Dynamical Modeling and Multi-Experiment Fitting with PottersWheel – Supplement

Thomas Maiwald and Jens Timmer
Freiburg Center for Data Analysis and Modeling
University of Freiburg, Germany
maiwald@fdm.uni-freiburg.de

June 10, 2008

Abstract

This supplement provides detailed information about the functionalities of the PottersWheel toolbox as described in the main text. For further information please use the documentation which is available at www.PottersWheel.de.

Contents

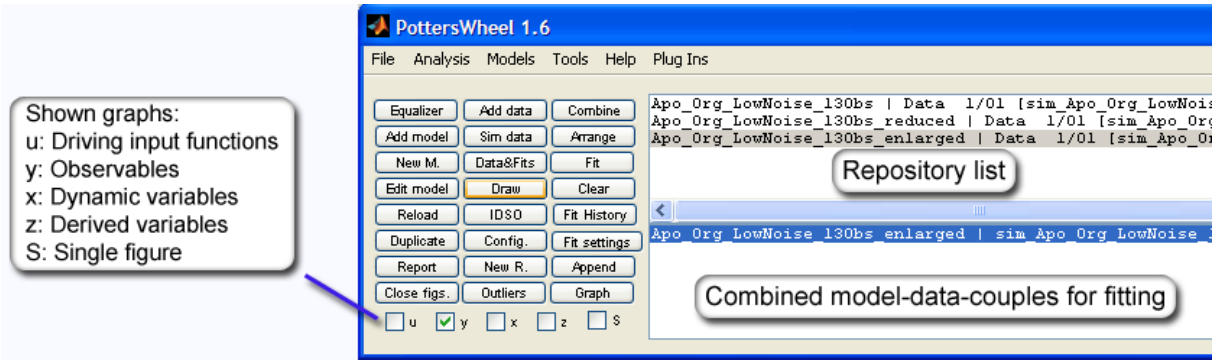

Figure 1: **The PottersWheel main user interface.**

1 The main graphical user interfaces

All key functionalities of PottersWheel are available by command line and via 30 graphical user interfaces. The mostly used interfaces are the main window comprising a list of available models and data sets, and the equalizer including sliders to change model parameters in real time. Both interfaces share a common set of keyboard shortcuts.

1.1 Main window

Fig. 1 displays the PottersWheel main user interface. Below the menu, 24 buttons provide direct access to the most important functions concerning model, data, and fits handling and to open other interfaces like the equalizer, the outliers dialog, or the report designer. Checkboxes control the set of shown graphs: Driving input functions, observables, dynamic and/or derived variables.

Two list boxes represent the current working state. The upper list, also called the *repository list*, displays all loaded models. To each model, one or more simulated or external data sets may be attached. The currently active data set can be changed in the 'Data & Fits' dialog (see button in second column). In order to fit a model to a data set, they have to be *combined* to the lower list box by pressing the 'Combine' button. For multi-experiment fitting, several model-data-couples can be selected before they are combined. The models can be fitted preferably after automatic arranging of all figures. Alternatively, the equalizer can be started to select a certain optimizer and parameter space before fitting or to manually adjust parameter values.

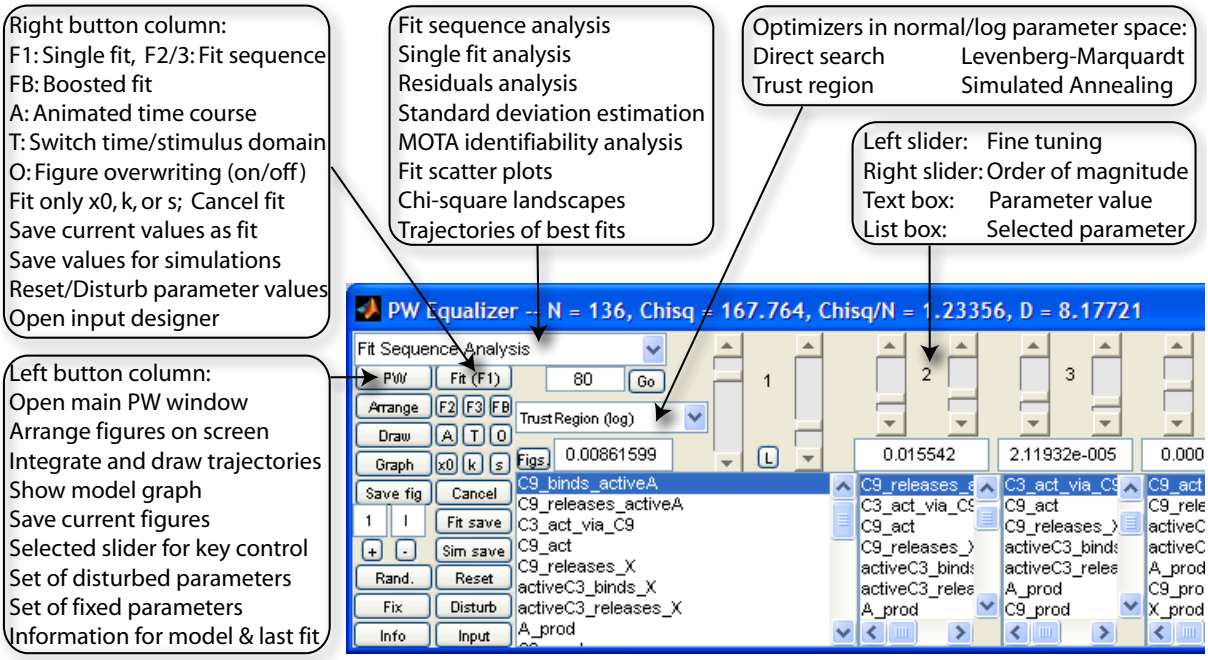

Figure 2: **The PottersWheel Equalizer.**

1.2 Equalizer

The PottersWheel Equalizer (Fig. 2) comprises 10 pairs of sliders. Each pair can be attached to one of the fitted parameters in the list box below the slider pair. The left slider is used for fine-tuning and the right slider changes the magnitude of the selected parameter. Text boxes on the one hand reflect the current parameter value and can on the other hand be used directly to specify a certain value.

The chosen optimizer (direct search, trust region, Levenberg-Marquard, or simulated annealing, see section 4) can be selected in normal or logarithmic parameter space. The 'Go' button starts an analysis, as in the example of Fig. 2 a fit sequence analysis based on 80% of the best fits. 'F1' fits the data set one time. 'F2' and 'F3' apply a fit sequence (see section 7.1), and 'FB' applies a hybrid fitting including a deterministic and a stochastic optimizer (see section 4.2). 'A' opens an animation of the time course, 'T' switches between time-domain and stimulus domain, and 'O' turns overwriting of figures on and off which is useful to compare trajectories for different parameter settings in one figure. The buttons 'x0', 'k', and 's' fit only start values x_0, dynamical parameters k or scaling parameters s instead of all parameters as specified in the fit settings. 'Input' starts the driving input designer where the effect of new experimental settings, e.g. pulsed or ramp stimulations can be investigated.

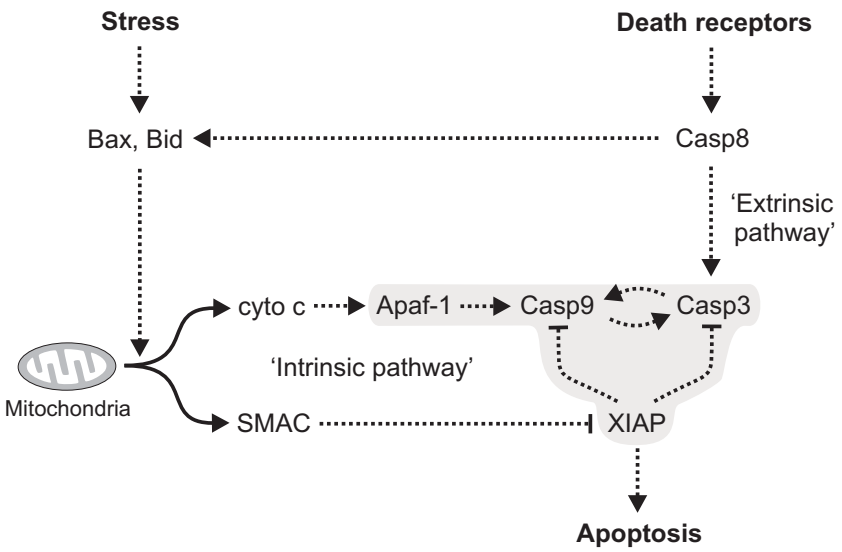

Figure 3: **The apoptotic signaling pathway.** Legewie et al. developed a detailed mathematical model for the gray colored subsystem of the apoptotic signaling pathway, which we use to demonstrate the functionalities of PottersWheel. We included the effect of cyto-c and SMAC as externally given driving input functions, in order to demonstrate the increased statistical power when multi-experiment fitting is applied under different experimental conditions corresponding to different characteristics of the cyto-c and SMAC time courses (Picture by S. Legewie).

2 Creating an apoptosis example model

A realistic medium sized model with 13 species, 41 reactions, and 13 kinetic parameters serves as an example to demonstrate the functionalities of PottersWheel. The model has been suggested by Legewie *et al.* (2006) and describes the feedback control of caspase-3 and caspase-9 in the intrinsic apoptosis signaling pathway (see Fig. 3). In order to investigate the increased statistical power to discriminate competing model hypothesis and to calibrate unknown parameters, we extended the model by two external driving functions representing an externally specified concentration of cyto-c and SMAC.

2.1 The model definition file

The systems biology markup language (SBML) has been designed to enable a standardized way to express, store, and exchange kinetic models based on reaction networks (Finney and Hucka, 2003). The supplement of Legewie et al. contains an SBML 2 level 1 file with all species and reactions of the system. Via the PottersWheel GUI or `pwImportSBML` the model can be imported into a PW model definition file, which is constructed as a Matlab function and contains the reaction network, initial values for the system species, parameter values, algebraic equations, observables, and driving input functions. In order to take advantage from the PottersWheel naming conventions (compare section 10.4), we renamed the original species. The sections of the resulting file are displayed in excerpts in the following. The PottersWheel documentation at www.PottersWheel.de contains a detailed description of model definition files and the utilized help functions like `pwAddR`

for adding of a reaction. The complete model is available on the same web-site.

2.1.1 Header

Since a PottersWheel model definition file is constructed as a MATLAB function, it has to start with the *function* keyword. Afterwards an empty model m is created which will be filled in the following paragraphs.

```
function m = Legewie_Apoptosis()

m = pwGetEmptyModel();
```

2.1.2 Dynamic variables

The dynamic variables section specifies the initial value for all species wich are non-zero at $t = 0$. Other species are collected automatically from the reactions paragraph.

```
% m=pwAddX(m, ID, startValue)
m=pwAddX(m, 'activeA', 20);
m=pwAddX(m, 'C9',      20);
```

2.1.3 Reactions

For each reaction reactants, products, modifiers such as enzymes, parameters, and the reaction kinetics are specified. In the rate signature, r_i, p_i, m_i, and k_i are placeholders for the i-th reactant, product, modifier, and parameter of the current reaction, respectively. This way, the rate signature does not change from reaction to reaction as long as the underlying kinetics is the same. Reactions 1 and 2 read:

```
%pwAddR(m, reactants, products, modifiers,type,options,rateSignature,params)
m=pwAddR(m,{'C9','X'}, {'C9_X'},   {}, 'C', [], 'k1*r1*r2', {'C9_binds_X'});
m=pwAddR(m,{},         {'activeA'},{}, 'C', [], 'k1',       {'A_prod'});
```

2.1.4 Dynamic parameters

All dynamic parameters have to be specified if their value or initial guess for fitting is known. Otherwise, they will be set to the default value of 0.1.

```
% m=pwAddK(m,ID, value)
m=pwAddK(m,'C9_binds_activeA',       0.002);
m=pwAddK(m,'C9_releases_activeA',    0.1);
```

2.1.5 Algebraic equations (rules)

The apoptosis model contains three algebraic equations, where three parameters have a fixed relationship to other parameters. This can be expressed by a *rule* (see section 10.2):

```
% m=pwAddRule(m,lhs, reactants, parameters, ruleSignature)
m=pwAddRule(m,'C3_act_via_activeA_C9',{},{'C3_act_via_C9'},'70*k1');
```

2.1.6 Observables

Observables are the interface between a mathematical model and experimental measurements. Usually, the dynamical system is only partially observed and some species are only measurable as a sum. For example Western blotting measurements do not distinguish between free and XIAP bound caspase-3, since the complexes are denatured before quantification. Hence, the sum 'C3 + C3_X' is measured as 'C3_obs'. A default error model can be specified in order to estimate the standard deviation of measurements (see section 11.2). We here use 10% relative and 10% absolute error (relative to the maximum of the measured species).

```
% m=pwAddY(m,rhs, ID, scalingParameter, errorModel)
m=pwAddY(m,'activeA' ,'activeA_obs','scale_activeA_obs','0.1*y+0.1*max(y)');
m=pwAddY(m,'C3+C3\_X','C3_obs',     'scale_C3_obs',     '0.1*y+0.1*max(y)');
```

2.1.7 Driving input functions

In order to *drive* the dynamical system, input functions are required representing the experimental setting. In the below example, the CytoC concentration is characterized as a step function jumping at time point 0 to a value of 2. The SMAC concentration on the other hand is given as a pulsed stimulation: At time point 0 the concentration jumps to 2 and after 10 minutes it returns to 0. Combination of different experimental settings, i.e. different driving input functions, is a crucial aspect in multi-experiment fitting (see sections 5 and 6).

```
% m = pwAddU(m, ID, uType, uTimes, uValues)
m = pwAddU(m, 'CytoC', 'steps', [-1 0],    [0 2]);
m = pwAddU(m, 'SMAC',  'steps', [-1 0 10], [0 2 0]);
```

2.2 Graphical visualization of the reaction network

The chemical reaction network is automatically visualized using the *graphviz* engine (Gansner and North, 2000). Reaction types are color coded. In large networks, sub-systems comprising all reactions related to a specific species can be displayed to allow for error checking.

Fig. 4 illustrates the reaction network of the apoptosis model. The external driving input functions, cyto-c and SMAC, are drawn as yellow octagons. Other species are blue ellipses. Blue arrows correspond to reactions with not more than one reactant and one product, whereas green arrows reflect multi-molecular reactions. Black arrows represent enzymatically triggered reactions, where the effect of the enzyme, i.e. modifier is dashed in red.

