BIOINFORMATICS

Vol. 00 no. 00 2009
Pages 1–3

Supplementary Information
Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood

A. Raue [1,*], C. Kreutz [1], T. Maiwald [2], J. Bachmann [3], M. Schilling [3], U. Klingmüller [3] and J. Timmer [1,4]

[1]Physics Institute, University of Freiburg, 79104 Freiburg, Germany
[2]Department of Systems Biology, Harvard Medical School, 02115 Boston, MA, USA
[3]Division of Systems Biology of Signal Transduction, DKFZ-ZMBH Alliance, German Cancer Research Center, 69120 Heidelberg, Germany
[4]Freiburg Institute for Advanced Studies, University of Freiburg, 79104 Freiburg, Germany

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

This supplement is intended to illustrate technical details of the approach presented in the main text.

In the first part, a description of an algorithm to compute the profile likelihood $\chi^2_{PL}(\theta_i)$ is given. Subsequently, an analysis of the computational complexity of the approach for a test case model is presented. Then, an implementation of the algorithm, that was used for the analysis in the main text, is explained. This freely available implementation is embedded in the PottersWheel fitting toolbox (Maiwald and Timmer, 2008), which is freely available for academic usage as well. Finally, all steps that were performed to produce the results shown in the main text are given.

1 ALGORITHM

In the following, a brief description how to compute the profile likelihood

$$\chi^2_{PL}(\theta_i) = \min_{\theta_{j\neq i}} \left[\chi^2(\theta)\right] \tag{1}$$

is provided.

Assume a numerical optimization of $\chi^2(\theta)$ yielding a calibrated set of parameters $\hat{\theta}$. Beginning from $\hat{\theta}_i$, sample along the profile likelihood in increasing/decreasing direction of θ_i by:

1. Take an incremental step θ_{step} in increasing/decreasing direction of θ_i.
2. Re-optimize all $\theta_{j\neq i}$.

Repeat until the desired threshold Δ_α is exceeded or a maximal amount of steps is reached.

θ_{step} should be chosen in an adaptive manner, taking large steps if the likelihood is flat and small steps if the increasing of the likelihood is steep. Therefore θ_{step} should fullfil the condition

$$\chi^2(\theta_{last} + \theta_{step}) - \chi^2(\theta_{last}) \approx q \cdot \Delta_\alpha \tag{2}$$

where θ_{last} are the parameter values of the previous iteration and $q \in [0, 1]$. For an identifiable parameter at least $1/q$ steps are required until the threshold Δ_α is reached. Good results are achieved using small values of q, which facilitates re-optimization in the presence of nearby local optima. Since the likelihood might be flat in presence of non-identifiability, a maximum step size is necessary. θ_{step} can be calculated using one of the following possibilities:

- **Direct step:**
$$\theta_{step} = \{0 \dots \theta^i_{step} \dots 0\} \tag{3}$$
where θ^i_{step} is aligned to fulfill Eq. 2.
- **Progressive step:** Given the supplied optimization routine returns reliable first and second derivative of the objective function with respect to the parameters $\vec{\beta} = \nabla\chi^2|_{\theta_{last}}$ and $\mathbf{H} = \nabla^{\mathrm{T}}\nabla\chi^2|_{\theta_{last}}$, the objective function can be approximated in the neighbourhood of θ_{last} by
$$\chi^2(\theta') = \gamma + \vec{\beta}\cdot\theta' + \theta'^{\mathrm{T}}\cdot\mathbf{H}\cdot\theta' \tag{4}$$
with $\theta' = \theta - \theta_{last}$ and $\gamma = \chi^2(\theta_{last})$. A direction ϕ_{step} with least increase in likelihood can be approximated by solving
$$\frac{\partial\chi^2(\theta')}{\partial\theta'}|_{\theta'_i=const} \stackrel{!}{=} 0 \tag{5}$$
where θ' in the direction of interest i is fixed to a small constant value. Advancing in this direction by
$$\theta_{step} = c \cdot \phi_{step} \tag{6}$$
where c is aligned to fulfill Eq. 2, enables to take larger steps, hence reduces the computational effort.

