feasible. As explained in Section 4, this is not necessary for ODE models.

Levenberg-Marquardt method. To circumvent stability problems of Newton methods the *Levenberg-Marquardt* method (Marquardt, 1963) uses

$$\Delta\boldsymbol{\theta} = -(\mathbf{H} + \lambda \cdot \mathbf{1})^{-1} \cdot \mathbf{g}. \tag{10}$$

For $\lambda \to 0$ the Newton step of Equation (9) is obtained, for $\lambda \to \infty$ the gradient descent step direction of Equation (7) is obtained. With increasing λ, the step size is reduced. Consequently, $L(\boldsymbol{\theta}_{i+1}) < L(\boldsymbol{\theta}_i)$ can be guaranteed but also efficient Newton steps can be performed if the approximation is a good one. It can be shown that the quality of the approximation (8) increases in the proximity of an optimum in the likelihood (Seber and Wild, 2003).

Trust region methods. Closely related to the Levenberg-Marquardt method are *trust region* methods (Coleman and Li, 1996). These methods minimize Equation (8) under the constraint $||\Delta\boldsymbol{\theta}|| \leq \mu$ to generate the trial step. The parameter μ that defines the radius of the trust region has a similar effect as λ in the Levenberg-Marquardt method, i.e. for small enough μ, $L(\boldsymbol{\theta}_{i+1}) < L(\boldsymbol{\theta}_i)$ can be guaranteed.

Here, the trust region algorithm LSQNONLIN implemented in MATLAB is used. To reduce expensiveness of the constrained minimization of Equation (8), the LSQNONLIN algorithm formulates a two-dimensional approximation to Equation (8) by a plane in parameter space. The gradient descent direction of Equation (7) and an approximate Newton step of Equation (9) span the plane of this new approximation. Since the gradient decent direction is contained $L(\boldsymbol{\theta}_{i+1}) < L(\boldsymbol{\theta}_i)$ for small μ can be guaranteed again. The approximate Newton step is obtained by *preconditioned conjugated gradients* methods (Barrett et al., 1994) instead of using direct matrix inversion that is considerably slower for large parameter space.

4 Derivative calculations for ODE models

For deterministic optimization algorithms, the gradient $\mathrm{g} = \nabla L(\boldsymbol{\theta})$ and Hessian matrix $\mathbf{H} = \nabla^\top \nabla\, L(\boldsymbol{\theta})$ of the likelihood are required. The first term in Equation (6) is a constant. The last term is the well-known weighted sum of squared residuals, sometimes denoted by $\chi^2(\boldsymbol{\theta})$. Equation (6) can be reformulated

$$L(\boldsymbol{\theta}) = const + \sum_{k=1}^{m}\sum_{i=1}^{d_k} 2\log(\sigma_k(t_i,\boldsymbol{\theta})) + \left(\frac{y^\dagger_{ki} - y_k(t_i,\boldsymbol{\theta})}{\sigma_k(t_i,\boldsymbol{\theta})}\right)^2 \tag{11}$$

$$= const + \sum_{q=1}^{s} \tilde{r}_q(\boldsymbol{\theta})^2 + \sum_{q=1}^{s} r_q(\boldsymbol{\theta})^2 \tag{12}$$

where the new sum index q runs over all i and j in (11) and $s = \sum_{k=1}^{m} d_k$. The first sum contains the elements

$$\tilde{r}_q(\boldsymbol{\theta}) = \sqrt{2\log(\sigma_k(t_i,\boldsymbol{\theta}))}. \tag{13}$$

Since $\log(\sigma_k(t_i,\boldsymbol{\theta}))$ can be negative, in practice

$$\tilde{r}_q(\boldsymbol{\theta}) = \sqrt{2\log(\sigma_k(t_i,\boldsymbol{\theta})) + c} \tag{14}$$

is used, where c is a large enough constant. The constant c adds another constant to Equation (12) but does not change the shape of the likelihood function, hence, maximum likelihood estimation and confidence intervals are not affected. The second sum contains the normalized residuals

$$r_q(\boldsymbol{\theta}) = \frac{y^\dagger_{ki} - y_k(t_i,\boldsymbol{\theta})}{\sigma_k(t_i,\boldsymbol{\theta})}. \tag{15}$$