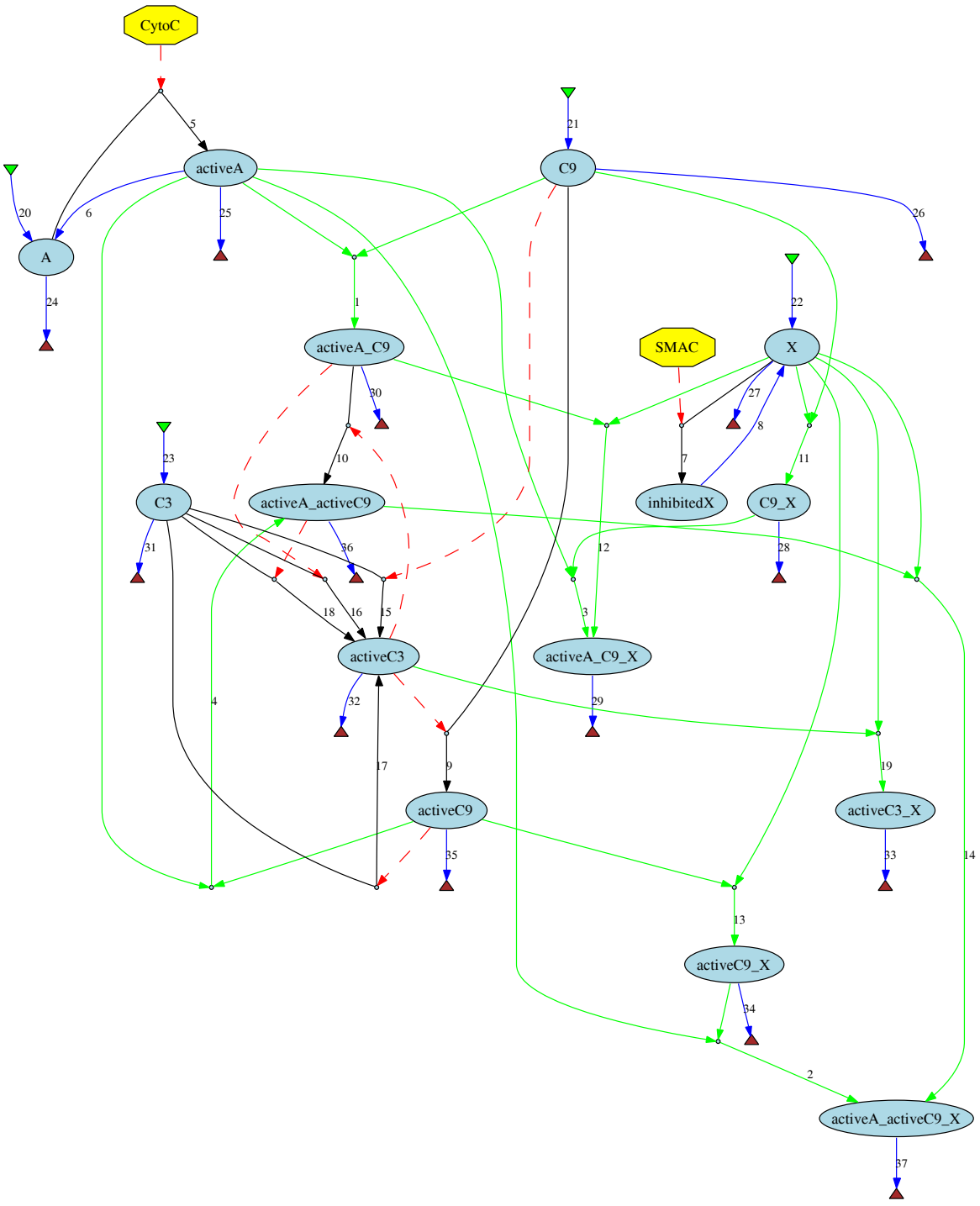

Figure 4: **Automatic model visualization.** In the reaction network of the blue colored system species, green and brown triangles represent their sources and sinks, respectively. Blue arrows correspond to reactions with not more than one reactant and one product, whereas green arrows reflect multi-molecular reactions. Black arrows represent enzyme-catalyzed reactions, where the effect of the enzyme, i.e., the modifier, is indicated by a red dashed line. Species that are manipulated by experimental conditions (e.g. by extracellular stimulation), so called driving inputs, are displayed as yellow octagons.

3 Integration performance

During parameter calibration, the model trajectories have to be calculated thousands of times until an optimal parameter setting is found. Hence, high integration speed is a crucial prerequisite for interactive dynamical modeling of experimental data. PottersWheel applies the following strategy to meet these requirements:

1. Use of fast and accurate FORTRAN integrators.
2. The differential equations are saved and compiled as C MEX files. If possible, the integrator, interface, and model are compiled into a single executable.
3. The merit function of the optimizer is dynamically generated containing no overhead or slow MATLAB functions like *eval* statements.
4. Calculation of observables or residuals is also based on dynamical C MEX files.

In the following, we describe the integrators and compare their performance with MATLAB built-in integrators, which can used interchangeably within PottersWheel. A total number of 13 integrators is currently available. Using the function `pwCompareIntegrators`, the user can determine the most appropriate one for the modeling problem at hand.

3.1 Stiff differential equations

Dynamical systems including many chemical reaction networks may be *stiff*, i.e. the time-scales of the variables can have huge differences in their range. Not all integrators can handle this situation. An indication of stiffness are parameter values of different order of magnitude. Within PottersWheel, the function `pwCheckForStiffness` detects this property. Even if the differential equations of the model are not stiff for the initial or fitted parameter values, during parameter calibration regions of the parameter space where the system behaves stiff may be crossed.

3.2 FORTRAN integrators

Currently, six FORTRAN integrators are supported by PottersWheel, being described in Hairer and Wanner (1996). We use the MATLAB interface of Ludwig (2006), which we extended in two cases to reduce overhead by circumventing calls between integrator and the model equations. Our modification improves the integration time by an additional factor of 10-35 and requires either FORTRAN compilers for Linux/Mac or the lcc compiler for Windows computers. The integrators are:

1. RADAU5: Implicit Runge-Kutta method of order 5 with dense output.
2. RADAU: Implicit Runge-Kutta method of variable order, switching automatically between orders 5, 9, and 13.
3. SEULEX: Extrapolation method based on linearly implicit Euler method.
4. DOP853: Explicit Runge-Kutta method of order 8(5,3) with dense output of order 7.

5. DOPRI5: Explicit Runge-Kutta method of order 5(4) with dense output of order 4.
6. ODEX: Extrapolation method (GBS) with dense output.

Integrators 1-3 are suitable for stiff problems.

3.3 MATLAB integrators

All MATLAB integrators can be used within PottersWheel:

1. ode45: Explicit Runge-Kutta (4,5), Dormand-Price. One-step solver requiring only the solution at the preceding time point (Dormand and Prince, 1980).
2. ode15s: Variable order solver based on the numerical differentiation formulas (NDFs). Multi-step solver (Shampine and Reichelt, 1997; Shampine *et al.*, 1999).
3. ode23: Explicit Runge-Kutta (2,3), Bogacki and Shampine. One-step solver (Bogacki and Shampine, 1989).
4. ode23s: Modified Rosenbrock of order 2. One-step solver (Shampine and Reichelt, 1997).
5. ode23t: Implementation of the trapezoidal rule (Shampine *et al.*, 1999).
6. ode23tb: Implementation of TR-BDF2, an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order 2 (Shampine and Hosea, 1996; R. E. Bankand W. C. Coughran *et al.*, 1985).
7. ode113: Variable order Adams-Bashforth-Moulton PECE solver. Multi-step solver (Shampine and Gordon, 1975).

Integrators ode15s, ode23s and ode23tb are applicable to stiff problems.

3.4 Dynamical compilation of ODE as C MEX file

The right hand side of the differential equations including algebraic equations, interpolation formulas and events is saved and compiled as a C MEX file when a model is loaded into PottersWheel. For illustration purpose and to compare the numerical performance, the model can also be saved as a MATLAB file. For the example model, calling the C MEX file is 20 times faster than calling the MATLAB function.

3.5 Comparing integration time and accuracy

We compare the 13 integrators on the basis of the medium sized apoptosis example model comprising 13 species and 41 reactions. Their performance was determined by four criteria:

1. Total integration time

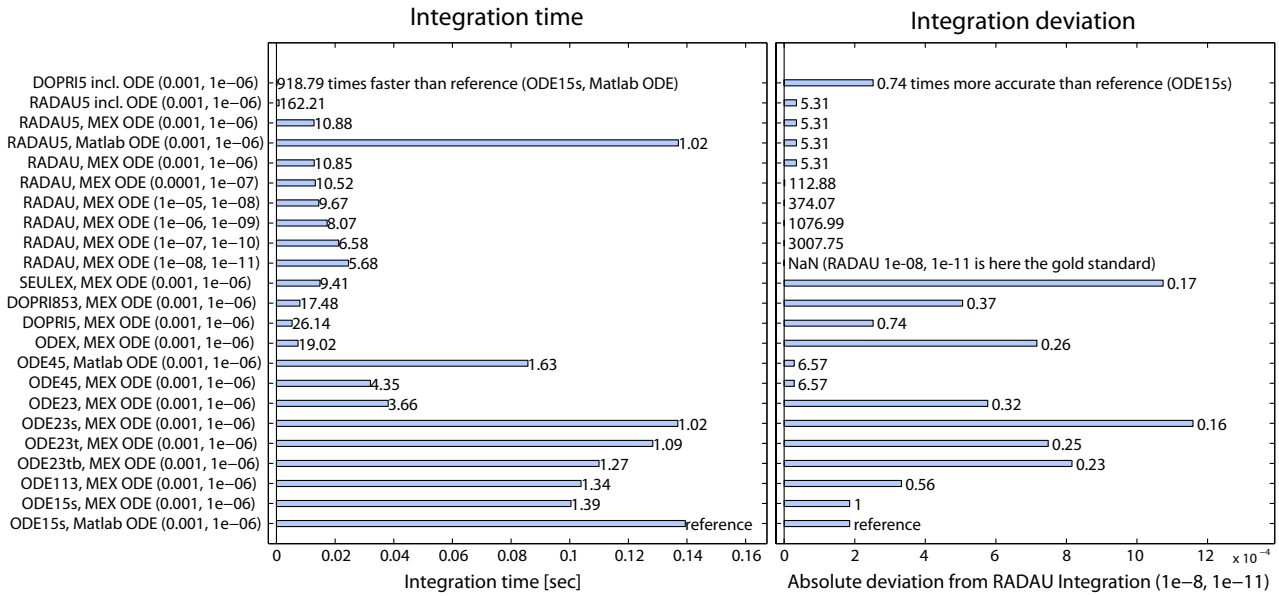

Figure 5: **Integration time and accuracy.** Left: The mean integration time of all 13 supported integrators with specified relative and absolute tolerance is displayed. The integrator either includes the ODE (rows 1 and 2) or is attached to an ODE compiled as C MEX file or saved as a normal MATLAB function. The reference is the integration time using MATLAB integrator ODE15s for stiff systems with a MATLAB ODE (ca. 0.1 seconds, last row). DOPRI5 for non-stiff systems (first row) is 919 times faster if the ODE is included, else 26 times faster using a MEX ODE (row 13). RADAU5, a stiff integrator, is 162 times faster (second row) with included ODE and 11 times faster with a MEX ODE compared to the reference. Calling RADAU with increasing integration accuracy leads to a slightly longer integration time. Right: Integration with RADAU using high tolerances of 10^{-8} and 10^{-11} serves here as a gold standard to estimate the accuracy of all integrators by quantifying the mean deviation between the calculated trajectories. RADAU5 $(10^{-3}, 10^{-6})$(row 2) is not only faster than ODE15s, but also 5 times more accurate.

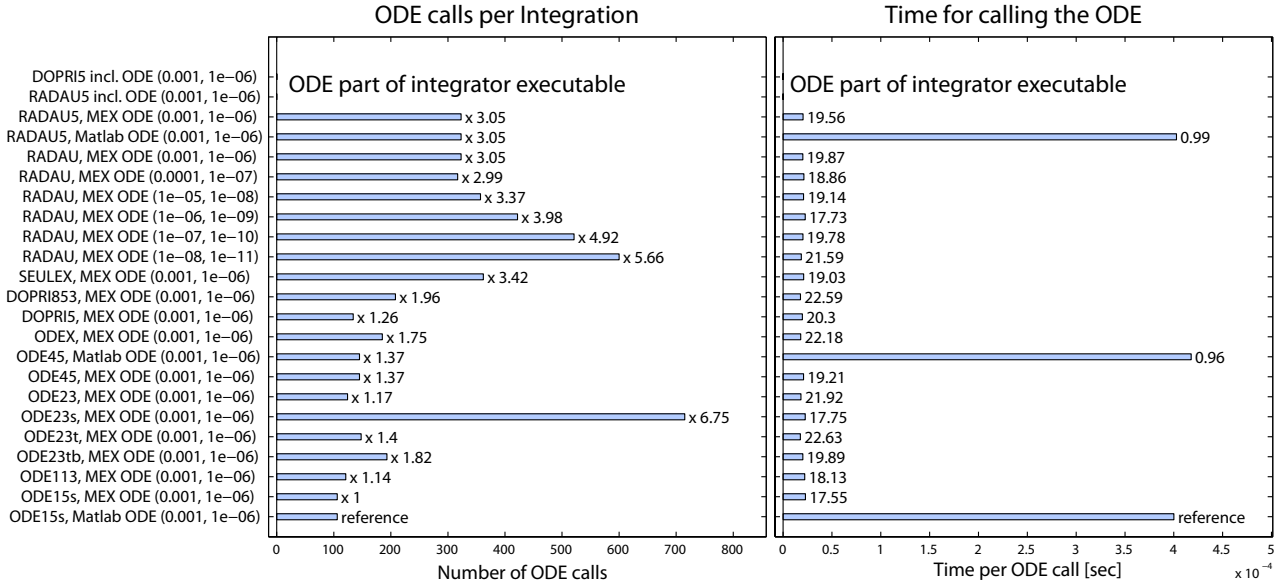

Figure 6: **ODE calls per integration.** Left: RADAU5 with external ODE file calls the ODE 3 times more often than the reference integrator, ODE15s, leading to a 5 times higher accuracy (see Fig. 5). The fast versions DOPRI5 and RADAU5 including the ODE have not an external call to the right hand side. Right: Using a MATLAB function for the ODE takes approximately 20 times more time than a compiled C MEX ODE.

2. Number of calls of the right hand side of the ODE system
3. Time per call of the right hand side
4. Accuracy, measured as the averaged absolute distance of the integrated trajectory to a highly accurate integration with RADAU with 10^{-11} absolute and 10^{-8} relative tolerance.

The relative and absolute tolerances of the integration are usually set to 10^{-6} and 10^{-3} respectively. Only the RADAU integrator is tested with a variety of tolerances, on the one hand to illustrate the effect on integration time and on number of calls to the ODE and on the other hand to serve here as a gold standard for the estimation of the integration accuracy. The integrations were applied on a Macintosh laptop with Intel Core 2 Duo 2.4 GHz with 2 GB RAM.

In summary, a compiled DOPRI5 including the ODE is approximately 900 times faster than using the reference MATLAB ode15s with an ODE as MATLAB or MEX file. If the ODE is not compiled into the integrator executable, DOPRI5 is still 26 times faster. Using RADAU5 which is also applicable to stiff systems, reaches an integration time 160 (incl. ODE) or 11 (attached ODE) times smaller than for the reference. Simultaneously, RADAU5 reaches a 5 times smaller deviation than the reference compared to the trajectory of the gold standard.

4 Optimization performance

The χ^2 merit function which is optimized within PottersWheel to fit the model $y = y(t; \mathbf{p})$ is

$$\chi^2(\mathbf{p}) = \sum_{i=1}^{N} \left(\frac{y_i - y(t_i; \mathbf{p})}{\sigma_i} \right)^2, \tag{1}$$

with y_i being data point i with standard deviation σ_i and $y(t_i; \mathbf{p})$ being the model value at time point i for parameter values $\mathbf{p}$. This setting belongs to the group of non-linear least square problems. If the measurement errors are normally distributed, the minimization of the weighted least-square error corresponds to applying a Maximum Likelihood estimator for the unknown parameters (see section 9.1). The PottersWheel documentation provides a short introduction into numerical optimization based on the book of Nocedal and Wright (1999). The χ^2 value is under the null hypothesis of a data compliant model with identifiable parameters distributed like a χ^2 distribution with $f = N - n_p$ degrees of freedom. Unidentifiable parameters increase the degrees of freedom. The expectation value of a χ^2 distribution equals to its degrees of freedom. Therefore, we display the normalized value χ^2/N. A value larger than 1 indicates that the model with the current paramter values can not explain the experimental data sufficiently. Section 9 discusses in more detail how statistical tests can be applied to quantify the model validity.