*to whom correspondence should be addressed

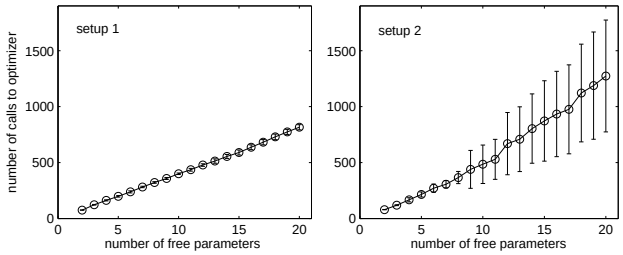

Fig. 1. Dependency of the number of calls to the optimization procedure on the number of free parameters. Both cases scale linearly with the number of free parameters. Error bars reflect variability within the set of free parameters.

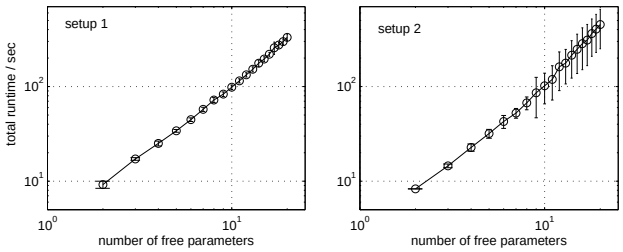

Fig. 2. Dependency of the total runtime on the number of free parameters, plotted double logarithmically. The exponent of the non-linearity is ≈ 1.56 for setup 1 (left panel) and ≈ 1.78 for setup 2 (right panel). Error bars reflect variability within the set of free parameters.

2 COMPUTATIONAL COMPLEXITY

The computational complexity of the approach depends on the efficiency of the supplied numerical optimization procedure and the shape of the likelihood. The latter depends on the specific model and the amount and quality of the experimental data that is available. Therefore it is not possible to give a general statement about this issue. Here, the dependency of the computational complexity of the approach on the number of estimated parameters is analysed for a specific test case model. It consist of twenty-one species $x_1, \ldots, x_{21}$ connected in a chain of Mass-action type reactions with dynamic parameters $p_1, \ldots, p_{20}$. We assume that: $x_1(0) = 10$ nM and $x_{i\neq 1}(0) = 0$ nM; every species can be observed $y_i = x_i$ (setup 1) or only every species with odd index can be observed $y_1 = x_1$, $y_2 = x_3$, $\ldots$, $y_{10} = x_{21}$ (setup 2); for each observable 26 measurements at time-points $t = [0, 2, \ldots, 50]$ minutes with measurement noise $\sigma^D = 0.1 + 0.1 \cdot y^D$ are available. Data is simulated using parameters $p_i = 0.5$ per minute and nM.

This test case setup allows to determine the computational complexity of the approach for differing number of parameters considered in the model. Therefore, the approach is applied to the model where only p_1 and p_2 are estimated while the remaining parameters are fixed to their true values. The number of calls to the optimization procedure and the runtime is measured for each parameter. This procedure is repeated with more and more parameters being freed consecutively, until all twenty parameters included in the model are considered. The algorithm introduced in the previous section is used, utilizing the direct step option to calculate θ_{step}. Fig. 1 shows the number of calls to the optimization procedure depending on the number of free parameters, growing linearly for both setups. Fig. 2 shows the total runtime of the approach depending on the number of free parameters, plotted double logarithmically. The total runtime is the sum of the runtimes needed for the calculation of the profile likelihood for each free parameter. Due to the increasing complexity of the optimization step, the total runtime T grows super-linearly like

$$T \sim \#p^{1.56} \quad \text{for setup 1 and like}$$
$$T \sim \#p^{1.78} \quad \text{for setup 2}$$

where $\#p$ is the number of free parameters. This illustrates the dependency on the specific experimental setup.

3 IMPLEMENTATION FOR POTTERSWHEEL

The package containing all necessary code bears the name *Profile Likelihood Exploit* (PLE) and is freely available at http://web.me.com/andreas.raue/profile/software.html. To install, download the package, extract the folder to a convenient location and include this location in the MATLAB search path. For instructions, how to set up the PottersWheel fitting toolbox see http://www.potterswheel.de.

In the following, all necessary function calls will be explained. They can be accessed from the MATLAB command line, or by a graphical user interface that is available for MATLAB version R2008a and later.

