Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-26 (2019) 32–37

Efficient computation of steady states in large-scale ODE models of biochemical reaction networks

Glenn Terje Lines [*,1] Łukasz Paszkowski [*]
Leonard Schmiester [**,***] Daniel Weindl [**] Paul Stapor [**,***]
Jan Hasenauer [**,***,****]

[*] *Simula Research Laboratory, 1325 Lysaker, Norway*
[**] *Institute of Computational Biology, Helmholtz Zentrum München – German Research Center for Environmental Health, 85764 Neuherberg, Germany*
[***] *Center for Mathematics, Technische Universität München, 85748 Garching, Germany*
[****] *Faculty of Mathematics and Natural Sciences, University of Bonn, 53113 Bonn, Germany*

Abstract:
In systems and computational biology, ordinary differential equations are used for the mechanistic modelling of biochemical networks. These models can easily have hundreds of states and parameters. Typically most parameters are unknown and estimated by fitting model output to observation. During parameter estimation the model needs to be solved repeatedly, sometimes millions of times. This can then be a computational bottleneck, and limits the employment of such models.
In many situations the experimental data provides information about the steady state of the biochemical reaction network. In such cases one only needs to obtain the equilibrium state for a given set of model parameters. In this paper we exploit this fact and solve the steady state problem directly rather than integrating the ODE forward in time until steady state is reached. We use Newton's method – like some previous studies – and develop several improvements to achieve robust convergence. To address the reliance of Newtons method on good initial guesses, we propose a continuation method. We show that the method works robustly in this setting and achieves a speed up of up to 100 compared to using ODE solves.

Keywords: Differential equations, Dynamic modelling, Large-scale systems, Steady states, Steady-state stability, Steady-state errors, Parameter estimation

1. INTRODUCTION

Models of biological processes based on ordinary differential equations (ODEs) have led to great contributions in the field of systems and computational biology. As in recent years, the speed of data acquisition and the availability of computational resources have been increasing, more detailed, and hence, larger models have been developed (Klipp et al., 2005; Chen et al., 2009; Hass et al., 2017; Bouhaddou et al., 2018; Fröhlich et al., 2018). Particularly in the field of cancer signaling, large-scale ODE models provide the possibility of gaining insights into complex processes and finding possible targets for new drugs.

In most applications, ODE models have unknown parameters, which need to be inferred from measurement data in order to yield meaningful predictions. As this parameter estimation problem should not be under-determined, large-scale models necessitate large-scale data sets. The acquisition of such data sets is expensive, and therefore, the most databases do not provide time-resolved data (e.g. the Cancer Cell Line Encyclopedia (Barretina et al., 2012) and the Genomics in Drug Sensitivity in Cancer database (Yang et al., 2013)). Instead, many databases provide only measurements of a given system in steady state.

Tackling such a parameter estimation problem quickly leads to millions of model simulations and solving ODE systems which describe thousands of chemical species is extremely time-consuming. Hence, despite the use of high performance computing infrastructures and although substantial improvements have been made in recent years (see (Hass et al., 2019; Villaverde et al., 2019) for comparisons), computation time remains one of the limiting factors in large-scale modeling. It is therefore desirable to further reduce computation time and to circumvent computationally expensive actions, where possible.

If only steady state data is available, the parameter estimation problem can be addressed without solving the ODE system but by (1) recasting the optimization problem or by (2) computing the steady state without ODE solvers. Concept (1) is used by constrained optimization, optimization

[1] To whom correspondence should be addressed

on manifolds or continuous analogues (see (Fiedler et al., 2016) for an overview). Unfortunately, the convergence of constrained optimization is often unsatisfactory (Fiedler et al., 2016; Rosenblatt et al., 2016) and the scalability of optimization on manifolds and continuous analogues unclear (Fiedler et al., 2016). Concept (2) is based on symbolic or direct numerical calculation of the steady state. Symbolic calculations are the method of choice if the problem possesses a particular structure (King and Altman, 1956) or an excess in the degrees of freedom (Rosenblatt et al., 2016). For more general problems, numerical approaches such as Newton's method or adaptations can be employed and some commonly used toolboxes already provide comparable methods (Hoops et al., 2006; Soetaert, 2009). However, it is a priori not clear whether these approaches are applicable to a given (large) system. In particular, radius of convergence of Newton's method might be too small to yield a reliable implementation.