Currently, five implementations of optimizers are available within PottersWheel: Direct search, trust region, Levenberg-Marquardt, genetic algorithm, and simulated annealing. The direct search method is only useful for illustration purposes or small models. The trust region and Levenberg-Marquardt algorithms are powerful deterministic least-square optimizers. The simulated annealing algorithm is as a stochastic approach able to handle local minima, but requires more time. We analyse the performance of the genetic algorithm to demonstrate that it is not suitable for model fitting. We quantify the accuracy of the optimization by the average deviation D of the n_p fitted parameters to the true parameters:

$$D = \frac{1}{n_p} \sum_{i=1}^{n_p} \max \left(\frac{p^i_{fit}}{p^i_{true}}, \frac{p^i_{true}}{p^i_{fit}} \right) \tag{2}$$

A value of 1 indicates a perfect fit. A higher value indicates that on average the fitted value is D times higher than the true one or visa versa. As the next subsections demonstrate, only the trust region and Levenberg-Marquardt algorithms are able to calibrate the parameters of an identifiable apoptosis model with 13 observables and low noise until the parameter deviation is below 10%. For this, the Levenberg-Marquardt routine requires $\sim$ 600 function calls and the trust region approach 2200. The direct search method has, as an unconstrained optimizer, a very high deviation of 16,000,000% after 8000 function calls. Simulated annealing requires 10,000 function calls to reach a deviation of 1260%, which however is only based on the deviation of one parameter. The genetic algorithm results in a devia-

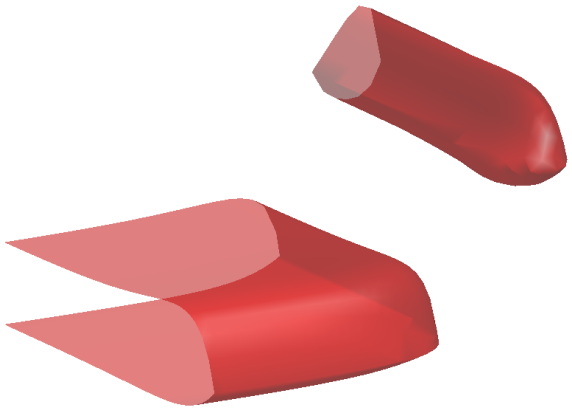

Figure 7: **Local minima.** Three parameters (C9_act, C9_releases_X, activeC3_binds_X) were changed systematically in logarithmic space. The figure shows all parameter combinations with the same χ^2/N value of 2.18. If the initial guess for a deterministic optimizer is set within the upper right manifold, the global optimum located in the lower left manifold can not be reached: a stochastic optimizer is required. The figure can be reproduced with `pwChiSquareMan`.

tion of 18800% after 40,000 function calls.

Before fitting, the parameters were set to the default value of 0.1 and the limits of possible parameter values were set to $10^7 p_{true}$ and $p_{true}/10^7$ for the trust region and Levenberg-Marquardt algorithms, to $100 p_{true}$ and $p_{true}/100$ for the direct search, to $1000 p_{true}$ and $p_{true}/1000$ for the direct search, and $400 p_{true}$ and $p_{true}/400$ for the simulated annealing. If the default value of 0.1 was not inside the permitted range of parameter values, it was shifted accordingly.

4.1 Fitting in logarithmic parameter space

If parameter limits extend several orders of magnitude, it is strongly recommended to fit in logarithmic parameter space, which was done in the performance analysis. This leads to a better representation of small parameter values `pwSetLogFitting`. In fact, the trust region and Levenberg-Marquardt methods were unable to locate a satisfying minimum in normal space.

4.2 Hybrid stochastic & deterministic approach

With `pwFitBoost`, the trust region and simulated annealing strategy in normal and logarithmic parameter space are combined consecutively in order to reduce local minima. This hybrid approach takes more time, but is superior for complicated optimization problems compared to a single optimization method. Fig. 7 illustrates a situation where a stochastic optimizer is required.

4.3 Direct search

The direct search method is the `fminsearch` algorithm as implemented in MATLAB. It uses the simplex search method of Lagarias *et al.* (1998), is not constrained, and does not use numerical or analytic gradients.

This optimizer does not exploit the structure of non-linear least square problems, e.g. only the squared sum over all weighted residuals is taken into account. In addition, no lower and upper parameter limits can be specified, making it difficult to control the optimization within reasonable borders to take into account a prior knowledge about the dynamic system. In our test scenario the direct search method actually increased on average the distance of the parameter values to the true ones, compared to the initial guess. Consequently, the direct search algorithm is not recommended.

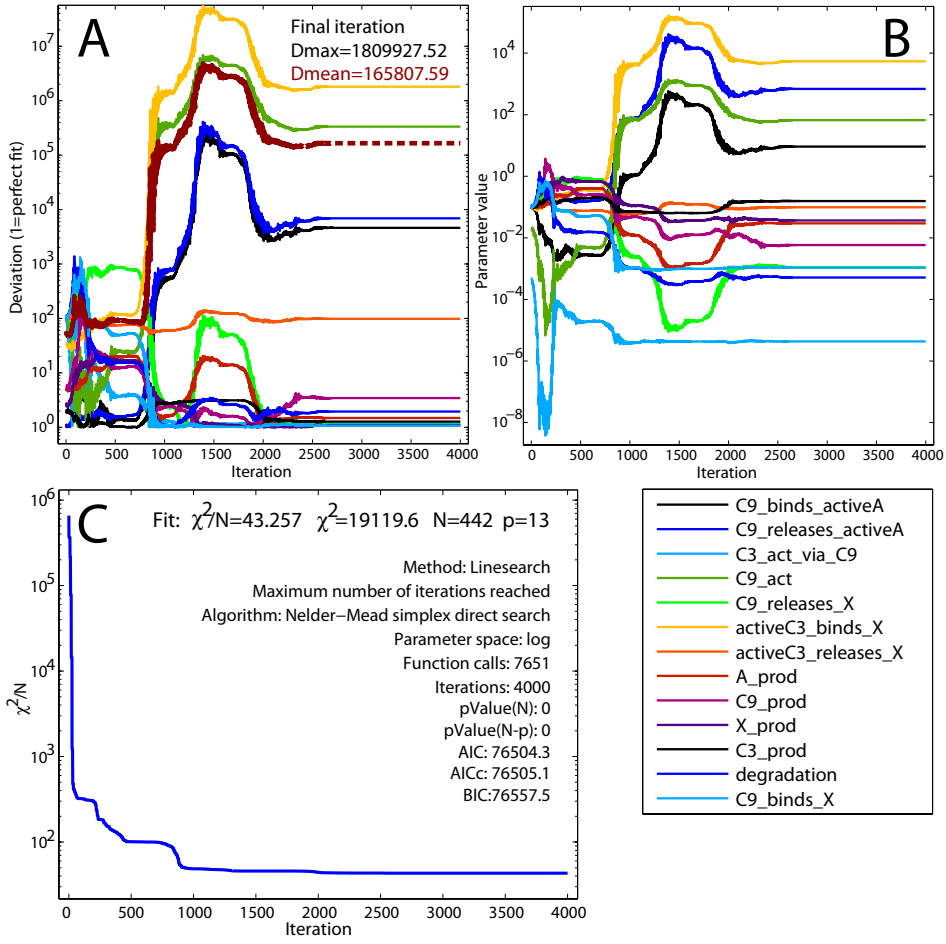

Figure 8: **Optimization performance of direct search.** A: Deviation compared to true parameters. The dashed red line displays the mean deviation over all parameters. B: Parameter values during fitting. C: χ^2 value during fitting and fit settings.

4.4 Trust region

The `lsqnonlin` algorithm of the MATLAB optimization toolbox is used as trust region approach. It is based on the interior-reflective Newton method described in Coleman and Li (1996) and Coleman and Verma (2001). Each iteration involves the approximate solution of a large linear system using the method of preconditioned conjugate gradients (PCG).

This optimizer exploits the structure of non-linear least square problems and simultaneously renders possible to specify limits for parameter values during calibration. It requires more function calls than the Levenberg-Marquardt optimizer, in the example case with a factor of 4. If the optimization toolbox is available and crucial limits for parameters exist, this is the method of choice.

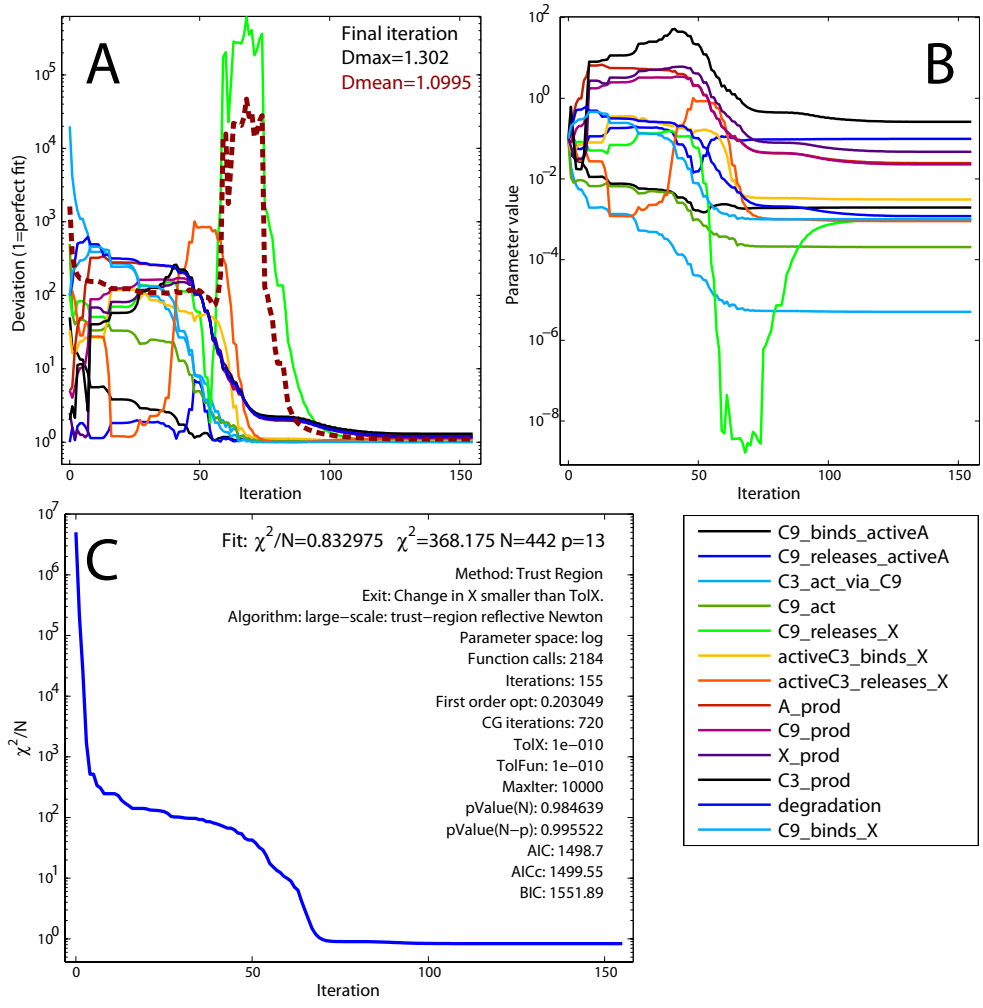

Figure 9: **Optimization performance of trust region.** A: Deviation compared to true parameters. The dashed red line displays the mean deviation over all parameters. B: Parameter values during fitting. C: χ^2 value during fitting and fit settings.

4.5 Levenberg-Marquardt

An unconstrained approximate Gauss-Newton approach with Levenberg-Marquardt damping and successive updating of Jacobian approximation is available in the immoptibox of Nielsen (2006). We modified the `smarquardt` algorithm slightly to fit into the PottersWheel framework.

This optimizer is very fast and accurate. The only drawback are missing limits for parameter values during calibration. If this is not important or the optimization toolbox is not available for using the trust region approach, this Levenberg-Marquardt implementation is recommended.

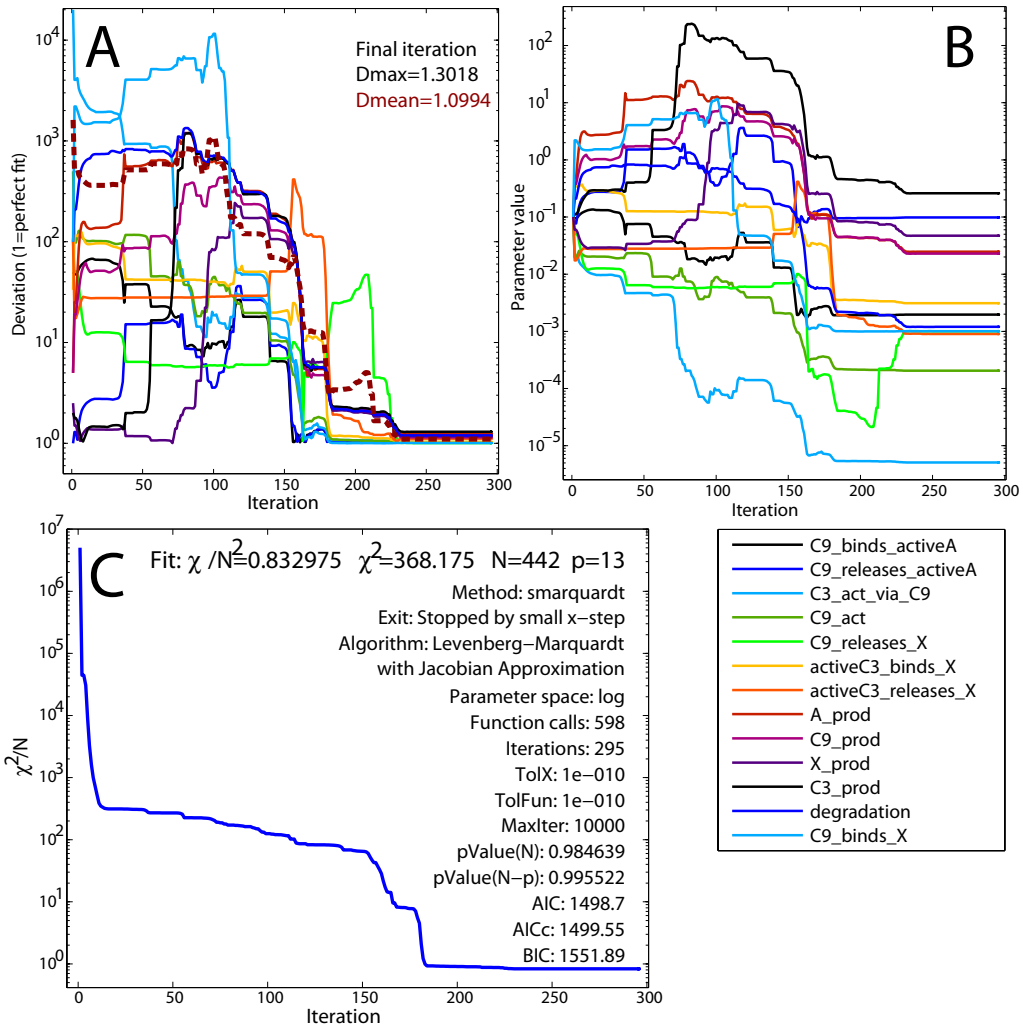

Figure 10: **Optimization performance of Levenberg-Marquardt.** A: Deviation compared to true parameters. The dashed red line displays the mean deviation over all parameters. B: Parameter values during fitting. C: χ^2 value during fitting and fit settings.

4.6 Simulated annealing

Ingber (1989) developed a fast simulated annealing algorithm. It is an algorithm to statistically find the best global fit of a nonlinear non-convex cost-function. The algorithm permits an annealing schedule for temperature T decreasing exponentially in annealing-time. We use the `asamin` MATLAB interface by Sakata (2001).

This optimizer generally copes better with local minima, but requires many function calls to locate the optimum. In the test scenario all but one parameter were calibrated accurately after 10,000 calls. It is recommended to use this method if a satisfying result can not be achieved with the trust region or Levenberg-Marquardt methods or as a cross-check.