The gradient can then be obtained by

$$\mathrm{g} = \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{\theta}} = 2\sum_{q=1}^{s}\left(\tilde{r}_q \cdot \frac{\mathrm{d}\tilde{r}_q}{\mathrm{d}\boldsymbol{\theta}} + r_q \cdot \frac{\mathrm{d}r_q}{\mathrm{d}\boldsymbol{\theta}}\right) \tag{16}$$

and the Hessian matrix by

$$\mathbf{H} = \frac{\mathrm{d}^2 L}{\mathrm{d}\boldsymbol{\theta}^2} = 2 \sum_{q=1}^{s} \left(\frac{\mathrm{d}\tilde{r}_q}{\mathrm{d}\boldsymbol{\theta}} \cdot \frac{\mathrm{d}\tilde{r}_q}{\mathrm{d}\boldsymbol{\theta}} + \tilde{r}_q \cdot \frac{\mathrm{d}^2 \tilde{r}_q}{\mathrm{d}\boldsymbol{\theta}^2} + \frac{\mathrm{d}r_q}{\mathrm{d}\boldsymbol{\theta}} \cdot \frac{\mathrm{d}r_q}{\mathrm{d}\boldsymbol{\theta}} + r_q \cdot \frac{\mathrm{d}^2 r_q}{\mathrm{d}\boldsymbol{\theta}^2} \right). \tag{17}$$

For the following calculations we look at each component q individually and drop the index for better readability. First and second order derivatives of $\tilde{r}$ can be evaluated by

$$\frac{\mathrm{d}\tilde{r}}{\mathrm{d}\boldsymbol{\theta}} = \frac{1}{\tilde{r}\sigma} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}} \tag{18}$$

and

$$\frac{\mathrm{d}^2 \tilde{r}}{\mathrm{d}\boldsymbol{\theta}^2} = \frac{1}{\tilde{r}\sigma} \frac{\mathrm{d}^2 \sigma}{\mathrm{d}\boldsymbol{\theta}^2} - \frac{1}{\tilde{r}\sigma^2} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}} \cdot \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}} - \frac{1}{\tilde{r}^2\sigma} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}} \cdot \frac{\mathrm{d}\tilde{r}}{\mathrm{d}\boldsymbol{\theta}}. \tag{19}$$

First order derivatives of r can be calculated by

$$\frac{\mathrm{d}r}{\mathrm{d}\boldsymbol{\theta}} = \frac{-1}{\sigma} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}} - \frac{r}{\sigma} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}}. \tag{20}$$

Let f_σ be the equation for the variance $\sigma_k(t_i, \boldsymbol{\theta})^2$. We can then further obtain

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial f_\sigma}{\partial y} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial f_\sigma}{\partial \boldsymbol{\theta}} \tag{21}$$

and

$$\frac{\mathrm{d}^2 \sigma}{\mathrm{d}\boldsymbol{\theta}^2} = 2 \frac{\partial^2 f_\sigma}{\partial y \partial \boldsymbol{\theta}} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial f_\sigma}{\partial y} \frac{\mathrm{d}^2 y}{\mathrm{d}\boldsymbol{\theta}^2} + \frac{\partial^2 f_\sigma}{\partial y^2} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}} + \frac{\mathrm{d}^2 f_\sigma}{\mathrm{d}\boldsymbol{\theta}^2}. \tag{22}$$

With the equation for the measurements f_y we can resolve

$$\frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial f_y}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial f_y}{\partial \mathbf{u}} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial f_y}{\partial \boldsymbol{\theta}}. \tag{23}$$

All expressions in Equations (18-23) can be calculated analytically except for terms indicated in red. In Equation (23), the derivatives of the dynamic variables $\mathrm{d}\mathbf{x}/\mathrm{d}\boldsymbol{\theta}$, the so called *sensitivities*, can be calculated numerically, as described in Section 5. The calculation of $\mathrm{d}^2 y/\mathrm{d}\boldsymbol{\theta}^2$ involves second order sensitivities $\mathrm{d}^2\mathbf{x}/\mathrm{d}\boldsymbol{\theta}^2$ of the dynamics that are expensive to evaluate numerically.

