Mathematical Modeling with POTTERSWHEEL

Version 2.0

Introduction
# Table of Contents

1. Introduction .......................................................................................................................... 4  
   1.1 Reactions, variables, and parameters ........................................................................... 5  
   1.2 Key modeling questions ............................................................................................... 8  
2. Overview and notation ......................................................................................................... 9  
3. Model creation .................................................................................................................... 10  
4. Data import ........................................................................................................................ 12  
   4.1 Meta information .......................................................................................................... 13  
   4.2 Data matrix .................................................................................................................. 15  
   4.3 Measurement error ...................................................................................................... 16  
   4.4 Creating a model-data couple ...................................................................................... 16  
5. Fitting .................................................................................................................................. 16  
6. Goodness-of-fit .................................................................................................................... 17  
7. Model refinement ................................................................................................................ 20  
8. Analysis and model prediction ......................................................................................... 21  
9. Design of new experiments ............................................................................................... 23  
10. Model exploration in the visual model designer .............................................................. 24  
11. Rule based modeling and naming convention ................................................................. 24  
12. Implicit and explicit multi-compartment modeling ......................................................... 25  
13. Configuration .................................................................................................................... 26  
14. Macros and custom Matlab programs ............................................................................. 26  
15. Reporting ........................................................................................................................... 27  
16. Model stiffness .................................................................................................................. 27  
17. Bibliography ..................................................................................................................... 28
Introduction

The program PottersWheel has been developed to provide an intuitive and yet powerful framework for data-based modeling of dynamical systems like biochemical reaction networks. Its key functionality is multi-experiment fitting, where several experimental data sets from different laboratory conditions are fitted simultaneously in order to improve the estimation of unknown model parameters, to check the validity of a given model, and to discriminate competing model hypotheses. New experiments can be designed interactively. Models are either created text based or using a visual model designer. Dynamically generated and compiled C files provide fast simulation and fitting procedures. Each function can either be accessed using a graphical user interface or via command line, allowing for batch processing within custom Matlab scripts. PottersWheel is designed as a Matlab toolbox, comprises 250,000 lines of Matlab and C code. The PottersWheel software requires Matlab 2006a or higher on any Linux, Mac, or Windows PC. The Matlab optimization toolbox and a few external programs are recommended, depending on the intended use of PottersWheel. A detailed installation description and introductory videos are given at www.potterswheel.de where the software can freely be downloaded for academic usage.

1. Introduction

Modeling aims to formulate a set of mathematical equations which are able to predict computationally the dynamic behavior of a real system, e.g. of a biochemical network. Entities of the system should preferably have a representation within the model in contrast to black-box modeling. This approach deepens the understanding about the network during the process of hypothesis generation, mathematical formulation, and comparison with experimental data potentially leading to new hypotheses. At the same time, laboratory measurements too complex to be understood by visual inspection are brought into relation. Beyond that, a formalized model is helpful to unambiguously communicate a hypothesis to colleagues. Finally, a suitable model renders possible new approaches to identify therapeutic strategies and to assess the feasibility of a research agenda minimizing the cost in terms of animals, time, and money.

Modeling approaches differ in the choice of possible model structures. Regression models use algebraic equations (1), Boolean models comprise logical gates (2), Bayesian networks (3) and stochastic models (4,5) are based on probability distributions and mechanistic models are expressed using either ordinary differential equations (ODEs) if spatial effects are neglected (6) or partial differential equations (PDEs) in the more general case (7). Each approach implies assumptions about the system at hand, requires a certain type of measurements, and permits conclusions at different levels of detail and accuracy. Moreover, the burden of computational implementation and analysis varies strongly between the modeling concepts.

Here, we focus on mechanistic modeling of biochemical networks using ODEs and the PottersWheel modeling framework (8). This approach assumes a sufficiently large number of molecules not below 1000 and no spatial effects, requires time-resolved measurements, allows predicting transient
system responses to changed environmental conditions, and often simplifies interpretation due to a one-to-one relationship to physiological entities. The numerical effort lies between regression/Boolean models and stochastic/PDE models and can usually be handled by a single PC or a small computer cluster, given that suitable algorithms are applied.

In the following we introduce key concepts and the PottersWheel terminology based on the JAK/STAT signal transduction pathway as an example for a biochemical reaction network (9).

1.1 Reactions, variables, and parameters

In the simple irreversible reaction „A to B“ assuming mass action kinetics, the reaction rate is proportional to the time-dependent concentration of species A, denoted as \([A](t)\), often abbreviated as \(A(t)\), neglecting the square brackets, or just \(A\). The proportionality is expressed using a rate coefficient, leading to the following ordinary differential equation (ODE) for the flux of the reaction:

\[
\dot{v}_R = \frac{d[A]}{dt} = [\dot{A}] = -k \cdot [A]
\]

or just \(\dot{A} = -k \cdot A\).