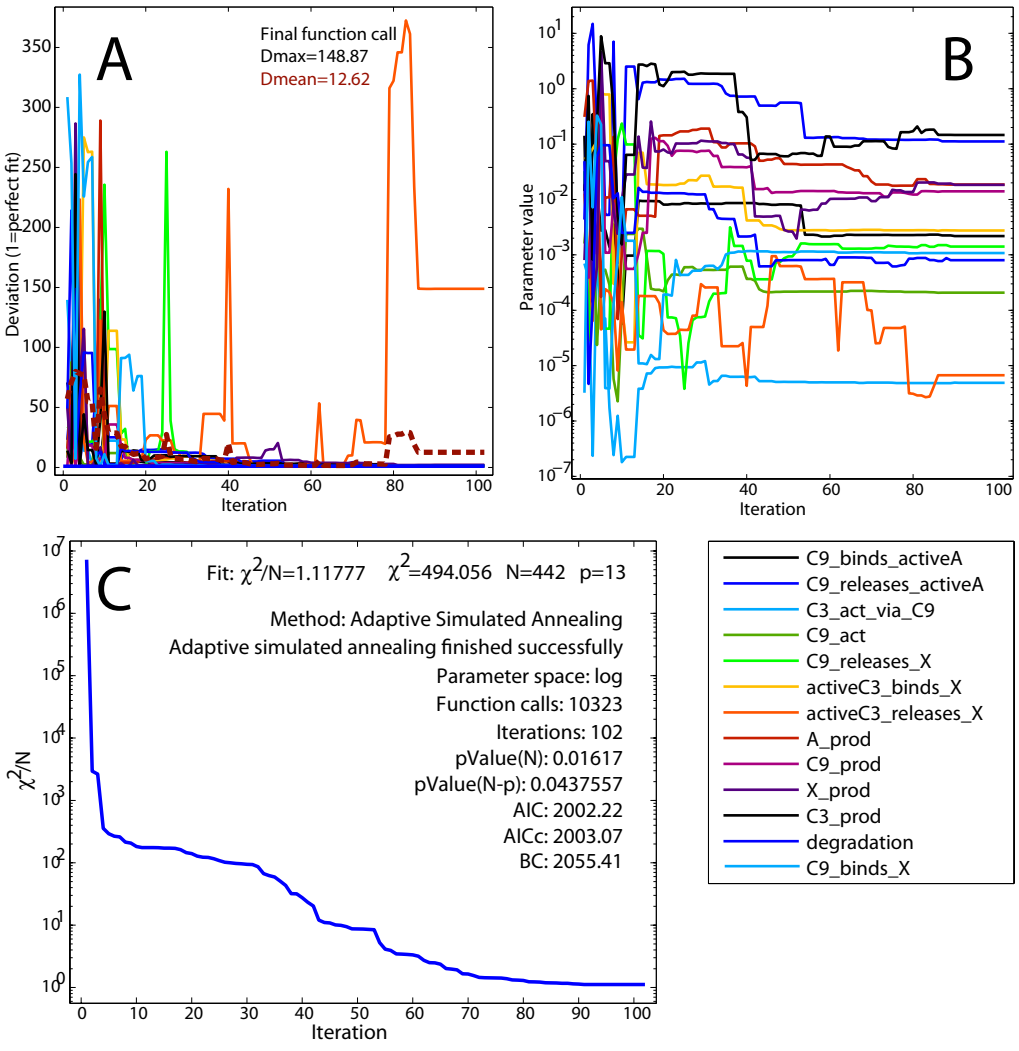

Figure 11: **Optimization performance of simulated annealing.** A: Deviation compared to true parameters. The dashed red line displays the mean deviation over all parameters. B: Parameter values during fitting. C: χ^2 value during fitting and fit settings.

4.7 Genetic algorithm

For illustration, we implemented an interface to the genetic algorithm `ga` of the homonymous MATLAB toolbox (Conn *et al.*, 1991; Goldberg, 1989; Conn *et al.*, 1997). This function does not exploit the structure of least-square optimization problems and does not possess as strong convergence characteristics as the simulated annealing approach.

After 40,000 function calls, the χ^2/N value is still larger than 100, but it should be around 1. The final deviation results in 18800%.

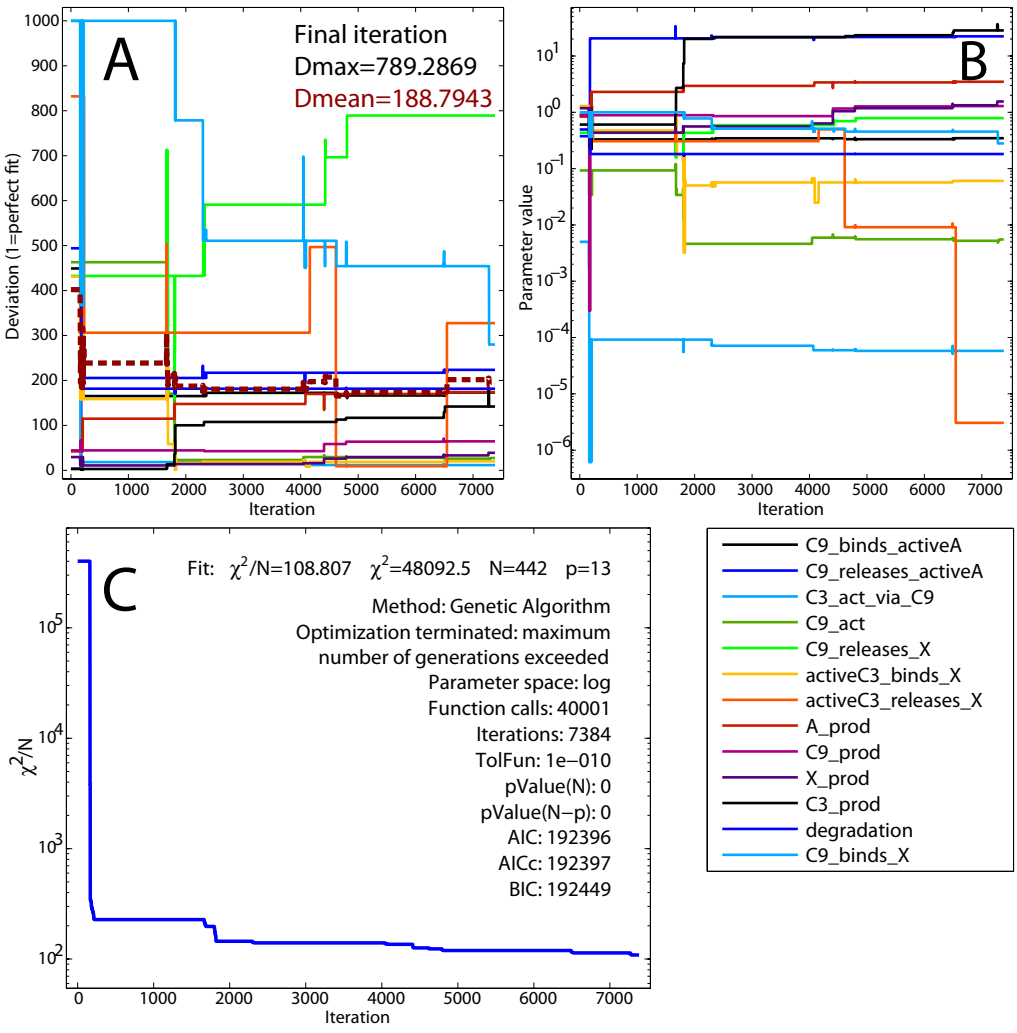

Figure 12: **Optimization performance of the genetic algorithm.** A: Deviation compared to true parameters. The dashed red line displays the mean deviation over all parameters. B: Parameter values during fitting. C: χ^2 value during fitting and fit settings.

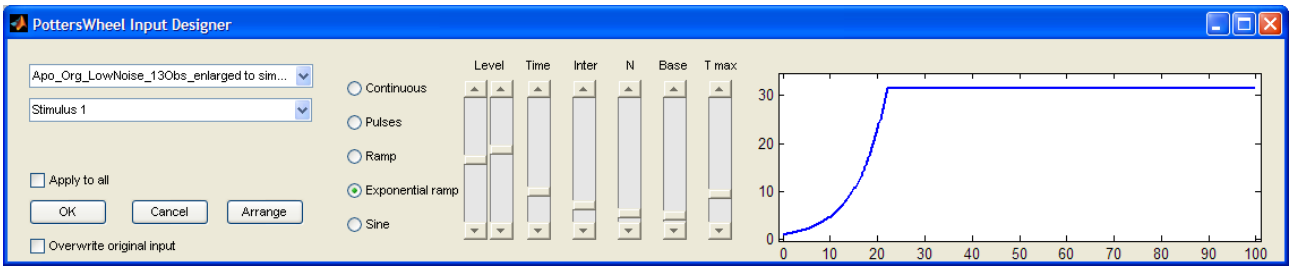

Figure 13: **Driving Input Designer.** Type and characteristics of the driving input variables of the dynamical system can be changed in real time. Currently selected is an exponential ramp stimulation, which may be useful to investigate the early phase or stimulation experiment in a stretched fashion and therefore with a higher sampling resolution.

5 Driving input

A key functionality of PottersWheel is multi-experiment fitting, where several data sets are fitted simultaneously. The power of this approach increases if experiments are applied under different experimental conditions, e.g. different dose levels, pulsed or ramp stimulations. The externally changed species, e.g. the ligand in models of signal transduction pathways, is called *driving input*.

The information about the experimental setting has to be saved within the data set (see PottersWheel documentation). To integrate, values of the driving input are required at arbitrary time points. This can be achieved by two approaches: analytic or smoothing spline interpolation.

5.1 Analytic interpolation

The driving input is specified analytically, i.e. no measurements are available, but the time-course is approximated by a step-wise or linear function.

5.2 Smoothing spline interpolation

If measurements of the driving input are available, they can be used for a smoothing-spline approximation.

5.3 Designing new driving inputs

The driving input designer is a graphical user interface where the shape of each input species can be changed in real-time (Fig. 13). Available functionalities inlcude continuous stimulations, pulses, ramps, exponential ramps, and sine waves. Characteristics like pulse number or duration can be changed by sliders. This way, new experiments can be designed, e.g. with significantly different trajectories than that of existing experiments in order to increase the power of model selection and parameter calibration.

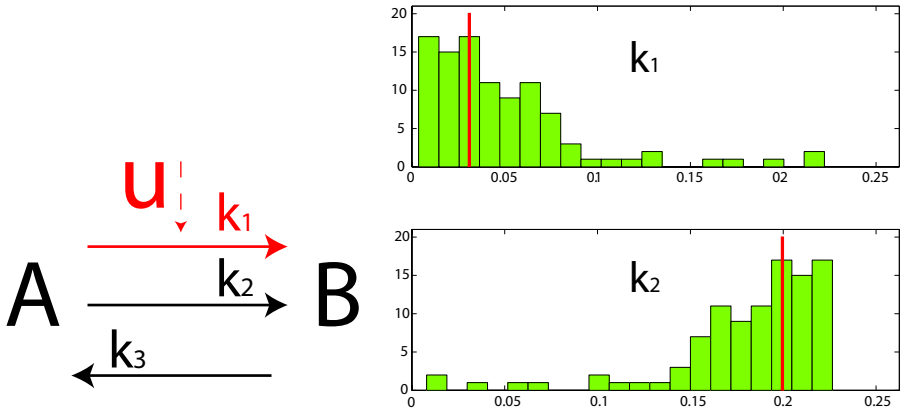

Figure 14: **Effect of multi-experiment fitting.** Left: Network with two species and three reactions. Reaction $A \rightarrow B$ takes place constitutively and is in parallel triggered by an external input function, for example an extracellular stimulus. Data is simulated for different input functions u and with 5% relative and 5% absolute error. parameter values $k_1 = 0.03, k_2 = 0.2$, and $k_3 = 0.1$. Right: If only one continuous stimulation experiment with $u(t) = 1$ is available to determine the parameters of the system, k_1 and k_2 can not be estimated accurately, but have a broad distribution depending on the initial guess of the parameter values before fitting. In fact, only the sum $k_1 u + k_2$ can be derived from the data; they are non-identifiable. If two experiments are combined with different stimulation strength $u(t) = 1$ and $u(t) = 2$, k_1 and k_2 can be determined correctly and non-identifiability is resolved(red vertical lines).

6 Multi-Experiment Fitting

The statistical power to estimate parameter values or to discriminate competing model hypotheses strongly increases if different experiments are modeled simultaneously. This *multi-experiment fitting* approach is a key functionality of Potters-Wheel and is illustrated in Fig. 14. In a small network, species A is transformed into species B via a constitutive reaction with rate $k_2 A$ and an induced reaction, which is triggered with reaction rate $k_1 u A$ by an externally given input function u, representing an external stimulus. Only one backward reaction exists with rate $k_3 B$. Data sets have been simulated with 5% relative and 5% absolute error. A sequence of 100 fits was applied, each fit starting at different initial guesses. Based on one hypothetical continuous stimulation experiment, parameters k_1 and k_2 can not be determined independently from each other, but only the sum $k_1 u + k_2$ can be derived from the data. However, if two continuous stimulations with different values of u corresponding to different doses are available, k_1 and k_2 can be estimated accurately depending on the observational noise of the measurements.

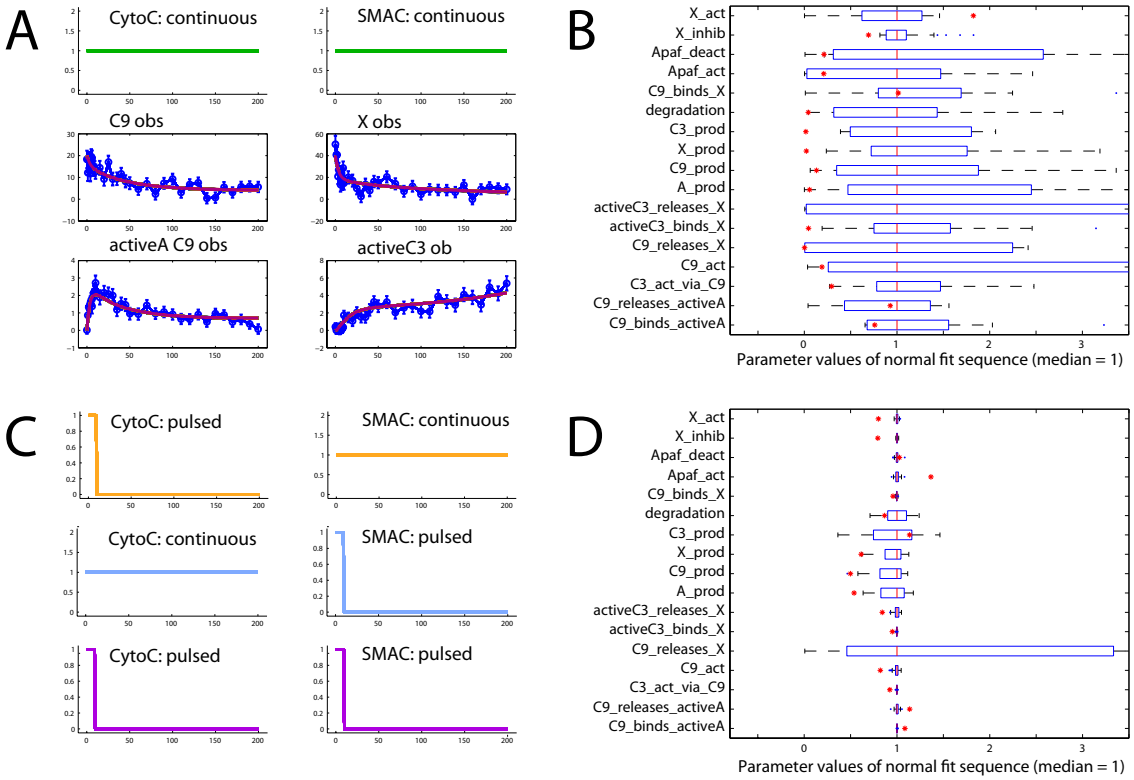

Figure 15: **Application of multi-experiment fitting.** If only one experiment with continuous stimulations for each driving input player cyto-c and SMAC (A, 4 observables, 10% rel. + 10% absolute error) is fitted 200 times with varying initial guess for the parameters, the distributions of the calibrated parameter values is rather broad: The parameters are not identifiable (B). C: Combination of all four experiments leads to significantly narrowed parameter distribution (D). Note that the distribution represented by the horizontal bars should not be mistaken for a confidence interval (see section 11.2). Therefore, true values (red stars) may lie outside of the parameter distributions.