The calculation of the last term in Equation (17), $r \cdot \mathrm{d}^2 r/\mathrm{d}\boldsymbol{\theta}^2$ is more complicated. Following the argumentation in Press et al. (1990, Section 15.5 Nonlinear Models), in the proximity of a good model fit, the normalized residuals r fluctuate in an uncorrelated manner around zero. Therefore, the term tends to cancel out when summed over q and quadratic convergence of optimization algorithms using Newton steps can be obtained without calculating second order sensitivities.

However, if measurement noise parameters are estimated as well, the argumentation is more involved. In the following, the parameters are collected in three groups: $\boldsymbol{\theta}_y$ are parameters that only affect the observables y, $\boldsymbol{\theta}_s$ are parameters that only affect the measurement noise σ and $\boldsymbol{\theta}_b$ are parameters that affect both. Correspondingly, the last term in Equation (17) can be reformulated to

$$r \cdot \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}^2} = r \cdot \begin{pmatrix} \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_y^2} & \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_y \mathrm{d}\boldsymbol{\theta}_s} & \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_y \mathrm{d}\boldsymbol{\theta}_b} \\ \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_s \mathrm{d}\boldsymbol{\theta}_y} & \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_s^2} & \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_s \mathrm{d}\boldsymbol{\theta}_b} \\ \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_b \mathrm{d}\boldsymbol{\theta}_y} & \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_b \mathrm{d}\boldsymbol{\theta}_s} & \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_b^2} \end{pmatrix}.$$

With Equation (15) we can calculate

$$r \cdot \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_y^2} = \frac{-r}{\sigma} \frac{\mathrm{d}^2 y}{\mathrm{d}\boldsymbol{\theta}_y^2} \tag{24}$$

$$r \cdot \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_s^2} = \frac{r^2}{\sigma^2} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_s} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_s} - \frac{r^2}{\sigma} \frac{\mathrm{d}^2 \sigma}{\mathrm{d}\boldsymbol{\theta}_s^2} \tag{25}$$

$$r \cdot \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_b^2} = \frac{r}{\sigma^2} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}_b} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_b} - \frac{r}{\sigma} \frac{\mathrm{d}r}{\mathrm{d}\boldsymbol{\theta}_b} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_b} - \frac{r}{\sigma} \frac{\mathrm{d}^2 y}{\mathrm{d}\boldsymbol{\theta}_b^2} + \frac{r^2}{\sigma^2} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_b} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_b} - \frac{r^2}{\sigma} \frac{\mathrm{d}^2 \sigma}{\mathrm{d}\boldsymbol{\theta}_b^2} \tag{26}$$

$$r \cdot \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_y \mathrm{d}\boldsymbol{\theta}_s} = \frac{r}{\sigma^2} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}_y} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_s} \tag{27}$$

$$r \cdot \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_y \mathrm{d}\boldsymbol{\theta}_b} = \frac{r}{\sigma^2} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{\theta}_y} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_b} - \frac{r}{\sigma} \frac{\mathrm{d}^2 y}{\mathrm{d}\boldsymbol{\theta}_y \mathrm{d}\boldsymbol{\theta}_b} \tag{28}$$

$$r \cdot \frac{\mathrm{d}^2 r}{\mathrm{d}\boldsymbol{\theta}_s \mathrm{d}\boldsymbol{\theta}_b} = \frac{r^2}{\sigma^2} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_s} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_b} - \frac{r}{\sigma} \frac{\mathrm{d}\sigma}{\mathrm{d}\boldsymbol{\theta}_s} \frac{\mathrm{d}r}{\mathrm{d}\boldsymbol{\theta}_b} - \frac{r^2}{\sigma} \frac{\mathrm{d}^2 \sigma}{\mathrm{d}\boldsymbol{\theta}_s \mathrm{d}\boldsymbol{\theta}_b}. \tag{29}$$