The time-dependent concentration for \(A\) is thus simply an exponential decay starting from an initial concentration at \(t=0\):

\[
A(t) = A_0 \cdot e^{-kt}
\]

In a more complex situation, the above analytical solution is not feasible. Instead, numerical integration is used to determine the concentration-time profiles. Several kinetic laws apart from mass action can be used, e.g. Michaelis-Menten and Hill kinetics (10).
Figure 1. Mathematical formalization of a biological hypothesis. The left panel shows a cartoon of the JAK/STAT signal transduction pathway. The right panel displays the related formalized and simplified mathematical model created and visualized using the PottersWheel model designer. In contrast to the cartoon, the receptor appears in the model three times: once for each modification state: the free receptor (R), the phosphorylated receptor (pR), and the internalized receptor iR. Every reaction is shown as a small rectangle with incoming flux from the reactants in light gray, outgoing flux to the products in dark gray, and in dashed red the effects of enzymes which are not consumed or produced within the reaction, also called modifiers. S denotes STAT. The prefixes p and np define phosphorylation in the cytoplasm and the nucleus, and the underscore “_” represents the bond of two species.

Figure 1 visualizes the formalization of a biological hypothesis into a reaction network. PottersWheel automatically creates a corresponding set of differential equations, in this case

\[
\begin{align*}
\dot{R} &= -k_1 \cdot R \cdot EPO + k_2 \cdot pR \\
\dot{pR} &= -k_2 \cdot pR + k_1 \cdot R \cdot EPO - k_3 \cdot pR \\
iR &= + k_3 \cdot pR \\
\dot{S} &= -k_4 \cdot S \cdot pR + k_8 \cdot nS \\
pS &= -2 \cdot k_5 \cdot pS \cdot pS + k_4 \cdot S \cdot pR \\
pS_{pS} &= -k_6 \cdot pS_{pS} + k_5 \cdot pS \cdot pS \\
npS_{npS} &= -k_7 \cdot npS_{npS} + k_6 \cdot pS_{pS} \\
nS &= -k_8 \cdot nS + k_7 \cdot 2 \cdot npS_{npS}
\end{align*}
\]

The network is equivalent to the differential equations and describes the structure of the mathematical model, also called its topology. Together with specific values for the dynamic parameters k1, ...
k8, initial values for each species $R_0 = R(t=0)$, ..., $nS_0 = nS(t=0)$, and information about the external Epo stimulation for each time point, $EPO(t)$, the model can be integrated, i.e. the concentration-time profiles can be determined computationally as shown in Figure 2. Table 1 provides an overview of important model elements.

**Figure 2: Model trajectory with measured data and main user interface.** Left: The experimental data (blue) of the phosphorylated receptor is shown over time with error bars corresponding to one standard-deviation. The parameters of the JAK/STAT model have been calibrated in order to minimize the distance of the model trajectory (red) to the data. The model cannot be rejected as a wrong model as long as the $\chi^2$ (Chi-Square) value does not exceed a threshold value depending on the desired significance level (see section 3.3). As a rule of thumb, the Chi-Square value should not exceed the number of fitted data points $N$. Whether the model is „true“ cannot be concluded, as every mathematical formulation of process in nature is only an idealization. Right: Shown is the main user interface with two windows comprising the observed variables (left) and internal variables (right) of a fitted JAK/STAT model.

<table>
<thead>
<tr>
<th>Element</th>
<th>Name</th>
<th>Description</th>
<th>Example</th>
<th>JAK/STAT model</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Dynamic variable</td>
<td>All model species which occur as a reactant or product</td>
<td>Cytoplasmic and nuclear proteins</td>
<td>$R$, $pR$, $iR$, $S$, $pS$, $pS_pS$</td>
</tr>
<tr>
<td>U</td>
<td>Driving input variable</td>
<td>Externally controlled species only occurring as modifiers</td>
<td>Ligands, drugs</td>
<td>$EPO$</td>
</tr>
<tr>
<td>Y</td>
<td>Observation variable (observable)</td>
<td>Sum of dynamic variables that can be measured experimentally</td>
<td>Proteins detected e.g. by Western blotting</td>
<td>$pS_{obs} = pS + 2 \times pS_pS$</td>
</tr>
<tr>
<td>Z</td>
<td>Derived variable</td>
<td>Dynamic variables or functions thereof of special interest</td>
<td></td>
<td>$total_nS = nS + 2 \times npS_{npS}$</td>
</tr>
</tbody>
</table>
### Table 1. Overview of important model elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Name</th>
<th>Description</th>
<th>Example</th>
<th>JAK/STAT model</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Compartment</td>
<td>A distinct reaction space</td>
<td>Cell compartments like cytoplasm or organs like liver</td>
<td>Cytoplasm, Nucleus</td>
</tr>
<tr>
<td>k</td>
<td>Dynamic parameter</td>
<td>A parameter occurring in a reaction</td>
<td>Rate and dissociation constants</td>
<td>(k_1, k_2, k_3, \ldots)</td>
</tr>
<tr>
<td>(X_0)</td>
<td>Start value parameter</td>
<td>Value of dynamic variables at time zero</td>
<td>Initial protein concentration before stimulation</td>
<td>(R(t=0), S(t=0))</td>
</tr>
<tr>
<td>S</td>
<td>Scaling parameter</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Variables are time-dependent quantities, whereas parameters have a time-independent value which can be calibrated by PottersWheel.

### 1.2 Key modeling questions

Often, the underlying model structure of a real biochemical network is unknown. In addition, usually only a subset of system variables can be measured. And since experiments are time-consuming and expensive, the question arises which measurements will provide the most information. This setting leads to several challenges:

- **Model discrimination**: How can we distinguish competing model hypotheses?
- **Parameter calibration**: Which parameter values are most probable given the experimental data? Do some parameters compensate each other’s effect?
- **Experimental design**: What is most informative next experiment given the research question?