6.1 Local and global parameters

Within the multi-experiment fitting approach, global and local parameters have to be distinguished. *Global parameters* have the same value for each experiment. They reflect structural parameters like rate constants. *Local parameters* on the other hand obtain different values depending on the experiment. This occurs for example for scaling parameters of observation functions for relative measurement techniques like Western Blotting or for initial values of protein concentrations.

6.2 Application to the apoptosis model

In section 4, the four available optimizers have been tested on simulated data with little noise and a large set of observables. When only a few species can be measured

with a medium noise level, how accurate can the original parameter values be obtained? In a hypothetical experiment, we simulate data for C9, X, activeA_C9 and activeC3 each with an error model of 10% relative and 10% absolute error. The box plot of Fig. 15B, points out that the fitted parameter values are very ambiguous. When four experiments are combined, each with a different set of continuous and pulsed stimulations for the two driving input species cyto-c and SMAC (see Fig. 15A+C), the parameter values are much better calibrated which is illustrated in Fig. 15D. The improved calibration is due to resolved non-identifiabilities by multi-experiment fitting. Six parameters still have a medium to large standard deviation. Either further experiments under different experimental settings corresponding to new driving input functions or with a different set of observable species are required to narrow the distribution of these parameters. In section 7.2 we will investigate the problem of non-identifiability in more detail. Note that the distributions in the box plots do not represent an estimation of the distribution of the calibrated parameters. Strategies to determine the distribution of fitted parameters are discussed in 11.2.

7 Fit and model analysis

7.1 Fit sequence analysis

A combination of model-data-couples can be fitted several times in a *fit sequence*. The initial parameter values are drawn from a random distribution before each fit. They are either chosen from a quasi-random-distribution between the minimum and maximum parameter values or are drawn from an exponential distribution around p_0:

$$p_{new} = p_0 \cdot 10^{s \cdot \varepsilon}$$

Here, s represents the *disturbance strength*. The random number ε is drawn from the standard normal distribution $N(0, 1)$. Depending on the fit sequence type, p_0 is either the fitted value of the last fit in the sequence, the parameter value of the best fit in the sequence, the initial value of the last fit, or given explicitly by the user.

In a normal fit sequence, the same data set is fitted repeatedly and the distribution of estimated parameter values can be used for an identifiability analysis (see section 7.2). In a Monte-Carlo fit sequence, new data sets for each fit are generated based on the model and an adequate observation error model and the calibrated values may be used to estimate the confidence regions numerically (see section 11.2).

A fit sequence results in a matrix of parameter values $p_{i,j}$, reflecting the value of parameter i after the fit j. Based on this matrix, several analyses can be applied (see Fig. 16). A subset of fits with the lowest χ^2 values can be selected in order to circumvent local minima in the parameter space (A). PottersWheel depicts mean and variance values of the estimated parameter with histograms and boxplots (B). Correlation analysis between pairs of parameters indicates whether and to what

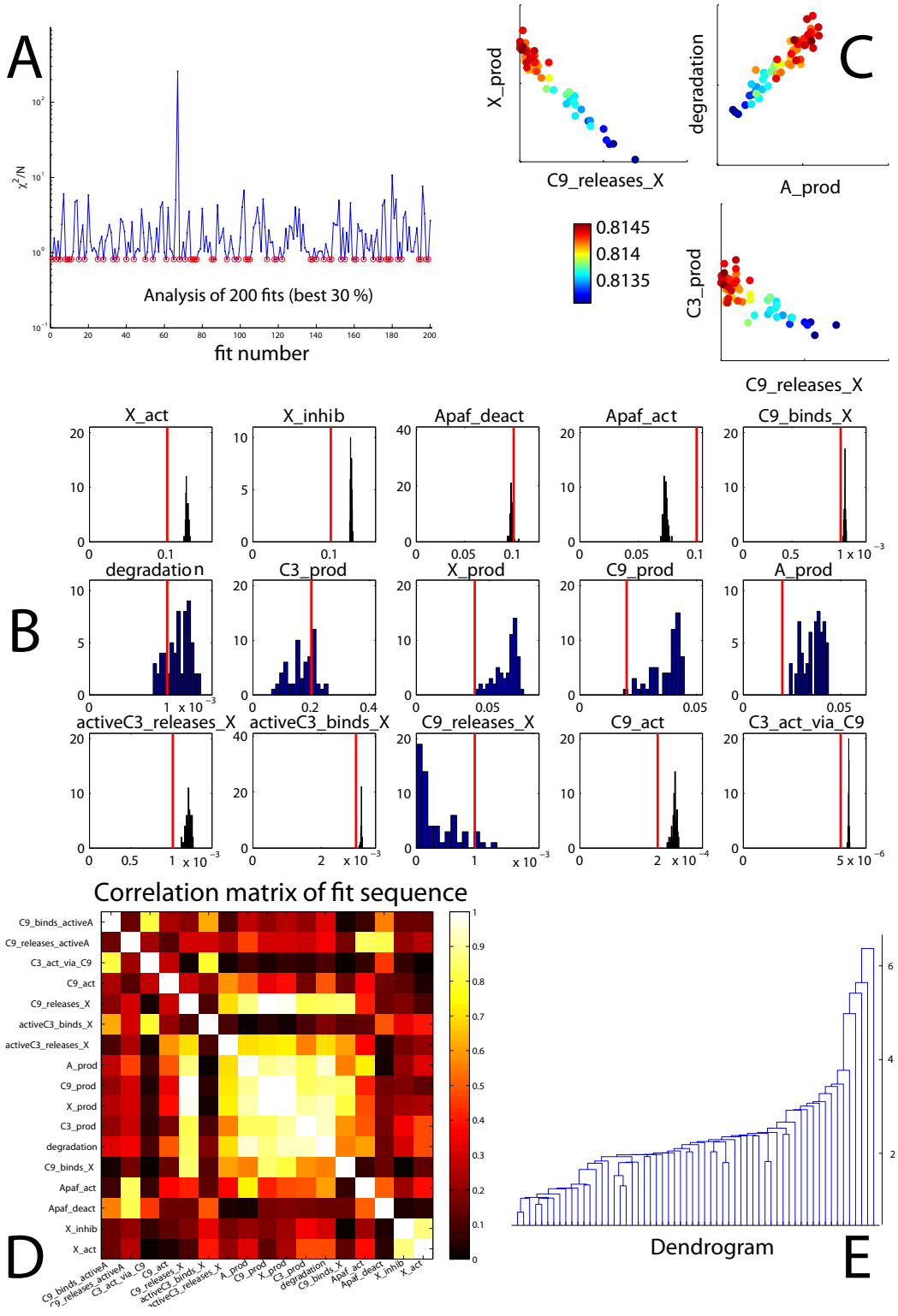

Figure 16: **Fit sequence analysis.** A: χ^2/N value of 200 fits. Only the best 30% percent will be analyzed further (red circles). B: Histogram of fitted parameter values. Red lines indicate the true value. C: Detailed scatter plots reveal functional dependencies between parameters. χ^2/N value is color coded. D: A correlation analysis demonstrates which parameters share a linear relationship. E: Hierarchical clustering of parameters based on the euclidian distance.

extent a linear correlation exists between the parameters (D). Detailed scatter plots for all significant correlations reveal linear non-identifiabilities in the model structure (C). Principal component analysis determines the total number of degrees of freedom of the system assuming linear dependencies and the two main components are shown in a biplot analysis. Hierarchical clustering of all parameter values based on their euclidian distance shows whether distinct local minima were found by the optimization routine, which is not the case here (E).

7.2 Parameter identifiability

The linear correlation analysis of fitted parameter values can only detect linear dependencies between two parameters. In order to detect nonlinear functional relationships between many parameters, Hengl *et al.* (2007) suggested a new approach called MOTA, which is based on optimal transformations using the alternating conditional expectation algorithm (Breimann and Friedman, 1985). MOTA is available as a C MEX plug-in within PottersWheel. It requires a sufficient number of fits and calculates groups of parameters sharing an unknown relationship. Each group is characterized by the amount of explained variance (r^2), the coefficient of variation (cv), and how often the same group is detected when a different parameter is taken as response. The most relevant groups are marked with a single or double score (for details see Hengl *et al.* (2007)).

We applied a normal fit sequence of 1500 multi-experiment fits to a non-identifiable model, namely the apoptosis model coupled to the four experiments as described in 14. The best best 50% of the fits were analyzed. MOTA identified 17 groups of related parameters and marked 6 with a single score. The groups are:

1. C9_releases_X, A_prod, C9_prod, X_prod, C9_binds_X ($r^2 = 0.994, cv = 0.207$)
2. A_prod, degradation, Apaf_act ($r^2 = 0.921, cv = 0.559$)
3. C9_releases_X, C9_prod, X_prod ($r^2 = 0.993, cv = 0.619$)
4. C9_releases_X, C9_prod, X_prod, degradation ($r^2 = 0.998, cv = 0.27$)
5. C9_act, C9_prod, X_prod, C3_prod, degradation ($r^2 = 0.933, cv = 0.612$)
6. C9_act, C3_prod, degradation ($r^2 = 0.972, cv = 0.395$)

In summary, 9 out of 13 parameters are affected by a scored non-identifiability: C9_act, C9_releases_X, A_prod, C9_prod, X_prod, C3_prod, degradation, C9_binds_X, and Apaf_act. This explains the strong variations in the box plot and histogram of the parameters (Figs. 15B and 16B).

7.3 System properties in case of non-identifiabilities

If not all non-identifiabilities can be removed from the system, e.g. by further measurements or reformulation of the mathematical model, statements about system

properties are hampered. The maximum of an unobserved state may depend from a non-identifiable parameter. Hence, it is not allowed to take the best fit and specify time point and value of the maximum. Instead, a distribution of values has to be determined based on all parameter settings leading to nearly the same χ^2 value or, to be very conservative, leading to a compliant data description.

PottersWheel supports the calculation of system properties for all best fits of a fit sequence. Also the trajectories can be plotted, illustrating the range of possible time courses (`pwTrajectoriesOfManyParameterSettings`).

7.4 Sensitivity Analysis

Sensitivity Analysis or control coefficients quantify the influence of input variables to output variables of a mathematical model. In our case, the input variables are usually the model parameters. The output could be the system state variables or observables themselves at one time-point, or derived characteristics like the integral over a certain time period. Hornberg *et al.* (2005) discuss several characteristics and derive summation laws for their control coefficients.

For the following measures, control coefficients can be calculated within PottersWheel (see Fig. 17):

1. Peak amplitude
2. Time of peak
3. Recovery time (definition depends on the monotony of the investigated state variable)
4. Signal duration
5. Area under curve
6. Mean value
7. Value at a certain time point

In order to increase the numerical accuracy which is required to fulfill the summation laws, PottersWheel uses a spline-interpolation to determine e.g. the value and time of the peak and the recovery time. This requires the investigated variable to be a sufficiently smooth function, which may not be the case in dynamic systems with pulsed input functions, for instance.

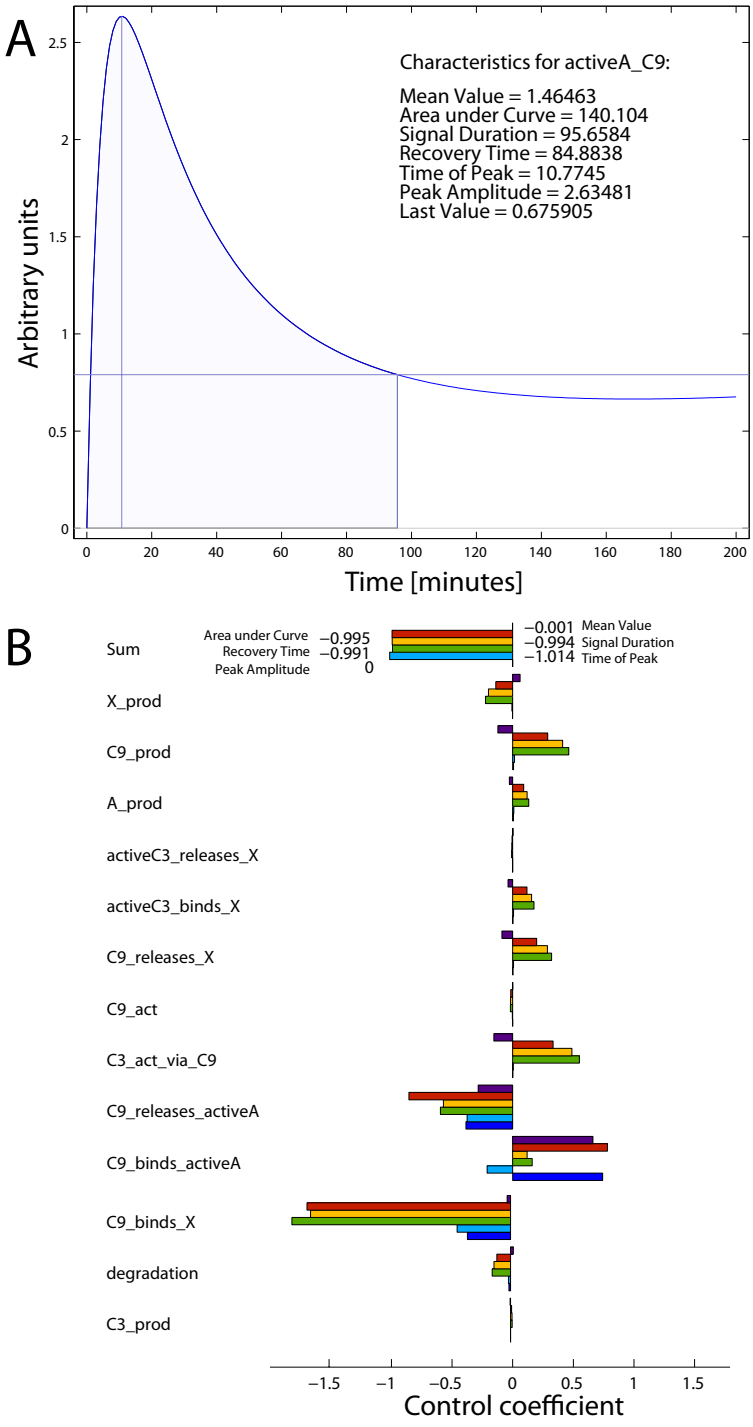

Figure 17: **Detailed sensitivity analysis.** A: Characteristics for the variable activeA_C9 are illustrated for a recovery threshold at 30% of the peak amplitude. The recovery time represents the time between peak and threshold crossing. The signal duration is the sum of time of peak and recovery time. Area under curve and mean value are determined for the signal duration. B: Control coefficients for signal characteristics (lila: mean value; red: area under curve; yellow: signal duration; green: recovery time; light blue: time of peak; dark blue: peak amplitude). The impact of each parameter on activeA_C9 is displayed. The sum over all parameters results to -1 or 0 which is compliant with the summation laws of Hornberg *et al.* (2005).

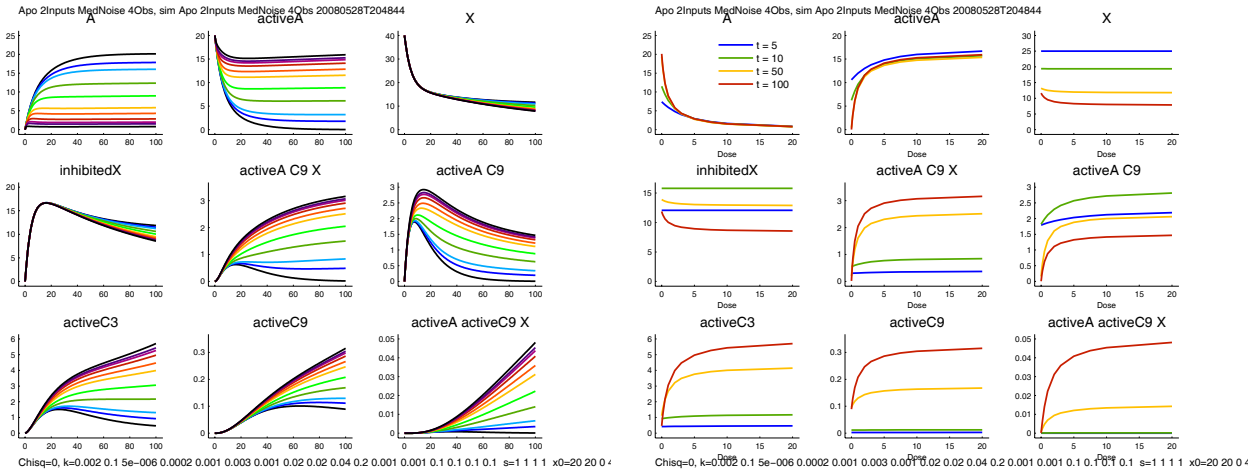

Figure 18: **Stimulus dependent view.** Several continuous stimulations with different dose-level for cyto-c are applied. Left panel: The effect of the dose-response experiment in time-domain. Right panel: The effect displayed over the dose level for four different time points $t = 5, 10, 50, 100$ minutes.