In this manuscript, we present a study on reliability of Newton's method for this problem type. As application example we use a published large-scale model with measurement data from a public database. A convergence analysis is performed where the basic method is compared to various extensions, e.g. continuation. We illustrate that Newton's method should be considered an option when dealing with steady state data, as it works reliably in many cases and is in general much faster compared to integrating the ODEs until steady state.

2. METHODS

2.1 The parameter estimation problem

The goal of the parameter estimation procedure is to find a parameter vector ϕ that minimizes the difference between the computed output of the model and the observations. This can be formulated as function minimization problem: Find $\phi^* = \arg\min E(\phi)$, where $E(\phi)$ quantifies the distance between model prediction and experimental observations, e.g.:

$$E(\phi) = ||g(x,\phi) - y||^2, \quad \text{where } x \text{ solves } \dot{x} = F(x,\phi).$$

Here y represents the observations, and g is some transformation of the solution. Typically g is a subset and/or a linear combinations of the states in x. The dynamics of x are described by the vector field $F(x,\phi)$ derived from the process of interest. The function $E(\phi)$ is minimized through an iterative procedure that can schematically be written as follows:

Start with some initial parameter vector ϕ.
Repeat until the error is small:

- Compute $E(\phi)$
- Obtain an incremental change $\Delta\phi$
- Update: $\phi := \phi + \Delta\phi$

In order to compute $E(\phi)$ we in general need to solve $\dot{x} = F(x,\phi)$. However, if the observations y are only obtained in steady state, we do not actually need the transient behaviour $x(t)$, but can instead just solve the algebraic problem $F(x,\phi) = 0$.

2.2 Newton's method

We want to solve the algebraic problem $F(x,\phi) = 0$ with respect to x for some given parameter vector ϕ. A standard solution technique is the iterative Newton's method:

$$x^{n+1} = x^n - J(x^n,\phi)^{-1}F(x^n,\phi)$$

Here x^n is the approximation of the solution after n-iterations and $J(x,\phi)$ is the Jacobi matrix of the system function F, i.e. $J_{ij} = \frac{\partial F_i}{\partial x_j}$. The method only works if the initial guess x^0 is sufficiently close to the solution. For a given problem, the method has a certain radius of convergence. Below we suggest a series of modifications to extend this radius.

2.3 Dampening

A common technique to improve the stability of Newton's method is to reduce the step length by a factor γ:

$$x^{n+1} = x^n - \gamma J(x^n,\phi)^{-1}F(x^n,\phi)$$

In this study, we considered an adaptive scheme, where γ is reduced by a factor of 4 if the error is not decreasing, and conversely increased by a factor of 10 if the error is reduced (without exceeding $\gamma = 1$). These values were chosen by trial and error, but the performance was not very sensitive to the exact choice.

2.4 Improved initial guess

We can exploit the fact that we are solving a series of similar problems to construct a better initial condition. If we have previously solved $F(x_0,\phi_0) = 0$, we now want to solve $F(x_1,\phi_1) = 0$ where presumably ϕ_0 and ϕ_1 are close. Instead of using x_0 as the initial guess for the new problem, we do a Taylor expansion in the parameter space to construct a better guess:

$$\begin{aligned} F(x_0+\Delta x, \phi_0+\Delta\phi) \approx\ & F(x_0,\phi_0) + \frac{\partial F}{\partial x}(x_0,\phi_0)\Delta x \\ & + \frac{\partial F}{\partial \phi}(x_0,\phi_0)\Delta\phi \end{aligned}$$

where $\Delta x = x_0 - x_{\text{guess}}$ (to be computed) and $\Delta\phi = \phi_0 - \phi_1$ (given). We compute Δx (and thus x_{guess}) by requiring the right hand side to be zero. Since $F(x_0,\phi_0) = 0$ the system to solve becomes:

$$\frac{\partial F}{\partial x}(x_0,\phi_0)\Delta x = -\frac{\partial F}{\partial \phi}(x_0,\phi_0)\Delta\phi$$

Thus we obtain a potentially much better initial guess at the cost of one linear solve plus one evaluation of the Jacobian with respect to ϕ, i.e. $\frac{\partial F}{\partial \phi}$.