A key approach to address the above challenges is multi-experiment fitting, where a single model is fitted to several experiments at the same time.
2. Overview and notation

The basic use of PottersWheel covers seven phases from model creation to the prediction of new experiments as shown in Figure 3. In the following we summarize all necessary steps for each phase assuming a simple case scenario. The use of advanced modeling techniques is described in the PottersWheel manual and in the online help, both available at www.potterswheel.de.

**Figure 3.** The seven modeling phases from model creation to the design of new experiments.

**Notation:**

Menu items are identified using the scheme „Window | Menu-main item | menu-subitem“ where Window corresponds to either PW for the PottersWheel main user interface, EQ for the equalizer, and MD for the model designer. To create a new model, use e.g. the menu-item „PW | Models | New model...“ If a button is available as a shortcut, its symbol is displayed in the page margin.

Configuration settings have a special notation which equals their command line usage for experienced users. E.g. „config.integration.useFastIntegration“ corresponds to the checkbox „useFastIntegration“ in the dialog „Configuration | Fields | integration“. To open the configuration
Model creation

dialog itself, use „PW | File | Configuration dialog“, the corresponding button, or the command pw-Configuration.

A type-writer font is used to refer to PottersWheel functions, e.g. `pwAddData`. Further information is available within the online function reference at www.potterswheel.de or using „help <functionName>“ in the Matlab command window, in this case „>> help pwAddData“.

In order to quickly search for a function given a keyword, use

```
>> pws keyword
```

3. Model creation

Based on a hypothesis regarding a biochemical reaction network, a mathematical model can be created and loaded into PottersWheel. We describe here the approach using first the model creation wizard and then the visual model designer. It is assumed that all reactions follow mass action kinetics. As shown in Figure 4, as a preparation the different variable classes must be identified, especially the dynamical, driving input, and observation variables.

![Variables classes diagram](image)

**Figure 4. Variables classes.** The concentration of dynamic variables (blue circles) is not externally controlled like driving inputs (yellow box). Observables (red boxes) are the sum of one or more species, potentially multiplied by the number of related bounding-partners, e.g. `pS_obs = pS + 2 * pS_pS`. Derived variables (green box) have no special experimental meaning, but can be plotted seperately within PottersWheel, e.g. to focus on a subset of interesting variables.

1. **Starting the model wizard.** Open the model creation wizard using “PW | Models | New
model...” Enter a model ID using only letters, numbers, and the underscore and select a reaction based model type.

2. **Number of variables and parameters.** Specify the number of dynamic variables X. These are all species in the model which are not controlled externally. Note that further species can be added later. Then, specify the number of dynamic parameters k comprising e.g. rate constants, the number of compartments C, e.g. for cytoplasm and nucleus, driving inputs U, i.e. experimentally controlled species as e.g. a drug concentration, and the number of observables Y corresponding to measurable species. See Table 1 for further details.

3. **IDs. Proceed with “Create and open in designer”**. A sequence of dialogs appears, where the ID of each variable and parameter can be entered, if not the default ID as for example X1, X2, etc. is used. The only non-optional entries are the right hand sides for the observables, usually a single or the sum of a few X variables, e.g. “X1” or “X1 + X2”. The recommended naming convention is described in section 3.9.

4. **Reactions.** Afterward, the reaction designer opens and a set of reactions can be added. For each reaction select reactants, products, possible modifiers like enzymes, and involved parameters. In many cases the reaction type can be left to “A (automatic)”, where the reaction rate is assumed to be the product of all reactants and modifiers multiplied by the first parameter. If two parameters are given, the reaction is assumed to be reversible and the back-reaction rate is given by the product of all products and the second parameter. Checkmark “None” if no reactant, product, or modifier takes part in the reaction.

5. **Model designer.** After closing the dialog, the model appears in the model designer. Here, the position of each species and reaction can be changed via drag and drop. The context menu of each object allows specifying or refining object properties like color, initial value, and fit setting.

6. **Model editor.** Saving (PW | Model | Save) and opening the model file in the Matlab text editor (PW | Model | Edit model file) allows to investigate the textual representation of the model. As shown in Figure 5, changes in the editor are reflected synchronously in the model designer and vice versa using the buttons “reload” and “save” in the visual model designer.

7. **Model repository.** The model is listed in the repository list of the main user interface. Now, either experimental data can be added (see section 3.2) or synthetic data is simulated.

Advanced modeling techniques comprise

- **rule based modeling**, where placeholders are used within a reaction representing several modification states of the same species. PottersWheel automatically creates all reactions that match the used pattern (see section 3.9).
Data import

- **multi-compartment models**, where species may be transferred from one compartment to another, as e.g. the STAT molecule crossing the membrane from cytoplasm to the nucleus (see section 3.10).

![Visual approach vs. Text based approach diagram]

**Visual approach**
- Graphical user interface (GUI)
- No Matlab knowledge required
- Steep learning curve

**Text based approach**
- Custom Matlab scripts and programs
- Batch processing
- Comprehensive and documented Application Programming Interface (API)

**Figure 5. Visual and text based approach.** PottersWheel functionalities can be accessed either using graphical user interfaces or by command line to support batch processing and creation of custom Matlab scripts based on the PottersWheel function library. The model designer is for example tightly connected to the textual representation of a model and the user can decide any time to change the model in the designer or editor.