7.5 Stimulus dependent view

For some experiments, the x axis should not be time, but rather the dosage or pulse-duration. In this case, PottersWheel allows to switch between time and stimulus dependent view, strongly improving the interpretation of fitted trajectories (see Fig. 18).

7.6 Residual analysis

The residuals, i.e. the differences between model trajectory and data points, should be Gaussian distributed and uncorrelated. Systematic deviations indicate a wrong model or a wrong error model. If the original measurements have a non-Gaussian distributed error, they should be transformed appropriately Kreutz *et al.* (2007). The residual analysis of PottersWheel comprises the following graphs:

1. Residuals against time
2. Residuals embedded, i.e. residual i against residual $i-1$
3. Autocorrelation of the residuals
4. Histogram
5. QQ-Plot
6. Residuals against predicted values
7. Predicted against observed values

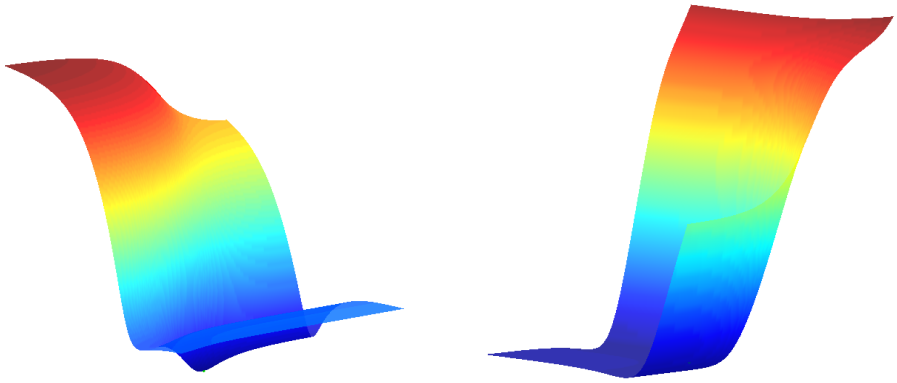

Figure 19: χ^2 **landscapes.** Two parameters are changed systematically and the resulting χ^2 value is plotted in a 2-dimensional manifold in triple-logarithmic space. Left: The minimum is located at the end of a long *bathtube* and may be reached by a powerful optimizer. Right: Two parameters are linked in a structural non-identifiability; they yield the same χ^2 value for a large set of parameter values leading to broad (infinite) distributions of calibrated parameter values. The figures can be reproduced with `pwShowChisqLandscape`

8 Confidence intervals

Calculation of confidence intervals on the estimated parameter values requires that the parameters are identifiable. If a Maximum Likelihood estimator is used to calibrate the parameters, the confidence intervals can be determined based on the Hessian of the objective function at the optimum. Since PottersWheel uses a weighted least-square optimization, this is the case for normally distributed errors. Otherwise, a Monte-Carlo approach is to be preferred, where new data sets have to be generated based on the fitted model and an adequate observation error model. PottersWheel supports both strategies, which are available from the command line via `pwConfidenceIntervals`.

8.1 The χ^2 landscape

Fig. 19 displays the dependency of the χ^2 value when two parameters are changed systematically. Structural non-identifiabilities as on the right lead to exactly the same χ^2 value for a large set of parameter values, rendering it impossible to locate a unique optimum. In this case, the confidence intervals are infinite: if the initial guesses of the parameters within a fit sequence fluctuates stronger, also the fitted parameters values will have a broader distribution.

8.2 Hessian-based confidence intervals

The χ^2 merit function which is optimized within PottersWheel to fit the model $y = y(t; (p))$ is

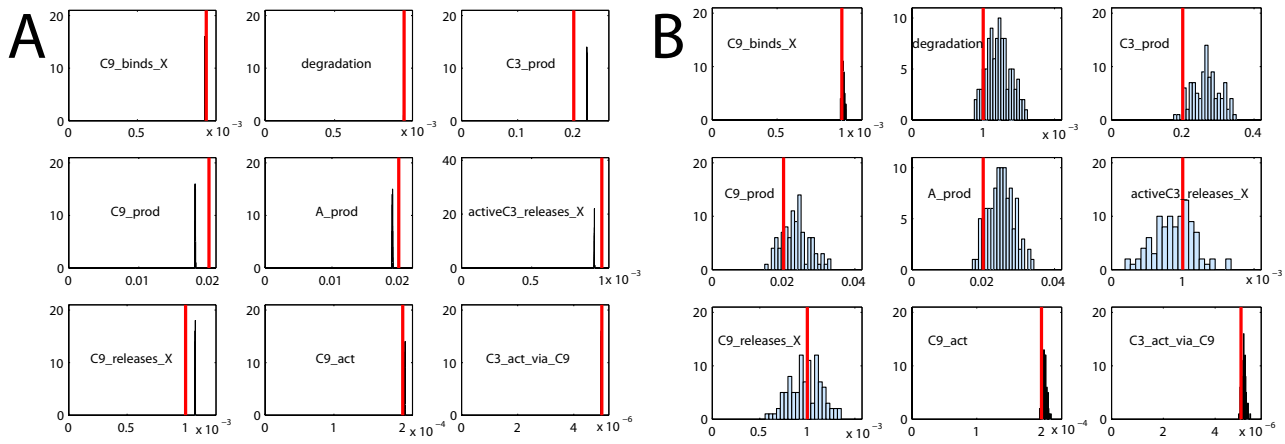

Figure 20: **Normal and Monte-Carlo fit sequence of an identifiable model.** A: Histogram of calibrated parameter values after fitting the same data set with random initial guesses. Red lines indicate the true value. In an identifiable model, a fit sequence to the same data set will lead to sharp peaks, which usually do not coincide with the true values. B: Histograms for Monte-Carlo generated data sets. If an adequate error model has been used, the distribution of calibrated parameter values of a Monte-Carlo fit sequence can be used to estimate confidence intervals.

$$\chi^2(\mathbf{p}) = \sum_{i=1}^{N} \left(\frac{y_i - y(t_i; \mathbf{p})}{\sigma_i} \right)^2 .$$

The components of the gradient and Hessian depending on the parameter vector $\mathbf{p}$ are given as

$$\frac{\partial \chi^2}{\partial p_k} = -2 \sum_{i=1}^{N} \frac{y_i - y(t_i; \mathbf{p})}{\sigma_i^2} \frac{\partial y(t_i; \mathbf{p})}{\partial p_k} \tag{3}$$

$$\frac{\partial^2 \chi^2}{\partial p_k \partial p_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left(\frac{\partial y(t_i; \mathbf{p})}{\partial p_k} \frac{\partial y(t_i; \mathbf{p})}{\partial p_l} - (y_i - y(t_i; \mathbf{p})) \frac{\partial^2 y(t_i; \mathbf{p})}{\partial p_l \partial p_k} \right) . \tag{4}$$

The second term of the Hessian is often negligible (Nocedal and Wright, 1999). Checchi and Marsili-Libelli (2005) suggest an approach to exploit a non-negligible second term to verify the success of the optimization. Based on the approximation of the Hessian, the covariance matrix of the estimated parameters is given as (Marsili-Libelli *et al.*, 2003; Press *et al.*, 1999):

$$C = \left(\sum_{i=1}^{N} \frac{1}{\sigma_i^2} \frac{\partial y(t_i; \mathbf{p})}{\partial p_k} \frac{\partial y(t_i; \mathbf{p})}{\partial p_l} \right)^{-1} .$$

The diagonal of C are the Hessian-based variances of the calibrated parameters. They are calculated by `pwConfidenceIntervals`.

8.3 Monte-Carlo approach

The fitted model is used in the Monte-Carlo approach to generate M new *synthetic* data sets (Press *et al.*, 1999):

$$y_j^{MC}(t_i; \mathbf{p}) = y(t_i; \mathbf{p}) + \varepsilon_i, \ 1 \leq j \leq M$$

with ε_i taken from an adequate error distribution, e.g. $\sim N(0, \sigma_i)$. The model is fitted to the M data sets and the confidence limits are given by the standard deviation of the calibrated parameter values.

Fig. 20 displays the distribution of calibrated parameters for 100 Monte-Carlo data sets. Their standard deviations are estimators for the standard deviations of the fitted parameter values.

8.4 Comparison of confidence intervals

The two approaches to determine the confidence intervals are compared for an identifiable model with 13 observables and small noise. Both methods yield similar results. Identifiability of parameter values is an important prerequisite.

ID	true value	Std. by Hessian		Std. by Monte-Carlo	
C9_binds_activeA	0.002	4.4012e-005	2.23%	4.4947e-005	2.28%
C9_releases_activeA	0.1	0.0026946	2.73%	0.0027774	2.81%
C3_act_via_C9	5e-06	7.9406e-008	1.59%	8.4125e-008	1.68%
C9_act	0.0002	3.076e-006	1.51%	3.2636e-006	1.6%
C9_releases_X	0.001	0.00017693	17.32%	0.00016578	16.23%
activeC3_binds_X	0.003	8.501e-005	2.74%	9.4822e-005	3.05%
activeC3_releases_X	0.001	0.00024517	25.5%	0.00026282	27.34%
A_prod	0.02	0.0039089	18.77%	0.003494	16.77%
C9_prod	0.02	0.0041509	20.94%	0.0038154	19.24%
X_prod	0.04	0.0071633	17.25%	0.0063447	15.28%
C3_prod	0.2	0.041051	17.13%	0.035656	14.88%
degradation	0.001	0.00018176	16.92%	0.00016506	15.37%
C9_binds_X	0.001	1.0194e-005	1.03%	1.0525e-005	1.06%

9 Statistical tests

Two questions arise when experimental data is modeled:

1. Is the model statistically compliant with the data, i.e. what is the probability that the model produces the data set?
2. If two models are compliant with the data, which one should be taken?

In order two answer the first question, usually the goodness-of-fit is determined, i.e. the distance between model and data is related to the expected value if the model were true: a χ^2 test is applied. For the second question, it is important to verify whether the models are nested, i.e. whether one model is an extension of the other model. In this case it may be permitted to apply a likelihood-ratio test with high statistical power Lehmann (1986). Criteria like AIC and BIC are suggested to establish a ranking of models.

9.1 Maximum Likelihood approach

The *likelihood* is defined as the probability to measure y_{Meas} given the parameters θ. The Maximum Likelihood approach calibrates parameters in order to increase the likelihood,

$$\theta_{ML} = \max_{\theta} \arg\, pr(y_{Meas}|\theta).$$

In the case of normally and independent distributed errors, the likelihood can be expressed as

$$\mathcal{L}(\theta|y_{Meas}) := pr(y_{Meas}|\theta)$$
$$= \prod_i \frac{1}{\sqrt{2\pi}\sigma_i} exp\left(-\frac{(y_i^{Meas} - y_i^{Model})^2}{2\sigma_i^2}\right).$$

For convenience, instead of the likelihood its logarithm is maximized – the *log-likelihood* L,

$$\begin{aligned} L(\theta|y_{Meas}) &= -\sum_i \frac{(y_i^{Meas} - y_i^{Model})^2}{2\sigma_i^2} \\ &\quad -N \log\sqrt{2\pi} - \sum_i \log\sigma_i. \end{aligned}$$

Since the second and third term do not depend on θ, it is sufficient to maximize the first expression which is equivalent to minimizing the half χ^2 value as it is done within PottersWheel.

9.2 χ^2 test

If the residuals between model and measurements follow a Gaussian distribution, L is a sum of N squares of normally distributed quantities, each normalized to unit variance (Press *et al.*, 1999). L is under the null-hypothesis distributed like a chi-square distribution χ^2_{N-M} with $N - M$ degrees of freedom, with M being the number of parameters. The null-hypothesis has three parts:

1. The model is sufficient to explain the measurements.
2. The true standard deviations do not exceed σ_i.
3. The residuals are Gaussian distributed.

If for a given fitted model the obtained L is significantly incompatible with the corresponding χ^2 distribution, one can conclude that at least one of the above assumptions is violated. This could mean that the model is wrong, but it could equally mean that just the residuals are not Gaussian distributed. The latter underlines the importance for correcting of outliers in the data or a proper error model. If the model is compliant with the measurements, one may only conclude that the null-hypothesis can not be rejected.

After each fit, PottersWheel calculates the p value for the given null-hypothesis based on $N - M$ degrees of freedom and also for N itself. This is due to situations where some parameters are identifiable resulting in a number of degrees of freedom between $N - M$ and N. The result is displayed in the command window and within a fit information figure when calling `pwInfoPlotsFitting`.

9.3 Likelihood ratio test for nested models

The likelihood-ratio test evaluates nested models $M_1 \subset M_2$ with $r_1 < r_2$ parameters to each other and determines the probability that the smaller model is sufficient to describe the measurements (Neyman and Pearson, 1933; Bauer *et al.*, 1988). The null-hypothesis is based on the following assumptions:

1. The maximum-likelihood estimated parameters $\hat{\theta}$ are normally distributed around the true parameters θ_0, i.e.

$$pr(\hat{\theta} - \theta_0) = \frac{|C|^{-\frac{1}{2}}}{2\pi} \exp\left(-\frac{1}{2}(\hat{\theta} - \theta_0)^T C^{-1}(\hat{\theta} - \theta_0)\right)$$

   with the negative inverse Hessian

$$C_{i,j} = -\left(\frac{\partial^2 L(\hat{\theta}|y^{Meas})}{\partial\theta_i \partial\theta_j}\right)^{-1}$$

2. M_1 is the *true* model.
3. M_1 is equivalent to M_2 with one or more fixed or depending parameters θ_i. All constant θ_i do not belong to the border of the parameter space used to fit the larger model M_2.
4. All parameters θ_i are identifiable.

Under the null-hypothesis, $2\left(L(\hat{\theta}) - L(\theta_0)\right)$ is χ^2 distributed with $r_2 - r_1$ degrees of freedom.

We applied this approach to a by four reactions extended and a reduced version with one missing reaction from the apoptosis model. The result of the fits can be summarized as follows:

Model	χ^2	goodness-of-fit	n_p	LRT against true model
true model	368.175	0.98	13	
reduced model	681.386	0	12	0
enlarged model	361.591	0.988	17	0.16

The reduced model is not compliant with the data (goodness-of-fit: p=0). The true and the enlarged model are, as expected, compliant with the data. The enlarged model has a lower χ^2 value, which is however not significant, i.e., taking into account 4 additional reactions can not describe the data significantly better then the original model (`pwLRT`).

9.4 AIC and BIC

Information theory-based approaches may be used to establish a ranking between competing hypotheses. Based on the Kullback-Leibler Information measure (Kullback, 1987), Akaike suggested the Information Criterion $AIC = -2LL + 2p$ with the number of parameters p (Akaike, 1973). The preferred model is the one with the lowest AIC value. The Bayesian Information Criterion, $BIC = -2LL + p\ln(N)$ also takes into account the number of data points N, and tends to be more conservative than AIC (Schwartz, 1978). If the set of investigated models contains the true one, the BIC criterion is asymptotically consistent, but this may not be the case for AIC (Bonate, 2006).