2.5 Continuation

Dampening often improves the convergence properties, but as we will see in the result section it does not always work. A drawback of the approach is that only the step length is modified, but not the search direction. To improve upon this performance we have implemented the method of numerical continuation. The basic idea is to transform the

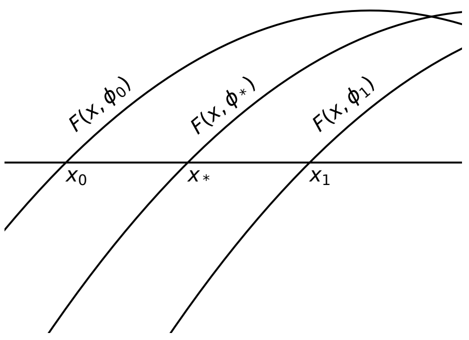

Fig. 1. The initial guess x_0 can potentially be quite bad for the problem $F(x, \phi_1) = 0$, but still work quite well for the easier problem $F(x, \phi_*)$. In that case, the intermediate solution x_* can instead be used as an initial guess.

current problem, $F(x_1, \phi_1) = 0$, to a problem more similar to the previous problem, $F(x_0, \phi_0) = 0$, of which we know the solution.

In case of failure, we then solve the simpler problem $F(x^*, \phi^*) = 0$, where ϕ^* is some average between ϕ_0 and ϕ_1 (see Figure 1).

The closer to ϕ^0, the simpler the problem, but also further from what we want to solve. The step size is controlled with δ: $\phi_\delta = \phi_0^{1-\delta}\phi_1^{\delta}$. We found that $\delta = 1/2$, i.e. $\phi^* = \sqrt{\phi_0\phi_1}$ to be a good compromise. If the new problem also fails, we repeat the process. In case of convergence we retry the harder problem, but now with a better initial guess. Pseudo code of the algorithm is shown in the appendix.

2.6 Implementation

The methods were implemented within the AMICI framework (Fröhlich et al., 2018), which provides an interface to the SUNDIALS library (Hindmarsh et al., 2005). Models specified in SBML are transformed into native C++ code. In particular it generates code for the Jacobian of F needed for Newton's method (derivation with respect to x) and the Jacobian needed for the improved initial guess (derivation with respect to ϕ). The benchmarking code is available at `http://doi.org/10.5281/zenodo.2641916`.

2.7 Model & Data

As a challenging benchmarking problem we consider the large-scale model of cancer signalling introduced by Fröhlich et al. (2018). The model is among the largest ODE models ever developed in systems biology, with 1228 states and 4234 parameters. In the original study, the authors used a simulation approach to compute the steady state. The parameter optimization was computationally demanding and required the use of high performance computing infrastructure, rendering an acceleration using more efficient numerical methods important.

For the evaluation of the methods, we consider a subset of the data used by Fröhlich et al. (2018). From the total of 5281 conditions we selected the 94 conditions corresponding to the control experiments.

Table 1. Performance of ODE solver using x_{eq} as the initial value and an adaptive time stepping scheme and then integrating until steady state.

μ	% fails	CPU (s)
0.01	0	0.887
0.05	0	1.124
0.10	0	1.301
0.20	0	1.463
0.50	0	2.150
1.00	0	3.393

3. RESULTS

3.1 Radius of convergence

In order to systematically evaluate and compare the different approaches we have constructed a series of problems with increasing difficulty. A parameter vector (ϕ) describing the experimental data was first obtained from a previous fitting to observational data (Schmiester et al., 2019). A total of 94 conditions were tested. These are included in the model as a separate constant parameter vector k: $\dot{x} = F(x, \phi, k)$. In each of these 94 cases an equilibrium solution (x_{eq}) was first obtained by solving the corresponding ODE problem until steady state. The methods were then tested by using this equilibrium as the initial guess for Newton's method. The problems were made progressively more challenging by perturbing the parameter vector away from the original steady state:

$$\phi_\epsilon = \phi \cdot 10^{\epsilon}$$

where ϵ was drawn uniformly from the interval $[-\mu, \mu]$. Thus μ sets the potential size of the perturbation. For example, with $\mu = 1$, each component of ϕ_ϵ will lie between 0.1 and 10 times of the corresponding value of the original parameter vector, corresponding to a large perturbation.

For reference, we have included the performance of the ODE solver (Table 1). Here we used x_{eq} as the initial value and then integrated until steady state. An adaptive time stepping scheme was used, and from the time consumption we can see that the problems that are close to the unperturbed original (small values of μ) require significantly less work compared to the cases where the initial value is further from steady state (larger values of μ). Also note that we were able to compute the steady state in all cases (0% failure).