4. **Data import**

External data saved in text or Excel files can be added to a loaded model. In contrast to Excel files (*.xls), in text files (*.txt) all lines without numeric data have to be marked with a “#” sign as the first character and columns are separated by tabulator. A PottersWheel data file encodes one experiment, consisting of one or more experimental conditions (stimuli), measured using the same (absolute or relative) units.

- **File preparation** | As shown in Figure 6, each measurement file contains two sections: **meta information** and a **data matrix**. The meta information comprises the used data format version, individual comments, formatted information about the experimental setting, and the column names of the data matrix.
Data import

Figure 6. Structure of a PottersWheel data file. The header of the data file contains meta information about the experimental data consisting of the PottersWheel data format version, comments, the experimental stimulation settings, and the data column names, whereas the data matrix comprises the values for time, stimulus number and the measurement observables. In the shown example the stimulation settings describe a first input function starting with 0 (no stimulation) and switching to 1 (full stimulation) at time 0 with no subsequent changes expressing a constant stimulation. It is possible to define more input functions, which are distinguished by their stimulus number and will be assigned to the experimental data in the stimulation column. Here the second experimental condition is lower Epo stimulation comprising a withdrawal after 5 minutes. Only rows starting with the keyword „Input“ will be recognized as information for the stimulus input function. In order to be recognized as a time and stimulus column the column names have to start with the keywords „Time“ and „Stimulus“ respectively (red marks). Additional text will be ignored unless a time-unit definition. The data points with no values are denoted by „NaN“ (Not a number, blue mark).

4.1 Meta information

Excel cells in the first column in the meta information section may not consist of a single number e.g. “1”.

► Data format | The first line of the meta information is obligatory and describes the used data format. Here, „PottersWheel Data Format 2.0.55“ denotes that PottersWheel version 2.0.55 or higher is required to load the data. If you use this declaration in your data file you have to use the current conventions described in this section.

► Comments | Afterwards, individual comments can be used to describe the measurements and will be ignored by PottersWheel unless a cell in the first column contains the keyword “Input” or “MCode” (case insensitive).
Data import

- **Experimental treatment** | In the input paragraph, the experimental setting can be defined represented by driving input functions for all externally controlled system species. The expected cells of an “Input” line are:

  “Input” | <Name> | <Stimulus> | <Type> | <Time points> | <Values>

In order to improve readability, a header line above the input paragraph may be used as shown in the example sheets. This line will be ignored by PottersWheel. The <Name> text field is the name of the driving input function, often the same as in the model (see pwAddU). The input unit is optional and can be specified after the name in round brackets e.g. „Epo (ng/ml)“. The <Stimulus> integer field defines the stimulus number of the described setting which will be used later to assign each data point to a certain stimulus. The <Type> text field denotes the type of the input function, e.g. a “linear” or “steps” interpolation. The two fields <Time points> and <Values> comprise a semicolon separated list of numbers defining points of the interpolation: When has the driving input which value? The number of time-points and values is arbitrary but must be the same. Specification of time point t = -1 is necessary to define from which value the step function starts in the past:

For example, a pulse-wash experiment of ten minutes length and an amplitude of 5 would be formally written with type „steps“, time points „-1; 0; 10“ and (step-) values „0; 5; 0“. Please note that Excel requires a blank before a minus sign, so that the cell content cannot be mistaken as a mathematical formula.

This approach assumes that the input is given as an analytic function. If measured values exist for the external species, they are saved as a normal data column and mapped during data loading to the corresponding input variable U.

- **Combining several driving input species** | Experimental designs often consist not only of one input species whose different time course give rise to different experimental conditions but of multiple input species which are combined for a specific treatment e.g. receptor ligand stimulation plus kinase inhibitor treatment. The number of input species and their possible variations specifies the number of possible experimental treatments (Figure 7A). Since each input function row of the PottersWheel data file contains only information about one input species such combinatory inputs are mapped by defining each treatment for each input species separately (Figure 7B).
Figure 7. Example with two driving input species and 6 different treatments. In the shown experimental design it is assumed that the biological system at hand is activated by a ligand under three different conditions (off / low concentration continuous / high concentration continuous) and can be inhibited (two different conditions: on/off). Regarding all possible combinations this leads to a typical stimulation scheme with 6 different experimental conditions (stimulus numbers), each a combination of the two input species (A). In the PottersWheel data file these 6 experimental conditions are specified for each of the two input species separately. Thus each stimulus will be described by two driving input functions (Ligand + Inhibitor) (B).

Several treatments in one file | Is it better to use two Excel files to describe two experimental settings or one Excel file with two stimulations? Answer: Two Excel files can always be used but one file with two stimulations can only be used if both experimental settings have been measured using the same (relative or absolute) units. Then, two data values of the Excel file for the same species can be compared. This situation occurs e.g. when different experimental conditions are measured using the same Western blotting gel.

Data column names | Finally, the last row of the meta information should be the column names of the measured data. Default column names are „Time“ and „Stimulus“ for specifying time points and stimulus numbers (see section 4.2). Column names for the measured species and their measurement errors can be added as required. Allowed characters for the species names are A-Z, a-z, 0-9, blank, and the underscore. The measurement unit is optional and is specified after the species name in round brackets, as in „pSTAT (nmol/l)“. Column names for measurement errors need the prefix “stdCol-“ + “species name” to be recognized as measurement error column of the species (see section 4.3).