10 Advanced modeling techniques

10.1 Model families

Often, a basic model is extended to investigate new hypotheses. If the basic model is changed, usually all extended versions have to be changed, as well. In order to avoid redundant work, *model families* can be used, essentially exploiting the fact, that PottersWheel model definition files are normal MATLAB functions: In the header section, not an empty model is created by `m = pwGetEmptyModel()`, but the mother model is used, `m = NameOfMotherModel()`.

10.2 Algebraic equations: Rules, start value assignments and events

The mathematical model may not only comprise differential equations, but also a set of algebraic equations, e.g. if one parameter is always defined by the fraction of two other parameters. Algebraic equations are distinguished by the time-point when they are evaluated:

- So called rules are evaluated before each integration step, i.e. they re-calculate functional relationships at each time-point of the model trajectory. In SBML, they are referred to as *assignment rules*.
- Start value assignments are only evaluated once, namely before integration starts. They can be used to determine parameters depending on the initial value of some species, e.g. in order to fulfill a basal level constraint.
- Events are algebraic equations which should be evaluated when a specified condition becomes true, which may happen several times during a time-course. An illustrative example is a model for a jumping ball with height h. When the ball hits the ground, i.e. $h = 0$, the movement direction has to be reversed.

PottersWheel supports all kind of algebraic equations. Currently, events are only possible for parameters and not dynamic variables.

10.3 Basal states

In the small reaction network $A \rightarrow B$ and $B \rightarrow A$ the steady state depends on the parameter values of forward and reverse reaction. If the steady state corresponds to an unstimulated system, modeling of the stimulated system requires initial values in accordance to the unstimulated case. Since this *basal* level depends on the parameter values, after each parameter change, e.g. during data fitting, a new basal level has to be calculated. If an analytic calculation like in the small network is possible, start value assignment can be used to set the corresponding variables. Otherwise, the system should be integrated as long as required before stimulation start in order to reach the basal steady state level. This can be achieved within PottersWheel by specifying m.tStart and m.tPlotStart. The integration will start at m.tStart, whereas the plotting starts not before m.tPlotStart which is useful if the system

reaches basal levels only after a long time period and the time span of interest is much shorter.

10.4 Naming conventions

PottersWheel supports arbitrary IDs for reactants, products, and modifiers (enzymes), as long as only letters, numbers and the underscore are used and the first character is not a number. However, in order to make use of subreaction–networks and combinatorial complexity functionalities, the following conventions are recommended:

- Basic species start with a capital letter, e.g. Erk and Mek
- Modifications are lower case prefixes, e.g. pErk and ppErk
- Species of a complex are separated by an underscore, e.g. pR_pR and Gab1_Grb2

10.5 Rule based modeling for combinatorial complexity

Suppose a protein E with n different binding states $E_1, ..., E_n$ and acting enzymatically on the reaction $A \rightarrow B$. An exact mathematical ODE formulation requires the ability to distinguish all different states of E leading to n reactions. If A also has m different states, then there are $m \cdot n$ reactions that need to be formulated. This phenomenon is called *combinatorial complexity* (Blinov *et al.*, 2004; Borisov *et al.*, 2005; Conzelmann *et al.*, 2006). In order to keep the number of reactions small, PottersWheel supports *rule-based* modeling. For this, a *regular expression*-like syntax has been developed for biochemical complexes. The above example with only one species A would be formulated as

$$A \rightarrow B \text{ via } *\_E\_*$$

with * representing any other species which is in a complex with E. Also considering all complexes involving A would lead to

$$*\_A\_* \rightarrow B \text{ via } *\_E\_*$$

In order to keep the unknown molecules, *tagged* expressions can be used:

$$\texttt{<1:*>\_A\_<2:*>} \rightarrow \texttt{<1>\_B\_<2>}$$

Not only different complexes involving a molecule can be distinguished, but also different *modification* states of one molecule itself. The Erk molecule exists for example in the states `Erk, pErk, ppErk` with none, one, or two phosphorylations. The expression `p|ppErk` represents all phosphorylated Erk molecules and `*Erk` all Erk molecules.

10.6 Derived variables and parameters

PottersWheel supports independent display and analysis of all dynamic variables and all observables. Beyond this, it is sometimes useful to focus on a subset of dynamic variables or on interesting function of variables, especially while working with large models. For this reason, derived variables can be defined depending – similar as observables – on the dynamic variables.

Similarly, not all parameters are of interest or for some parameters only their fraction or another functional relationship detected by identifiability analysis should be applied in a fit sequence analysis. Then derived parameters can be defined being functions of the dynamical parameters.

10.7 Soft constraints

Implicit algebraic equations and inequalities can be considered during parameter calibration as *soft constraints*. If a soft constraint is not fulfilled, a penalty term is added to the minimized merit function. The scaling between model-data-residuals and constraints can be specified by a λ factor.

An example of a simple constraint is $\max(A)/\max(B) > 5$, where the maximum of species A should be at least 5 times higher than the maximum of species B. Advanced constraints using inline MATLAB expressions are also possible, e.g. if species A should be 5 times higher than the value of B at the maximum of A: $\max(A)/B(A == \max(A))$.

10.8 Delay reactions

Signal transduction pathway models often contain delays, as in the example of protein synthesis starting from transcription to translation. Using delay integrators for delay differential equations requires specification of the first τ seconds of the system variables with τ being the largest occurring delay. This approach leads to exact delays. In $A \xrightarrow{\tau} B$, a step input for A would lead to a delayed step input for B. Since many biochemical processes are not expected to last always exactly the same period of time but follow rather a distribution of time values, PottersWheel applies the *linear chain trick* suggested by MacDonald (1976) where the delay of an input signal $g(t)$ is approximated by N steps with equal reaction rate k:

$$\begin{aligned} \dot{x}_1 &= kg(t) - kx_1 \\ \dot{x}_2 &= kx_1 - kx_2 \\ &\vdots \\ \dot{x}_N &= kx_{N-1} \end{aligned}$$

The mean delay is given as $\bar{\tau} = \frac{N}{k}$. Its standard deviation can be calculated as $\sigma_\tau = \frac{\tau}{\sqrt{N}}$ (for a proof see section B). Hence, for a standard deviation of 33%, $N = 9$ steps are necessary.

10.9 Network reconstruction

PottersWheel supports reaction based modeling, where a set of chemical reactions is specified and will be translated into ordinary differential equations. In addition, an ODE system can be entered directly, e.g. if the underlying system does not derive from a chemical network or if only the differential equations are at hand. In order to display a graphical representation of the system, PottersWheel translates the ODE system into a network.

Saving the apoptosis example model using `pwExportCouplesToPWODE` as an PW ODE model definition file yields:

$$
\begin{aligned}
\dot{x}_1 &= -k_1 x_1 x_2 + k_2 x_6 - k_1 x_1 x_{10} + k_2 x_{13} - k_1 x_1 x_3 + k_2 x_5 - k_1 x_{11} x_1 + k_2 x_{12} + k_8 - k_{12} x_1 \\
\dot{x}_2 &= -k_1 x_1 x_2 + k_2 x_6 - k_4 x_2 x_8 - k_{13} x_2 x_4 + k_5 x_3 + k_9 - k_{12} x_2 \\
\dot{x}_3 &= -k_1 x_1 x_3 + k_2 x_5 + k_{13} x_2 x_4 - k_5 x_3 - k_{12} x_3 \\
\ldots &= \ldots \\
\dot{x}_{14} &= +k_1 x_1 x_{10} - k_2 x_{13} + k_{13} x_{12} x_4 - k_5 x_{13} - k_{12} x_{13}
\end{aligned}
$$

Loading an ODE model based on these equations using `pwAddODEModel` leads to exactly the same reaction network as the original one. The identity can be tested with `pwCompareReactionNetworks` listing all reactions occurring only in one of the networks.

10.10 Subnetworks

If a model is created in accordance to the PottersWheel naming convention (see section 10.4), subnetworks can be visualized and printed as reaction lists. Then, it is possible to detect typos, missing or wrong reactions even in large networks.

10.11 Model refinement

The general concept of data-based modeling is to fit an existing model to the observed experimental data, to investigate the discrepancies between model trajectories and data points, and if necessary to adapt the model structure accordingly. Changes to the model often involve time-consuming tasks, which are automated within PottersWheel. Newly modified models can be reloaded and the experimental data is applied automatically. Fitting or analysis procedures can be formulated as MATLAB programs, reducing the time to apply the same procedures on a refined model. Since crucial model parsing functions are implemented as C files, loading of a model is very fast.

11 Experimental data

The PottersWheel documentation provides a detailed description of the supported format of external data files. Here we describe two important features of PottersWheel concerning efficient data handling and calculation of standard deviations.

11.1 Mapping dialog

Experimental data is saved in external ASCII or Excel files with arbitrary column names. In order to efficiently map the names of model observables to the column names, PottersWheel provides a mapping dialog. The *model-data-connection* can be saved and is reused automatically when the data set is used with the same observation again. Often, different experiments also have a different set of measured system species. In this case, it would be very tedious to define data-set specific model observations to match the correct set of observed species. Instead we recommend to set up a model observation with the union of all potentially measured species and to select 'not available' in the mapping dialog for unavailable species in the currently added data set.

11.2 Calculation of standard deviations

Each data point must have a standard deviation, expressing on the one hand the relative uncertainty to trust the data point during parameter calibration and on the other hand to render possible statistical conclusions about the model-data-compliance. Data sets with too high standard deviations will be compliant with many models.

Three approaches are available in PottersWheel to determine the standard deviation of each data point.

11.2.1 Direct specification

In the external data sheets, the standard deviation can be entered directly using extra columns with the prefix `stdCol-`, e.g. for a species saved in column 'Erk', the standard deviations should be saved in a column 'stdCol-Erk'.

11.2.2 Error-model-based

Often, an error model is determined based on experimental test-measurements with known true values. Within PottersWheel, this error model can be specified together with the observables definition in `pwAddY`. For example an error model with 5% relative and 10% absolute error (relative to the maximum of the observable over all time points) is given by $\sigma_i = 0.05 y_i + 0.1 \max(y)$.

11.2.3 Smoothing-spline-based

A spline is usually a piecewise cubic function which can be used to approximate a discrete time series at in-between points. Smoothing splines are not forced to coincide with the original data points allowing for smoother, less oscillating approximations (Reinsch, 1967). PottersWheel uses a smoothing spline of the measured time course to estimate the observable, $\hat{y}$. The variance of point i is estimated by

$$\hat{\sigma}_i^2 = \frac{1}{2h} \sum_{i=-h}^{h} (y_{i+h} - \hat{y}_{i+h})^2 \tag{5}$$

The window width $2h + 1$ results from the characteristics of the time course. Note that this approach is only reasonable when a sufficiently dense sampling of the time course is available.

Different smoothing spline algorithms are supported by PottersWheel:

1. `csaps` of the MATLAB spline toolbox with adjustable smoothness parameter based on de Boor (1978).
2. `mgcv` splines of the free statistics program R using generalized cross validation (Wood, 2000).
3. `csvspl` splines (Craven and Wahba, 1979) using the freely available MATLAB interface of Xie (2004), again with generalized cross validation to estimate the unknown smoothness parameter.

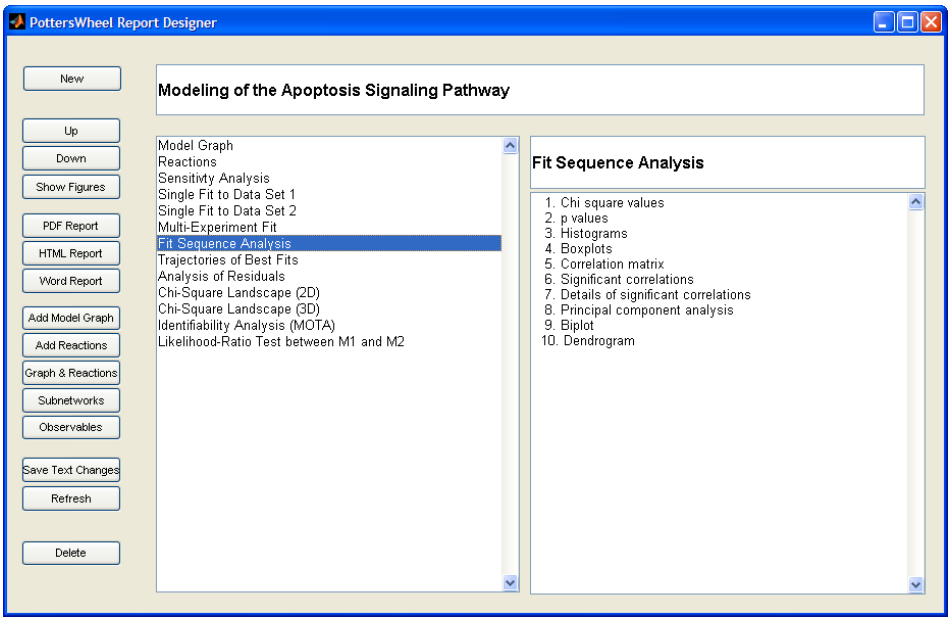

Figure 21: **PottersWheel Report Designer.** Every analysis of PottersWheel can easily be appended as a section to a report. Each section contains the figures and results of the analysis, e.g. the values of fitted parameters. Sections can be renamed, reordered, and deleted. Figures specific to a single section can be shown. Finally, the complete report can be saved as a pdf, doc, or html document.

12 Reports, workspace, and SBML

PottersWheel provides a powerful reporting mechanism, with which after each analysis, the current figures can be added as a section to a report object by clicking the 'Append' button of the main user interface. The order and title of each section can be changed in the *Report Designer*. Fig. 21 shows the graphical user interface after adding several analyses to a modeling report for the apoptosis signaling pathway. The figures specific to one section can be viewed with the 'Figures' button. In addition, selected sections can be deleted. LaTeX-like citations, such as the description of reactions or parameters are referenced in the report using bibtex. The report can be saved as a PDF LaTeX file, as a MS Word document or an HTML web-site. The PDF file contains in addition to the pictures also information about the analysis results, e.g. the values of fitted parameters if a fit has been added to the report.

The complete workspace can be saved, reloaded, and shared with colleagues. In addition, the use of custom MATLAB functions based on the PottersWheel API, allows the modeler to reproduce each step of the modeling and helps to automate repetitive tasks.

In order to follow the MIRIAM standard (Novere *et al.*, 2005), modeling results can be saved within the public, standardized SBML format.

APPENDIX

A System requirements and the PottersWheel API

PottersWheel requires MATLAB 7.1 or better. Since 2005, it is intensively used on Windows, Linux and Intel Mac OS X computers, for which system specific MEX files are available. Please contact Thomas Maiwald if you are working on a different platform. We tested the MEX compiler setup on Windows with lcc, on Linux with gcc 3.3, 3.4, and 4.0.3, and on Mac with i686-apple-darwin9-gcc-4.0.1.

In order to use the fast integration scheme, where the differential equations and the FORTRAN integrator are compiled into one executable, Windows user need to install the freely available lcc.exe and fortran.exe. This is not required for Linux and Mac OS X, since g77 is usually already available. Please download the PottersWheel documentation for further installation details.

No special MATLAB toolboxes are required, since optimization and splines algorithms of freely available toolboxes are included into PottersWheel.