Indicated in red color are the before-mentioned second order derivatives that are problematic to evaluate numerically. The terms in Equations (24- 29) do contain either r or r^2. As discussed above, the terms containing r do cancel out when summed over q in the proximity of a good model fit whereas the terms containing r^2 do not. Fortunately, the problematic terms that contain second order derivatives $\mathrm{d}^2 y / \mathrm{d}\boldsymbol{\theta}^2$, marked in red, are amongst those that do cancel out when summed over q in Equations (24- 29). With the remaining terms of Equation (17), an approximate Hessian matrix $\hat{\mathbf{H}}$ can be calculated without the second order derivatives $\mathrm{d}^2 y / \mathrm{d}\boldsymbol{\theta}^2$. As the quality of the model fit improves, the quality of $\hat{\mathbf{H}}$ improves. For cases where the measurement noise function does not explicitly depend on the observables, the missing term in Equation (22) vanishes as well because $\partial f_\sigma / \partial y = 0$. In these cases, as the quality of the model fit improves, $\hat{\mathbf{H}} \to \mathbf{H}$, and quadratic convergence of optimization algorithms can be obtained even if measurement noise parameters are estimated.

The assumption $\partial f_\sigma / \partial y = 0$ holds true for many applications. To which extend more general functions with $\partial f_\sigma / \partial y \neq 0$ for the measurement noise have to be considered for applications, and to which extent this would impact on optimization performance, remains to be clarified.

If, due to log-normal measurement noise, the observables and experimental data points are compared on a logarithmic scale, e.g. using $\log_{10}$, an additional transformation

$$\frac{\mathrm{d}y_k^{\log}}{\mathrm{d}\boldsymbol{\theta}} = \frac{1}{\log(10) \cdot y_k} \frac{\mathrm{d}y_k}{\mathrm{d}\boldsymbol{\theta}} \tag{30}$$

has to be applied. If, for efficiency, parameter values are estimated on a logarithmic scale, e.g. using $\log_{10}$, an additional transformation

$$\frac{\mathrm{d}}{\mathrm{d}\theta_j^{\log}} = \log(10) \cdot \theta_j \cdot \frac{\mathrm{d}}{\mathrm{d}\theta_j} \tag{31}$$

has to be used.

5 Efficient solution of the sensitivity equations

The derivatives $\mathrm{d}\mathbf{x}(t, \boldsymbol{\theta}) / \mathrm{d}\boldsymbol{\theta}$, also called *sensitivities*, can be calculated accurately using the sensitivity equations. The sensitivity equations represent an additional ODE system

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{\mathrm{d}\mathbf{x}(t, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \mathbf{f}_x}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}(t, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \mathbf{f}_x}{\partial \mathbf{u}} \frac{\mathrm{d}\mathbf{u}(t, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \mathbf{f}_x}{\partial \boldsymbol{\theta}} \tag{32}$$

for the derivatives (Leis and Kramer, 1988) that is solved simultaneously with the original ODE system, see Equation (1). Using the abbreviation $s_{ij} = \mathrm{d}x_i / \mathrm{d}\theta_j$ with $i = 1 \ldots n$ and $j = 1 \ldots l$ and Equation (4), we rewrite component-wise

$$\frac{\mathrm{d}}{\mathrm{d}t} s_{ij} = \mathbf{J} \cdot s_{ij} + \frac{\partial \mathbf{f}_x}{\partial \mathbf{u}} \frac{\mathrm{d}\mathbf{u}(t, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \mathbf{f}_x}{\partial \boldsymbol{\theta}}, \tag{33}$$

which are $n \times l$ additional ODEs. The initial conditions for Equation (33) can be calculated by

$$\mathbf{s}(t = 0, \boldsymbol{\theta}) = \frac{\mathrm{d}\mathbf{f}_{x_0}}{\mathrm{d}\boldsymbol{\theta},} \tag{34}$$

see Equation (2). An efficient algorithm for solving the enlarged ODE system is the CVODES solver. There are some numerical properties that allow to increase the performance of solving the enlarged ODE system, see Hindmarsh et al. (2005) for details. Similar to the calculation of the Jacobian matrix $\mathbf{J}$, see Equation (4), of the right hand side of Equation (1), it is favorable to provide analytically the Jacobian matrix $\mathbf{J}_s$ of the

right hand side of Equation (33) with respect to the sensitivities. From Equation (33), we see that $\mathbf{J}_s$ is composed blockwise of $\mathbf{J}$ by

$$\mathbf{J}_s = \begin{pmatrix} \frac{\partial s_{i1}}{\partial s_{i1}} & \frac{\partial s_{i2}}{\partial s_{i1}} & \dots \\ \frac{\partial s_{i1}}{\partial s_{i2}} & \frac{\partial s_{i2}}{\partial s_{i2}} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} = \begin{pmatrix} \mathbf{J} & 0 & \dots \\ 0 & \mathbf{J} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}.$$