Table 2 shows the performance of Newton's method in the most basic form. There is a dramatic decrease in time consumption compared to solving the ODEs to steady state, with Newton's method being more than a factor of 100 faster. This was achieved by using KLU (Davis and Natarajan, 2010) as the linear solver. We also tried an iterative method (BiCGStab, Van der Vorst (1992)) but for this problem size it was not competitive, being at least a threefold slower. Contrary to the impressive reduction in CPU time, we see from the failed column that the method is not very robust. For moderately large perturbations the failure rate is over 50%.

Table 2. Performance of Newton's method with no dampening using the KLU linear solver. The number of Newton steps among the converged cases are reported in the last column (CPU times and nl-steps, are median numbers for individual solves).

μ	% fails	CPU (s)	nl-steps
0.01	3.9	0.004	4
0.05	24.5	0.005	4
0.10	42.3	0.006	5
0.20	63.0	0.006	5
0.50	92.0	0.008	7
1.00	99.7	0.012	9

Table 3. Performance with dampening.

μ	% fails	CPU (s)	nl-steps
0.01	0.0	0.005	4
0.05	0.1	0.006	4
0.10	0.0	0.006	5
0.20	0.7	0.008	8
0.50	11.6	0.011	12
1.00	49.8	0.019	23

Table 4. Performance of combined improved initial guesses and dampening.

μ	% fails	CPU (s)	nl-steps
0.01	0.5	0.004	3
0.05	0.3	0.004	3
0.10	0.3	0.005	4
0.20	3.5	0.007	5
0.50	47.7	0.013	13
1.00	96.0	0.044	40

For Table 3 the same problems were run again but now with the dampening scheme described above. This method copes much better with the large perturbations, only approaching 50% failure rate for the most extreme perturbation ($\mu = 1$). This robustness comes at the cost of more Newton iterations, and consequently longer computation times. However, the increase is not dramatic, being around a doubling for the most challenging cases.

In Table 4 we see the performance of combining dampening with the technique to improve the initial guess as described above. The basic idea here was to exploit the fact that we are solving a problem that is close to some known related problem to which we know the solution. Comparing with the failure rates of Table 3, we can conclude that this approach is not sound when the two problems are in fact not close. However, for small perturbations ($\phi \leq 0.1$) we see a typical reduction of one Newton iteration. The method requires an extra linear solve, so the overall time consumption is only moderately reduced for these cases (third column).

Finally, in Table 5 we see the performance using the continuation method instead of dampening. This achieved the best results so far in terms of failure rate, but it comes with a hefty computational price, approaching the times used by the ODE solver for very large perturbations. For small perturbations we observe a substantial reduction in computation time.

Figure 2 compares the failure rates of the three main variants graphically. Clearly, a lot is gained by applying dampening. The continuation was, despite excellent convergence properties in general, not competitive and so left out of the following plots (but see Section 3.3 below). Figure 3 compares the number of Newton iterations required by each method and shows the superior performance of the combined approach on the moderately hard problems ($\mu \leq 0.1$). This is echoed in Figure 4 which compares the CPU times.

Table 5. Performance using the continuation method instead of dampening. The last column (# steps) shows how many separate Newton problems are solved for each case.

μ	% fails	CPU (s)	nl-steps	# steps
0.01	0.0	0.004	4	1
0.05	0.0	0.006	4	1
0.10	0.0	0.007	5	1
0.20	0.1	0.233	110	3
0.50	0.7	0.817	519	11
1.00	8.0	1.558	856	19

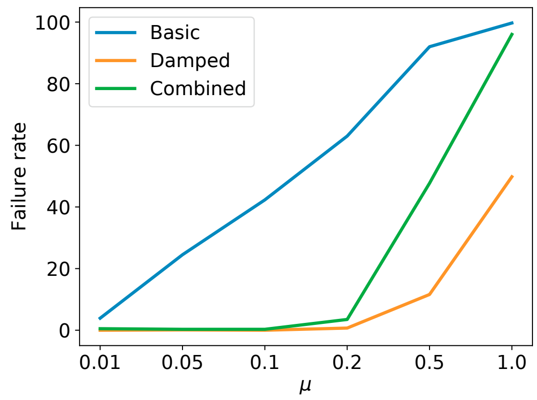

Fig. 2. Comparison of failure rates for the undamped Newton (Basic), damped Newton (Damped) and the method using improved initial guess combined with damping (Combined).