4.2 Data matrix

The first line with a number in the first column marks the beginning of the data matrix. One row of the data matrix represents the measured data e.g. a protein concentration for one time-point and one stimulus number. One column comprises the time points and has a column name starting with „Time“, again with an optional specification of the time unit in round brackets, e.g. „Time (min)“. Another column named „Stimulus“ refers to the stimulus number. If no stimulus column is specified, all lines are assumed to belong to stimulus 1.
The remaining columns contain the measured data. Missing values can be denoted as "NaN". It is not allowed to have a "NaN" (not a number) entry in the upper left corner of the matrix. This can be avoided by using "Time" as the first column.

### 4.3 Measurement error

Each data-point has an individual weight, i.e. importance, during parameter calibration. Data-points with a larger uncertainty, i.e. standard deviation, “attract” the model trajectory less strongly. Assuming Gaussian distributed errors, the weight is proportional to the inverse standard deviation. The statistical interpretation of a fit based on the chi-square value strongly relies on realistic standard deviations. They are either given explicitly in the data matrix using the prefix “stdCol-” in the corresponding column name, e.g. “stdCol-pEpoReceptor”, are determined using an error model as given in the model file, or will be estimated by PottersWheel based on the measured data (see pwAddData). Please note that PottersWheel currently assumes Gaussian distributed errors, so that data with a different error distribution must be transformed accordingly (11).

### 4.4 Creating a model-data couple

After loading the model, use “PW | Data | Add data” to add the data file to the currently selected model. PottersWheel will import all given data columns to the model observables. If the columns in the external file differ from the observable IDs, a mapping dialog appears. Here, the mapping has to be specified by the user and can be saved for automatic name mapping in the future.

### 5. Fitting

All unknown model parameters comprising start values $X_0$, dynamic parameters $k$, and scaling factors which are not marked as “fix” in the model definition file or in the “PW | Couples | Fit settings” dialog will be optimized by PottersWheel based on an initial guess until the experimental measurements match the model trajectories as best as possible, i.e. to minimize the “Chi-Square” value, given as

$$
    \chi^2 = \frac{d_1^2}{\sigma_1^2} + \ldots + \frac{d_N^2}{\sigma_N^2}
$$

with $d_i$ being the distance between model trajectory and data point $i$, and its standard deviation (see Figure 6). With multi-experiment fitting, it is possible to fit several data sets at once, which is more challenging for the model, i.e. unsuitable models are easier to detect and to rule out. The parameter calibration procedure has many knobs to be twiddled depending on the given model and data. A
Goodness-of-fit

good starting point is to use the RADAU5 (12) or CVODES (13) integrator in conjunction with the trust region optimizer in logarithmic parameter space.

Figure 8. Parameter calibration. Given an initial guess, unknown model parameters, e.g. $k_1$ and $k_2$ (panel B), are changed during optimization in order to decrease the distance of N measured data points $d_1$, ..., $d_N$ to the model trajectory (red line), quantified using the Chi-Square value (see text).

1. **Integration settings.** Open the configuration dialog and change “config.integration.integrator” to either RADAU5 or CVODES.

2. **Assembly creation.** Select one or more model-data-couples in the upper list of the main PottersWheel user interface. Use “PW | Couples | Combine” to combine the selection into one assembly which appears in the lower list.

3. **Fit settings and optimizer.** Choose “PW | Couples | Fit settings” to define which parameters should be fitted. Then, open the equalizer and select “EQ | View | Fitting”. Here choose e.g. the trust-region optimizer in logarithmic parameter space.

4. **Applying a fit.** Arrange all figures using “EQ | Figures | Arrange primary figures” and start fitting (Button: Single fit). Depending on the model size, the fit will take a few seconds up to several minutes. The calibrated model can be saved using “PW | Couples | Save fitted model”.

6. **Goodness-of-fit**

The quality of a fit is characterized by the Chi-Square value (see section 3.3). As a rule of thumb, for N fitted data points and p calibrated parameters, the Chi-Square value should have a similar value as N-p or at least N. Statistically this is expressed in a p-value above a significance threshold of e.g. 0.05. For lower p values, the model is not able to explain the data and has to be refined (section 3.5), the standard deviation of the data points should be larger, or the used fitting strategy was not successful and the fit was trapped in a local minimum. A model-fit usually depends on the initial guess for the parameters, as depicted in Figure 8. If the Chi-Square value of each fit is similar, the model
Goodness-of-fit

might suffer from a so-called non-identifiability where several parameters compensate each other’s effect, e.g. an increased value for $k_1$ and a decreased value for $k_2$ may lead to the same observation time-profile.