Therefore, explicit calculation of $\mathbf{J}_s$ can be omited. In addition, this feature leads to some numerical properties that allow to further increase the performance of solving the enlarged ODE system, see in Hindmarsh et al. (2005) for details. For instance, Equation (32) inherits the stiffness properties of Equation (1). Consequently, the adaptive step size control for the enlarged ODE system can be chosen equally to that of the original system. Again, Equation (3) can be used to simplify computation of Equation (32) by

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}\mathbf{x}(t,\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \mathbf{N}\cdot\left(\frac{\partial\mathbf{v}}{\partial\mathbf{x}}\frac{\mathrm{d}\mathbf{x}(t,\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial\mathbf{v}}{\partial\mathbf{u}}\frac{\mathrm{d}\mathbf{u}(t,\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial\mathbf{v}}{\partial\boldsymbol{\theta}}\right). \tag{35}$$

The matrices $\partial\mathbf{v}/\partial\mathbf{x}$, $\partial\mathbf{v}/\partial\mathbf{u}$ and $\partial\mathbf{v}/\partial\boldsymbol{\theta}$ can be calculated analytically.

References

Bachmann J, Raue A, Schilling M, Böhm M, Kreutz C, Kaschek D, Busch H, Gretz N, Lehmann W, Timmer J, and Klingmüller U. Division of labor by dual feedback regulators controls JAK2/STAT5 signaling over broad ligand range. *Molecular Systems Biology* **7**, 516, 2011.

Barrett R, Berry M, and Chan TF. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, USA, 1994.

Byrne G and Hindmarsh A. A polyalgorithm for the numerical solution of ordinary differential equations. *ACM T Math Software* **1**(1), 71–96, 1975.

Coleman T and Li Y. An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization* **6**(2), 418–445, 1996.

Gutenkunst R, Waterfall J, Casey F, Brown K, Myers C, and Sethna J. Universally sloppy parameter sensitivities in systems biology models. *PLOS Comput Biol* **3**(10), e189, 2007.

Hindmarsh A, Brown P, Grant K, Lee S, Serban R, Shumaker D, and Woodward C. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM T Math Software* **31**(3), 363–396, 2005.

Kronfeld M, Planatscher H, and Zell A. The EvA2 optimization framework. *Learning and Intelligent Optimization* **6073**, 247–250, 2010.

Lambert J. *The State of the Art in Numerical Analysis*, chapter The initial value problem for ordinary differential equations, pages 451–500. Academic Press, London, UK, 1977.

Leis J and Kramer M. The simultaneous solution and sensitivity analysis of systems described by ordinary differential equations. *ACM T Math Software* **14**(1), 45–60, 1988.

Marquardt D. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics* **11**(2), 431–441, 1963.

Press W, Teukolsky S, Flannery B, and Vetterling W. *Numerical Recipes: FORTRAN*. Cambridge University Press, Cambridge, UK, 1990.

Raue A, Schilling M, Bachmann J, Matteson A, Schelker M, Kaschek D, Hug S, Kreutz C, Harms B, Theis F, Klingmüller U, and Timmer J. Lessons learned from quantitative dynamical modeling in systems biology. *PLOS ONE* **8**(9), e74335, 2013.

Rosenbrock H. An automatic method for finding the greatest or least value of a function. *The Computer Journal* **3**(3), 175–184, 1960.

Schelker M, Raue A, Timmer J, and Kreutz C. Comprehensive estimation of input signals and dynamical parameters in biochemical reaction networks. *Bioinformatics* **28**(18), i522–i528, 2012.

Seber G and Wild C. *Nonlinear Regression*. Wiley, Weinheim, Germany, 2003.

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

Stoer J and Bulirsch R. *Numerische Mathematik 2*. Springer, Heidelberg, Germany, 2005.