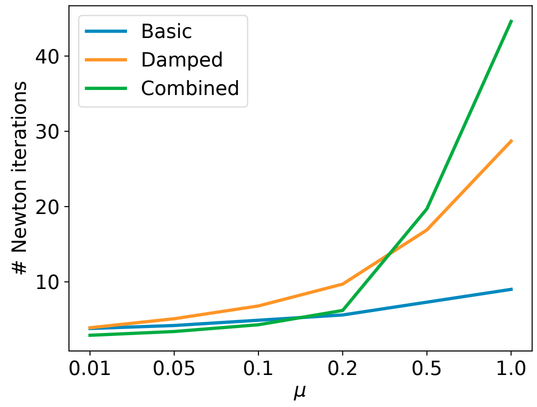

Fig. 3. Comparison of the number of Newton iteration used for the different methods.

3.2 Performance along a given parameter trajectory

To assess the performance in a setting better reflecting the requirements for parameter estimation, the parameter changes in this section are taken from an actual optimiza-

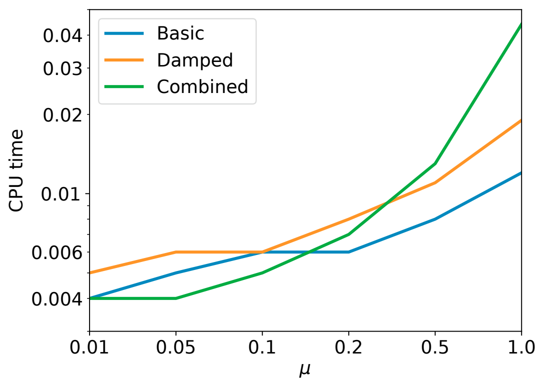

Fig. 4. Solve times for the different methods, mean over all 94 conditions.

Table 6. Performance of the dampened Newton method along a given parameter trajectory from optimization using Ipopt from Schmiester et al. (2019). For the first initial guess the ODE problem was solved to steady state, in subsequent steps the previous solution was used as the initial guess for the new problem. Summary of 150 steps with 94 conditions each, totalling to 14100 problems. Mean solve time reported.

% fails	CPU (s)	nl steps
0.46	0.004272	3.2

tion run (Schmiester et al., 2019). The search consisted of 150 steps, and again 94 conditions, yielding a total of 14100 problems to be solved. As above, we computed the first initial guess by solving the ODE problem to steady state. In subsequent steps the previous solution was used as the initial guess for the new problem. Overall, the steps were quite small and we decided to not use the more robust but slower approaches for this problem, but instead rely on the damped-Newton method. Table 6 shows the results. The method found the solution in most cases, and it did so efficiently and with few iterations.

3.3 Computing the initial steady state

As seen above, Newton's methods works very well if a good initial guess is provided. The bottleneck is to compute the first initial guess. For this we have relied on solving the ODE problem until steady state. When using the multi start method in the parameter optimization this computation must be done for each run. Here we report on the use of the continuation method as an alternative.

A total of 16 cases were considered. One of these was picked as the base case and the steady state was computed by solving the ODE problem in the usual way. Two approaches to compute the remaining 15 steady states were then compared: the ODE approach and the continuation method. Table 7 shows the performance. Clearly, the continuation method is superior.

4. DISCUSSION AND CONCLUSION

We have seen that computing the steady state solutions of large-scale ODE models by solving the corresponding equilibrium equation $F(x) = 0$ can be done quite successfully using Newton's method. This is not an obvious result and can not be expected to hold in general. However, for the given model structure and over a large set of parameter values we observed quite a robust performance, especially with the suggested improvements.

Table 7. Performance of the continuation method. Numbers are collected from 15 cases, where each case consists of 94 conditions. Last column shows the CPU time spent using ODE solves.

	# sub problems	# steps	CPU time	ODE
Mean	2.2	63.8	0.0478	65.8
Median	1	42	0.030	3.04
Max	21	340	0.263	2474
Min	1	9	0.00616	1.52

Our study of one of the largest published ODE models revealed that the method is especially well suited in connection with parameter estimation where we have a series of related problems, only separated by some moderately sized step $\Delta\phi$ in the parameter space. In this case, a good initial guess can be obtained by simply using the previous steady state or by computing an improved initial guess as described in Section 2.5. Along a typical parameter trajectory only 2-3 Newton iterations were sufficient in this case.