Figure 9. **Fit-sequence.** Depending on the initial guess, the optimization procedure may end up in different positions in parameter space if the model comprises non-identifiable parameters.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Abbreviation</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square value</td>
<td></td>
<td>(see Text)</td>
</tr>
<tr>
<td>Number of fitted data points</td>
<td>$N$</td>
<td>$N$</td>
</tr>
<tr>
<td>Number of fitted parameters</td>
<td>$p$</td>
<td>$p$</td>
</tr>
<tr>
<td>$N$ based Chi-Square test</td>
<td>$p\text{Value}_N$</td>
<td>$1 - \text{Cdf}(N)$</td>
</tr>
<tr>
<td>$N$-$p$ based Chi-Square test</td>
<td>$p\text{Value}_N$</td>
<td>$1 - \text{Cdf}(N-p)$</td>
</tr>
<tr>
<td><strong>Rule of thumb:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reject model, if:</td>
<td></td>
<td>$&gt; 2 \cdot N$</td>
</tr>
<tr>
<td><strong>Statistical cases for significance level 0.05:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Model is rejected, if:</td>
<td></td>
<td>$p\text{Value}_N &lt; 0.05$</td>
</tr>
<tr>
<td>2. Model might be rejected, if:</td>
<td></td>
<td>$p\text{Value}_{N-p} &lt; 0.05 &lt; p\text{Value}_N$</td>
</tr>
<tr>
<td>3. Model is not rejected if:</td>
<td></td>
<td>$0.05 &lt; p\text{Value}_{N-p}$</td>
</tr>
<tr>
<td><strong>Advanced criteria for model comparison:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log-Likelihood</td>
<td>$LL$</td>
<td>$-0.5 \cdot -N/2 \cdot \log(2\pi)$</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>$AIC$</td>
<td>$-2 \cdot LL + 2 \cdot p$</td>
</tr>
<tr>
<td>Corrected AIC</td>
<td>$AICc$</td>
<td>$AIC + 2 \cdot p \cdot (p+1) / (N-p-1)$</td>
</tr>
<tr>
<td>Bayesian Information Criterion</td>
<td>$BIC$</td>
<td>$-2 \cdot LL + p \cdot \log(N)$</td>
</tr>
</tbody>
</table>

Table 2. **Goodness-of-fit criteria.** Different measures exist to characterize statistically the fitting quality. PottersWheel displays the value of each criterion when trajectories are plotted, e.g. after a fit or a manual parameter change. Cdf(.,$f$) denotes the cumulative density function of a chi-square distribution with $f$ degrees of freedom.
1. Calculating the goodness-of-fit. After each fit or parameter change, PottersWheel displays several statistical measures on the Matlab command line characterizing the fitting quality (see Table 2). We recommend applying a so-called Chi-Square test using a significance level of e.g. 0.05. PottersWheel determines two p-values, pValueN and pValueN-p. If the larger pValueN is already below 0.05 the model can be rejected or the error bars of the data points have been underestimated and should be larger. If the pValueN is above 0.05, but the smaller pValueN-p is below 0.05, a decision whether to reject the model is difficult. If both p-values are above the threshold, the model has not to be rejected. Further measures are the Akaike (14) and Bayesian information criterion (15) AIC, AICc, and BIC to compare competing models. Their proper statistical interpretation is beyond this introductory manuscript.

2. Improving the fit-quality. To achieve a better fit, try one or more of the following:
   a. Improve the fitting accuracy by reducing the tolerance for changes in the Chi-Square value and in the parameters to 1e-8 and increasing the maximum number of iterations (config.optimization.trustregion.TolFun / TolX / maxIter).
   b. Switch between the RADAU5 and CVODES integrators (config.integration.integrator).
   c. Use the Matlab symbolic math toolbox to enable PottersWheel to determine analytical derivatives for integration and optimization (config.integration.useJacobian and config.optimization.useJacobian).
   d. Use the sliders of the equalizer to manually find a better initial guess for the parameters.
   e. Fit with simulated annealing (asamin (16,17)), scatter search (SSM (18)), or a boosted fit, selectable in the fitting view of the equalizer (EQ | View | Fitting).
   f. Use trust-region in logarithmic parameter space within a fit-sequence from equally sampled initial guesses throughout the parameter space (config.optimization.useQRNG).

3. Pair-wise linear identifiability analysis. Based on a valid fit, the so-called “identifiability” of the calibrated parameters should be investigated. In many cases the amount of measured data is not sufficient to unambiguously determine parameter values, i.e. a forced change to one parameter may be compensated perfectly by other parameters leading to the same goodness-of-fit. Using “EQ | View | Fit sequence analysis”, a linear identifiability analysis is applied using the fit-sequence F2 with n > 100 fits, preferably > 300 fits.
Model refinement

4. **Multivariate non-linear identifiability analysis.** The MOTA algorithm can be used to identify groups of non-linearly related parameters after a fit sequence has been applied (19). The PLE method provides a means to distinguish between practical and structural non-identifiabilities (20). In the former case, a better quality of measurements may resolve the identifiability problem. For the structural case, the model should be re-parameterized or additional species should be measured, preferably under new experimental conditions. PLE is available under “EQ | View | Single Fit Analysis”.