A steadily growing number of currently 250 documented MATLAB functions exist to apply PottersWheel procedures within own MATLAB scripts or from command line. Please visit the documentation section at www.PottersWheel.de for an up-to-date list and description or use `help functionName` within the MATLAB command window. The following program loads a model, adds two data sets, fits the model, and generates a pdf report including a model graph and the fitted trajectories with fitted parameter values:

```
pwAddModel('Model1.m'); pwDuplicate;
pwSelect(1); pwAddData('DataSet1.xls');
pwSelect(2); pwAddData('DataSet2.xls');
pwSelect('all'); pwCombine; pwFit;
pwReportAppendGraphsAndReactions;
pwReportAppendLastStep; pwReportGenerate;
```

B Standard deviation for linear-chain-trick

In the linear chain trick (MacDonald, 1976), the delay of a scaled input signal $g(t)$ is approximated by N steps with equal reaction rate k:

$$\begin{aligned}
\dot{x}_1 &= kg(t) - kx_1 && (6)\\
\dot{x}_2 &= kx_1 - kx_2 && (7)\\
\dot{x}_3 &= kx_2 - kx_3 && (8)\\
&\vdots\\
\dot{x}_N &= kx_{N-1} - kx_N && (9)
\end{aligned}$$

Then, it holds for the mean delay time and its standard deviation:

$$\bar{\tau} = \frac{N}{k},\ \sigma_\tau = \frac{\bar{\tau}}{\sqrt{N}} \tag{10}$$

These relations can be proven by means of the Laplace transformation defined as

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t) dt \tag{11}$$

with the following properties for a suitable real-valued function $f(t)$:

- Linearity

$$\mathcal{L}\{a_1 f_1(t) + a_2 f_2(t)\} = a_1 \mathcal{L}\{f_1(t)\} + a_2 \mathcal{L}\{f_2(t)\} \tag{12}$$

- Convolution

$$\mathcal{L}\left\{\int_0^t f_1(t-\tau) f_2(\tau) d\tau\right\} = \mathcal{L}\{f_1(t)\}\mathcal{L}\{f_2(t)\} \tag{13}$$

- Differentiation

$$\begin{aligned} \mathcal{L}\{f^{(n)}(t)\} &= s^n F(s) - s^{n-1} f_0 - \ldots - s f_0^{n-2} - f_0^{n-1}, \\ \text{with } f_0^\nu &= \lim_{t \to +0} \frac{d^\nu f(t)}{dt^\nu} \end{aligned} \tag{14}$$

- Shifting

$$\mathcal{L}\{f(t-b)\} = e^{-bs} F(s) \tag{15}$$

- Damping

$$\mathcal{L}\{e^{-\alpha t} f(t)\} = F(s+\alpha) \tag{16}$$

- Multiplication

$$\mathcal{L}\{t^n f(t)\} = (-1)^n F^{(n)}(s) \tag{17}$$

Application of the Laplace transformation to equation 6 using rules 12 and 14 yields:

$$\mathcal{L}\{\dot{x}_1\} = \mathcal{L}\{k g(t) - k x_1\} \tag{18}$$

$$\Leftrightarrow\ s\mathcal{L}\{x_1\} - x_1(0) = k\mathcal{L}\{g(t)\} - k\mathcal{L}\{x_1\} \tag{19}$$

With $x_i(0) = 0\, \forall\, i \in \{1, ..., N\}$ this reduces to

$$(s+k)\mathcal{L}\{x_1\} = k\mathcal{L}\{g(t)\} \tag{20}$$

$$\Leftrightarrow \mathcal{L}\{x_1\} = \frac{k}{s+k}\mathcal{L}\{g(t)\} \tag{21}$$

In parallel, it holds for $\dot{x}_i$

$$\mathcal{L}\{\dot{x}_i\} = \mathcal{L}\{k x_{i-1} - k x_i\} \tag{22}$$

$$\Leftrightarrow s\mathcal{L}\{x_i\} = k\mathcal{L}\{x_{i-1}\} - k\mathcal{L}\{x_i\} \tag{23}$$

$$\Leftrightarrow (s+k)\mathcal{L}\{x_i\} = k\mathcal{L}\{x_{i-1}\} \tag{24}$$

$$\Leftrightarrow \mathcal{L}\{x_i\} = \frac{k}{s+k}\mathcal{L}\{x_{i-1}\} \tag{25}$$

$$\tag{26}$$

Hence, the Laplace transformation of x_N can be written as

$$\begin{aligned}
\mathcal{L}\{x_N\} &= \frac{k}{s+k}\mathcal{L}\{x_{N-1}\} && (27)\\
&= \left(\frac{k}{s+k}\right)^2 \mathcal{L}\{x_{N-2}\} && (28)\\
&= \left(\frac{k}{s+k}\right)^{N-1} \mathcal{L}\{x_1\} && (29)\\
&= \left(\frac{k}{s+k}\right)^{N} \mathcal{L}\{g(t)\} && (30)
\end{aligned}$$

In order to apply the convolution property to equation 30 the back-transformation $f(t)$ of $F(s) = (\frac{k}{s+k})^N$ is required. For this purpose, a special case of the multiplication property is considered. Let $f(t) \equiv 1$ with $F(s) = \frac{1}{s}$, then

$$\begin{aligned}
\mathcal{L}\{t^N\} &= (-1)^N F^{(N)}(s) && (31)\\
&= (-1)^N \left(\frac{1}{s}\right)^{(N)} && (32)\\
&= (-1)^N (-1)^N N! \frac{1}{s^{N+1}} && (33)\\
&= N! \frac{1}{s^{N+1}} && (34)
\end{aligned}$$

Applying the Laplace transformation backwards one gets

$$\begin{aligned}
\left(\frac{k}{s+k}\right)^N &= k^N \frac{1}{(s+k)^N} \frac{(N-1)!}{(N-1)!} && (35)\\
&= \frac{k^N}{(N-1)!}\mathcal{L}\{t^{N-1}\}(s+k) && (36)\\
&= \frac{k^N}{(N-1)!}\int_0^\infty e^{-(s+k)t}\, t^{N-1} dt && (37)\\
&= \frac{k^N}{(N-1)!}\int_0^\infty e^{-st}\, e^{-kt}\, t^{N-1} dt && (38)\\
&= \frac{k^N}{(N-1)!}\mathcal{L}\{e^{-kt}\, t^{N-1}\} && (39)\\
&= \mathcal{L}\left\{k^N e^{-kt}\, \frac{t^{N-1}}{(N-1)!}\right\} && (40)\\
& && (41)
\end{aligned}$$

Now equation 30 reads

$$\mathcal{L}\{x_N\} = \mathcal{L}\left\{k^N e^{-kt}\, \frac{t^{N-1}}{(N-1)!}\right\} \mathcal{L}\{g(t)\} \qquad (42)$$

which can be back transformed by means of the convolution property 13:

$$\boldsymbol{x}_N(t) = \frac{k^N}{(N-1)!}\int_0^t \tau^{N-1}e^{-k\tau}g(t-\tau)d\tau \tag{43}$$

For a real delay differential equation with delay Δt this equation would be

$$\boldsymbol{x}_N(t) = \int_0^t \delta(\Delta t-\tau)g(t-\tau)d\tau \tag{44}$$

reducing the integral to

$$\boldsymbol{x}_N(t) = g(t-\Delta t), \tag{45}$$

i.e., the value of x_N at time t equals exactly the value of g at time $t-\Delta t$. Hence, the integral kernel of equation 43 should converge with increasing N to the delta function, mathematically

$$\lim_{N\to\infty}\frac{k^N}{(N-1)!}\tau^{N-1}e^{-k\tau} = \delta(\Delta t-\tau) \tag{46}$$

for some suitable $\tau = \tau(k)$. The Laplace transformation of the delta distribution reads

$$\begin{aligned}\mathcal{L}\{\delta(\Delta t-\tau)\} &= \int_0^\infty \delta(\Delta t-\tau)\,e^{-s\tau}d\tau &(47)\\ &= e^{-\Delta t\,s} &(48)\end{aligned}$$

which can be regarded as

$$\begin{aligned}e^{-\Delta t\,s} &= \lim_{N\to\infty}\left(1+\frac{\Delta t\,s}{N}\right)^{-N} &(49)\\ &= \left(\frac{N/\Delta t}{s+N/\Delta t}\right)^N. &(50)\end{aligned}$$

On the other hand, the Laplace transformation of the integral kernel in equation 43 yields

$$\mathcal{L}\left\{\frac{k^N}{(N-1)!}\tau^{N-1}e^{-k\tau}\right\} = \left(\frac{k}{s+k}\right)^N \tag{51}$$

which equals equation 50 if $k = \frac{N}{\Delta t}$. Hence, a delay differential equation with delay Δt can be approximated via the composition of N compartments with rate constant $k = \frac{N}{\Delta t}$. The question arises how strong the kernel deviates from the δ function for finite N, e.g. measured via the standard deviation. For this, the first two moments

of the kernel κ_N are calculated:

$$\begin{aligned}
\int_0^\infty \kappa_N(t)\, t\, dt &= k^N \int_0^\infty \frac{t^N}{(N-1)!}\, e^{-kt} dt && (52)\\
&= k^N N \int_0^\infty \frac{t^N}{N!}\, e^{-kt} dt && (53)\\
&= k^N N \mathcal{L}\{t^N\}(k) && (54)\\
&= k^N N \frac{1}{k^{N+1}} && (55)\\
&= \frac{N}{k} && (56)\\
&= \Delta t && (57)\\
& && (58)\\
\int_0^\infty \kappa_N(t)\, t^2\, dt &= k^N \int_0^\infty \frac{t^{N+1}}{(N-1)!}\, e^{-kt} dt && (59)\\
&= k^N N(N+1) \int_0^\infty \frac{t^{N+1}}{(N+1)!}\, e^{-kt} dt && (60)\\
&= k^N N(N+1) \mathcal{L}\{t^{N+1}\}(k) && (61)\\
&= k^N N(N+1) \frac{1}{k^{N+2}} && (62)\\
&= \frac{N^2+N}{k^2} && (63)\\
&= (\Delta t)^2 + \frac{(\Delta t)^2}{N} && (64)
\end{aligned}$$

The variance is

$$\begin{aligned}
< \kappa_N^2 > - < \kappa_N >^2 &= (\Delta t)^2 + \frac{(\Delta t)^2}{N} - (\Delta t)^2 && (65)\\
&= \frac{(\Delta t)^2}{N} && (66)
\end{aligned}$$

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. Petrov and F. Csaki, editors, *2nd International Symposium on Information Theory*, pages 267–281, Budapest. Akademiai Kiado.

Bauer, P., Pötscher, B., and Hackl, P. (1988). Model selection by multiple test procedures. *Statistics*, **1**, 39–44.

Blinov, M. L., Faeder, J. R., Goldstein, B., and Hlavacek, W. S. (2004). BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, **20**(17), 3289–3291.

Bogacki, P. and Shampine, L. (1989). A 3(2) pair of Runge-Kutta formulas. *Appl. Math. Letters*, **2**, 1–9.

Bonate, P. L. (2006). *Pharmacokinetic-Pharmacodynamic Modeling and Simulation*. Springer.

Borisov, N. M., Markevich, N. I., Hoek, J. B., and Kholodenko, B. N. (2005). Signaling through receptors and scaffolds: independent interactions reduce combinatorial complexity. *Biophys J*, **89**(2), 951–966.

Breimann, L. and Friedman, J. (1985). Estimating optimal transformations for multiple regression and correlation. *Journal of the Americal Statistical Association*, **80**, 580–598.

Checchi, N. and Marsili-Libelli, S. (2005). Reliability of parameter estimation in respirometric models. *Water Res*, **39**(15), 3686–3696.

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

Coleman, T. F. and Verma, A. (2001). A preconditioned conjugate gradient approach to linear equality constrained minimization. *Computational Optimization and Applications*, **20**(1), 61–72.

Conn, A., Gould, N., and Toint, P. (1991). A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, **28**, 545 572.

Conn, A., Gould, N. I. M., and Toint, P. L. (1997). A globally convergent augmented lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Mathematics of Computation*, **66**, 261 – 288.

Conzelmann, H., Saez-Rodriguez, J., Sauter, T., Kholodenko, B. N., and Gilles, E. D. (2006). A domain-oriented approach to the reduction of combinatorial complexity in signal transduction networks. *BMC Bioinformatics*, **7**, 34.

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions. *Numerische Mathematik*, **31**, 377–403.

de Boor, C. (1978). *A Practical Guide to Splines*. Springer.

Dormand, J. and Prince, P. (1980). A family of embedded Runge-Kutta formulae. *J. Comp. Appl. Math.*, **6**, 19–26.

Finney, A. and Hucka, M. (2003). Systems biology markup language: Level 2 and beyond. *Biochem Soc Trans*, **31**, 1472–1473.

Gansner, E. R. and North, S. C. (2000). An open graph visualization system and its applications to software engineering.

Goldberg, D. E. (1989). *Genetic Algorithms in Search , Optimization and Machine Learning*. Addison-Wesley.

Hairer, E. and Wanner, G. (1996). *Solving Ordinary Differential Equations II: Stiff and Differentialalgebraic Problems*. Springer Verlag.

Hengl, S., Kreutz, C., Timmer, J., and Maiwald, T. (2007). Data-based identifiability analysis of non-linear dynamical models. *Bioinformatics*, **23**(19), 2612–2618.

Hornberg, J. J., Bruggeman, F. J., Binder, B., Geest, C. R., de Vaate, A. J. M. B., Lankelma, J., Heinrich, R., and Westerhoff, H. V. (2005). Principles behind the multifarious control of signal transduction. ERK phosphorylation and kinase/phosphatase control. *FEBS J*, **272**(1), 244–258.

Ingber, L. (1989). Very fast simulated re-annealing. *Math. Comput. Modelling*, **12**, 967–973.

Kreutz, C., Rodriguez, M. M. B., Maiwald, T., Seidl, M., Blum, H. E., Mohr, L., and Timmer, J. (2007). An error model for protein quantification. *Bioinformatics*, **23**(20), 2747–2753.

Kullback, S. (1987). The Kullback-Leibler distance. *American Statistician*, **41**, 340–341.

Lagarias, J., Reeds, J. A., Wright, M. H., and Wright., P. E. (1998). Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal of Optimization*, **9**, 112–147.

Legewie, S., Blüthgen, N., and Herzel, H. (2006). Mathematical modeling identifies inhibitors of apoptosis as mediators of positive feedback and bistability. *PLoS Comput Biol*, **2**(9), e120.

Lehmann, E. (1986). *Testing Statistical Hypothesis*. Wiley.

Ludwig, C. (2006). www-m3.ma.tum.de/m3/software/odes.

MacDonald, N. (1976). Time delay in simple chemostat models. *Biotechnol Bioeng*, **18**(6), 805–812.

Marsili-Libelli, S., Guerrizio, S., and Checchi, N. (2003). Confidence regions of estimated parameters for ecological systems. *Ecological Modeling*, **165**, 127–146.

Neyman, L. and Pearson, E. (1933). On the problem of the most efficient tests of statistical hypothesis. *Phil Trans Roy Soc A*, **231**, 289–337.

Nielsen, H. (2006). Immoptibox. http://www2.imm.dtu.dk/ hbn/immoptibox.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer, New York.

Novere, N. L., Finney, A., Hucka, M., Bhalla, U. S., Campagne, F., Collado-Vides, J., Crampin, E. J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J. L., Spence, H. D., and Wanner, B. L. (2005). Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat Biotechnol*, **23**(12), 1509–1515.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1999). *Numerical Recipes in C, The Art of Scientific Computing, 2nd ed.* Cambridge University Press.

R. E. Bankand W. C. Coughran, J., Fichtner, W., Grosse, E., Rose, D., and Smith, R. (1985). Transient simulation of silicon devices and circuits. *IEEE Trans. CAD*, **4**, 436–451.

Reinsch, C. (1967). Smooting by spline functions. *Numerische Mathematik*, **10**, 177–183.

Sakata, S. (2001). ASAMIN - a MATLAB gateway routine to adaptive simulated annealing software. http://www.econ.lsa.umich.edu/.

Schwartz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**, 461–464.

Shampine, L. and Reichelt, M. W. (1997). The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, **18**, 1–22.

Shampine, L. F. and Gordon, M. K. (1975). *Computer Solution of Ordinary Differential Equations: the Initial Value Problem*. W. H. Freeman.

Shampine, L. F. and Hosea, M. E. (1996). Analysis and implementation of tr-bdf2. *Applied Numerical Mathematics*, **20**.

Shampine, L. F., Reichelt, M. W., and Kierzenka, J. (1999). Solving index-1 DAEs in MATLAB and Simulink. *SIAM Review*, **41**, 538–552.

Wood, S. (2000). Modelling and smoothing parameter estimation with multiple quadratic penalties. *J.R.Statist.Soc.B 62*, **2**, 413–428.

Xie, X. (2004). gcvsplmat toolbox. http://www.stat.wisc.edu/x̃ie/.