The CPU time for all considered approaches is dominated by the linear solves. Accordingly, the approaches share the scaling behaviour of the linear solver. While numerical integration for the ODE has the same scaling properties, the required number of linear solves is usually much higher and so is the computation time.

The biggest challenge was to arrive at the initial steady state. For this, one can rely on the fallback method, i.e. to solve the ODE system, but we found that the method of continuation was very useful here.

A drawback with not solving the ODE problem is that the computation of the gradient of the cost function might become a lot more costly. When we just solve the problem as an algebraic system of equations, techniques such as adjoint sensitivity analysis are not readily available. However, an extension of the method to obtain gradients also in this case is conceivable and is something that we will pursue in future work.

ACKNOWLEDGEMENTS

This work was supported by the European Union's Horizon 2020 research and innovation program (CanPathPro; Grant No. 686282) and the German Research Foundation (DFG) (Grant No. HA7376/1-1).

REFERENCES

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

Bouhaddou, M., Barrette, A.M., Stern, A.D., Koch, R.J., DiStefano, M.S., Riesel, E.A., Santos, L.C., Tan, A.L., Mertz, A.E., and Birtwistle, M.R. (2018). 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.

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

Davis, T.A. and Natarajan, E.P. (2010). Algorithm 907: Klu, a direct sparse solver for circuit simulation problems. *ACM Trans. Math. Softw.*, 37(36), 1–36.

Fiedler, A., Raeth, S., Theis, F.J., Hausser, A., and Hasenauer, J. (2016). Tailored parameter optimization methods for ordinary differential equation models with steady-state constraints. *BMC Syst. Biol.*, 10(80). doi: 10.1186/s12918-016-0319-7.

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

Hass, H., Loos, C., Raimúndez-Álvarez, E., Timmer, J., Hasenauer, J., and Kreutz, C. (2019). Benchmark problems for dynamic modeling of intracellular processes. *Bioinformatics*, btz020.

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

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 Transactions on Mathematical Software (TOMS)*, 31(3), 363–396.

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

King, E.L. and Altman, C. (1956). A schematic method of deriving the rate laws for enzyme-catalyzed reactions. *J. Phys. Chem.*, 60(10), 1375–1378.

Klipp, E., Nordlander, B., Krüger, R., Gennemark, P., and Hohmann, S. (2005). Integrative model of the response of yeast to osmotic shock. *Nat. Biotechnol.*, 23(8), 975–982.

Rosenblatt, M., Timmer, J., and Kaschek, D. (2016). Customized steady-state constraints for parameter estimation in non-linear ordinary differential equation models. *Front. Cell Dev Biol.*, 4(5).

Schmiester, L., Schaelte, Y., Froehlich, F., Hasenauer, J., and Weindl, D. (2019). Efficient parameterization of large-scale dynamic models based on relative measurements. *bioRxiv*. doi:10.1101/579045.

Soetaert, K. (2009). *rootSolve: Nonlinear root finding, equilibrium and steady-state analysis of ordinary differential equations.* R package 1.6.

Van der Vorst, H.A. (1992). Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2), 631644.

Villaverde, A.F., Froehlich, 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.

Yang, W., Soares, J., Greninger, P., Edelman, E.J., Lightfoot, H., Forbes, S., Bindal, N., Beare, D., Smith, J.A., Thompson, I.R., Ramaswamy, S., Futreal, P.A., Haber, D.A., Stratton, M.R., Benes, C., McDermott, U., and Garnett, M.J. (2013). Genomics of Drug Sensitivity in Cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells. *Nucl. Acids Res.*, 41(Database issue), D955–D961.

Appendix A. THE CONTINUATION ALGORITHM

Below is pseudo code for the continuation algorithm. The `Newton` function takes as input a parameter vector and an initial guess and returns the solution of the equilibrium problem if it converges, which is indicated with the second return argument.

```
// struct to hold latest parameter
// with known steady-state:
memory{P,x} := (P0, x0)

done = False
P := P1 // a new dynamic parameter
do
  [x, converged] := Newton(P, memory.x)

  if !converged then
    stack.put(P)
    P := mean(P, memory.P, gamma)
    done := ||stack.top() - memory.P|| < tol

  else if stack.size() > 0 then
    memory{P,x} := (P, x)
    P := stack.pop()

  else
    done = True

while !done

if stack.size() > 0 then
  Failed to find a steady-state for P1
```