7. **Model refinement**

If the model structure is not able to explain the experimental measurements, create a set of physiologically reasonable alternative models. In order to avoid redundant model paragraphs and copy-and-paste errors, this could be done using a common core-model which is the same for all variants. Then, „daughter“-models are created (see `pwGetEmptyModel`). Each model variant should be fitted to the data, preferably using batch processing strategies based on PottersWheel macros (see 3.8). As a starting point to envision suitable model variants, use the PottersWheel equalizer to understand the dynamic behavior of the original system (see 3.6).
Figure 11. Detailed sensitivity analysis. PottersWheel calculates several characteristics of the trajectory of a species, e.g. time and value of its maximum. The shown example displays the time-profile of the activated JAK/STAT receptor pR. The inlet depicts the dependency of a characteristic value on a changed initial concentration or parameter value, i.e. the sensitivity, in words „What is the relative increase of a trajectory property when a parameter is increased?“. A sensitivity of 1 corresponds (in a linear approximation) to a doubled trajectory property for a doubled parameter value. R0 denotes the initial value of the free receptor, intR, deactR, actR are rate constants for the internalization, deactivation, and activation, i.e. phosphorylation of the receptor. The time of peak (yellow) is not affected when R0 is increased. However, increasing any of the rate constants leads to an earlier peak. The signal duration (red) is here defined as the time when the trajectory falls below a 20% threshold compared to the maximum. It will be smaller for increase intR and actR, but will be larger for increased deactR. This was expected, since a slower deactivation will lead to a more sustained signal.

A mathematical model can be used for example to display the concentration time-profile of unobserved species, to determine sensitive parameters which could serve as a target within a clinical setting, and to calculate further model characteristics like the half-life of a species. A common analysis could be:

1. **Open the PottersWheel equalizer.** Combine one or more model-data-couples into the lower list of the main user interface. Then open the equalizer using “PW | Assembly | Open equalizer”. The PottersWheel equalizer is the key user interface for interactive model analysis.

2. **Changing parameter values.** Each parameter can be changed using a pair of sliders. The right slider changes the order of magnitude and the left slider can be used for fine-tuning – it changes the mantissa. Alternatively the parameter value can be entered directly using the
Analysis and model prediction

3. **Displaying A, U, X, Y, Z variables.** PottersWheel allows for plotting different classes of variables at the same time, which are algebraic variables $A$, driving inputs $U$, dynamic variables $X$, observables $Y$, and derived variables $Z$ (see 3.1). Use the corresponding toggle buttons.

4. **Grouped and hidden variables.** Using “EQ | View | Plotting”, several variables can be grouped and will be shown within the same subplot, offering direct comparison of trajectories. Uninteresting variables can be hidden. The manual plot settings can be saved to hard disk or into the original model file for later use.

5. **Phase space analysis.** Using “EQ | View | Phase space”, the concentration profile of two variables can be plotted against each other in a phase space diagram. For related variables, a potentially complex dynamic behavior can often be seen in a simpler form, e.g. a circle or a line.

6. **Figure handling.** The figure menu provides several functions to arrange or copy figures and subplots. Especially when working with many trajectories, it can be useful to copy a single subplot into a separate figure. Use the figure saving button to store the current figures into various formats which can be specified in the configuration, e.g. “config.plotting.saveJPG”. Alternatively, activate the figure toolbar and menu for individual saving (config.plotting.toolbar / menuBar).

7. **Sensitivity analysis.** Using “PW | Models | Sensitivity Analysis” in the main user interface, several sensitivity analyses are available to determine how strong a variable or a characteristic value like the time of the maximum depend on a parameter or an initial value (see Figure 10).

8. **Residual analysis.** Using “EQ | View | Single Fit Analysis” in the equalizer, the distribution of residuals can e.g. be compared to a Gaussian distribution. A suitable model should not suffer from systematic errors.

9. **Chi-Square landscape.** Again using “EQ | View | Single Fit Analysis”, the Chi-Square can be calculated systematically in 2 or 3 dimensions of the parameter space around the current position. This is a useful visualization of non-identifiability manifolds.
9. Design of new experiments

An experimental setting corresponds to specific characteristics of the driving input functions and initial concentrations. In the JAK/STAT system the concentration of EPO is controlled experimentally. The driving input designer allows investigating the effect of a continuous, ramp, or pulse stimulation as shown in Figure 12 in combination with varying initial concentrations using the equalizer (EQ | View | Driving input designer). In order to discriminate competing model hypotheses, the next experiment should have as different observable time-profiles as possible which can be verified computationally.

![Figure 12. Stimulations and the PottersWheel input designer. Left: Depending on the used cell-system continuous, pulsed and ramp ligand concentrations, here of an enzyme E, may be feasible. In order to obtain different responses by the system, it is important to apply qualitatively different driving inputs, i.e. stimulations. Right: The driving input designer allows to test the model under different experimental conditions, e.g. the impact of an oscillatory stimulation.](image)
10. Model exploration in the visual model designer

As displayed in Figure 5, reaction based dynamical models can be created and modified either directly in the model definition file or using the visual model designer. The latter also allows exploring the dynamic properties of the modeled system by hovering the mouse over a species or a reaction anchor. The corresponding trajectories or reaction fluxes are displayed in a subfigure as shown in Figure 13.

![Figure 13. Trajectory visualization in PottersWheel model designer.](image)

The inlet displays the trajectory of species B when the mouse hovers over the corresponding object in the model layout. The flux of a reaction can be displayed by moving the mouse on top of a reaction anchor.

11. Rule based modeling and naming convention

Suppose an enzyme E in n different states $E_1, ..., E_n$ which triggers the reaction “A to B”, independent of the current state of E itself, leading to n similar reactions. If protein A possesses m different states, the number of reactions explodes to $n^m$, which is called combinatorial complexity (21,22). The corresponding model definition file would be difficult to maintain and is prone to copy-and-paste errors. Therefore, PottersWheel supports rule based modeling, where reactions may contain placeholders that are replaced automatically by all matching species. This approach requires that variable IDs follow the recommended PottersWheel naming convention:

1. Basic, i.e. unmodified and unbound species start with a capital letter, e.g. “Erk”.
2. Modifications are lowercase prefixes, e.g. “ppErk”.
3. Complexes-bounds are denoted by an underscore “_”.
Implicit and explicit multi-compartment modeling

4. The order of species in a complex is always the same, i.e. “ppMek_Erk” is not equivalent to “Erk_ppMek”.

For further information see pwAddR and pwTutorial Rule based modeling.

12. Implicit and explicit multi-compartment modeling

Biochemical processes often benefit from compartmentalization, e.g. in order to increase locally the pH-value required to degrade waste products. Therefore, many biochemical models comprise compartmental information, essentially the volume of each compartment and the classification of each species to exactly one compartment. As discussed in the SBML specifications (see www.sbml.org), the kinetic rate of a reaction with reactants and products in different compartments, has to be expressed in amount-based units per time instead of concentration changes per time. This is due to the fact, that the loss of a single molecule from a small into a 100 times larger compartment results in a 100 times stronger concentration change in the smaller compartment. The additional burden to determine the amount of a kinetic rate is usually handled by multiplication with the size of the corresponding volume. Note that in reversible reactions from compartment C1 to C2, in most cases the part of the kinetic rate comprising reactants is multiplied with c1, the size of C1, and the part comprising the products is multiplied with c2, e.g.

\[ k_1*r_1*c_1 - k_2*p_1*c_2 \]

The question arises whether the modeling framework can take care of the compartment size multiplication automatically, so that the user avoids specifying redundant or repetitive information, as in a set of reactions all taking place in the same compartment. At the same time, the readability of the kinetic law would be improved and correspond closer to the expression in a biochemistry textbook. In fact, PottersWheel is able to correct for the compartment sizes automatically for most models which is called implicit multi-compartment modeling. In the other cases, when importing an amount based model, or to have full control, the compartment correction can be turned off by the following line in the model definition file:

```
m.amountBasedRates = true;
```

Since SBML models are amount-based, an imported SBML model will comprise the above line below the meta information paragraph corresponding to explicit multi-compartment modeling.
13. Configuration

To allow for a detailed individual configuration of PottersWheel and the applied algorithms, hundreds of settings can be adjusted either within the configuration dialog or directly from command line. Elements in the configuration dialog share the same name as required for command line usage, which enables the user to quickly becoming an experienced command line user and thereby accelerating creation of custom Matlab script, e.g. for batch processing. Changes to entries are affecting PottersWheel instantaneously. Closing of the dialog is not necessary, but consequently canceling is not possible either. Use the file saving and opening menu to store preferred settings for use. Within own programs or in the command line, all configuration settings are accessible, e.g.:

```
>> config = pwGetConfig;
>> config.integration.useJacobian = true;
>> pwSetConfig(config);
```

This is especially useful if you are working with PottersWheel macros, since all settings, modeling, fitting, and analysis commands are documented in the same file.

14. Macros and custom Matlab programs

Since PottersWheel is created as a Matlab toolbox, the user is not restricted to graphical user interfaces and by creating own Matlab programs employing the PottersWheel function library (API – application programming interface) he or she may gain and improve his or her competence in a major and widespread numerical framework. As exemplified for the model designer and the model editor in Figure 5, PottersWheel allows for applying mathematical modeling using a visual or text based approach, targeting the beginning as well as the advanced user. All PottersWheel functions are available as Matlab functions named as “pw...”. A macro, i.e. a script based on the PottersWheel library to fit a model to a dataset could be written as follows:

```
pwAddModel('Model1.m');
pwAddData('Experiment1.xls');
pwCombine;
pwArrange;
pwFit;
```

When saving these lines into a file named „myFittingScript.m“, each command will be executed sequentially after typing in the Matlab command window

```
>> myFittingScript
```

Using macros is not only useful to apply batch-processing but also ensures that each modeling step is documented and can be reproduced at any later time.
15. **Reporting**

Each analysis or visualization step can be added as a section to a report. The Report Designer user interface allows removing and reordering of sections, as well as displaying only those figures which are related to the section. Finally, a report can be created either as a Latex-based PDF, a Microsoft Word document, or as an HTML file. Several predefined reports can be created in the report menu (PW | Report), e.g. to summarize the model structure.

16. **Model stiffness**

The trajectories, i.e. the time-profile of differential equation based models are approximated computationally using numerical integration. The accuracy of the integration depends on the **stiffness** of the system at hand: the stiffer the ODE system, the more difficult and time-consuming the integration (12). Most biochemical networks are stiff, which is based on very fast and slow processes happening at the same time. Specialized integrators have been developed for this situation. PottersWheel supports several of them: ode15s, ode23s, RADAU, RADAU5, and CVODES. In our experience, ode15s is the slowest but also most robust integrator. It is used automatically if the current integrator fails.

A model may be integrable for the current set of parameter values and also for the optimal set of parameters, but during optimization the model may become extremely stiff, because a certain region of parameter values may be entered. Consequently the integration fails. As a remedy, the limits of parameter optimization could be narrowed to avoid the stiff region. In addition, a deterministic optimizer like trust-region can be used instead of a more „jumping“ stochastic optimizer like asamin (16,17). Finally, re-formulation of the mathematical model may be useful in order to avoid too many extremely fast reactions.
17. Bibliography