3.1 Command line usage

1. Load the desired model(s) and data-set(s) into PottersWheel and calibrate the parameters to best possible values. To get an overview about the parameters incorporated in the model, execute `pwInfo` at the command line. PLE will consider all parameters that are not fixed. If necessary use the command `pwFixParameters` to free or fix parameters.
2. Initialise PLE for command line usage by the command `pwPLEInit(false)`. All results computed in the following will be stored in a subfolder of the current working folder, bearing a name `PLE-yyyymmddTHHMMSS` with corresponding date and time according to ISO 8601.
3. To run PLE for all parameters execute `pwPLE`. For a specific parameter θ_i execute `pwPLE(i)` whereas `i` refers to the number of the parameter as listed by `pwInfo`. A number of further tuning parameters for the algorithm can be specified by `pwPLE(i, a, b, c, d, e)`, where is

   a. the number of maximal steps in increasing and decreasing direction. (default: 100)
   b. the aspired χ^2 increase of each step, measured in percentage q of Δ_α. (default: $q = 0.1$)
   c. the maximal size of a step. (default: $0.2 \cdot \theta_i$)
   d. the minimal size of a step. (default: 10^{-6})
   e. a flag that stops the sampling, if a parameter $\theta_{j\neq i}$ hits its parameter bounds as listed by `pwInfo`. (default: true)

Adjusting these values may improve results if the algorithm exceptionally fails to sample along the profile likelihood.

4. Execute `plePrint` to obtain a summary of the results, including an automatically generated flag `IDflag` stating the type of identifiability, which should be inspected visually. `LowerPL` and `UpperPL` indicate likelihood based confidence intervals $\sigma^{\pm,PL}$, `LowerHes` and `UpperHes` confidence intervals approximated by the inverse of the Hessian matrix $\sigma^{\pm,Hess}$ and `Rel.PL` and `Rel.Hess` the relative size of the confidence intervals $(\sigma^+ - \sigma^-)/(2 \cdot \theta_i)$.
5. Execute `plePlot` to generate figures showing profile likelihood versus parameter and the corresponding changes in the other parameters.
6. Execute `plePlotRelations(js)` where `js` is a vector of parameter indices to generate a scatterplot of parameters to reveal functional relations. This is most convenient for two or three indices. One index generates a histogram and if `js` is omitted, a matrix plot is produced.
7. Execute `pwPLETrajectories(i)` to plot trajectories corresponding to parameter values sampled for the profile likelihood $\chi^2_{PL}(\theta_i)$. This depicts model variability due to uncertainties in this parameter.

3.2 Graphical User Interface

Follow step 1 as described in the previous section. To initialize PLE and raise the graphical user interface execute `pwPLEInit` at the command line. All steps described in the previous section can be executed via buttons in the middle of the main window (see Fig. 3). It consists of two tables, the upper one contains information analog to `pwInfo` and allows to specify tuning values of the algorithm, as described in step 3 of the previous section. The lower one displays the results of the analysis analog to step 4 of the previous section. All results will be stored as described in step 2 of the previous section.

4 APPLICATION

The required models and data-sets to reproduce the results shown in the main text are available at
http://web.me.com/andreas.raue/profile/software.html. Extract the bundle to a convenient location and change the MATLAB working directory to this location.

Load the model calibrated to the original experimental data by executing the script `setup1.m` or load the PottersWheel repository `repo_setup1.m` (PW 1.6). Execute the script `analysis1.m` to calculate the profile likelihood for all parameters and to display the results. Execute the script `analysis1_strNonID.m` to plot the trajectories along the structural non-identifiability. Execute the script `analysis1_praNonID.m` to plot the trajectories along the practical non-identifiability. Load the model calibrated to the extended data by executing the script `setup2.m` or load the PottersWheel repository `repo_setup2.m`. Execute the script `analysis2.m` to calculate the profile likelihood for all parameters and to display the results.

Details of the function calls can be learned about inside these script file.

REFERENCES

Maiwald, T. and Timmer, J. (2008). Dynamical modeling and multi-experiment fitting with PottersWheel. *Bioinformatics*, **24**(18), 2037–2043. http://www.potterwheel.de.

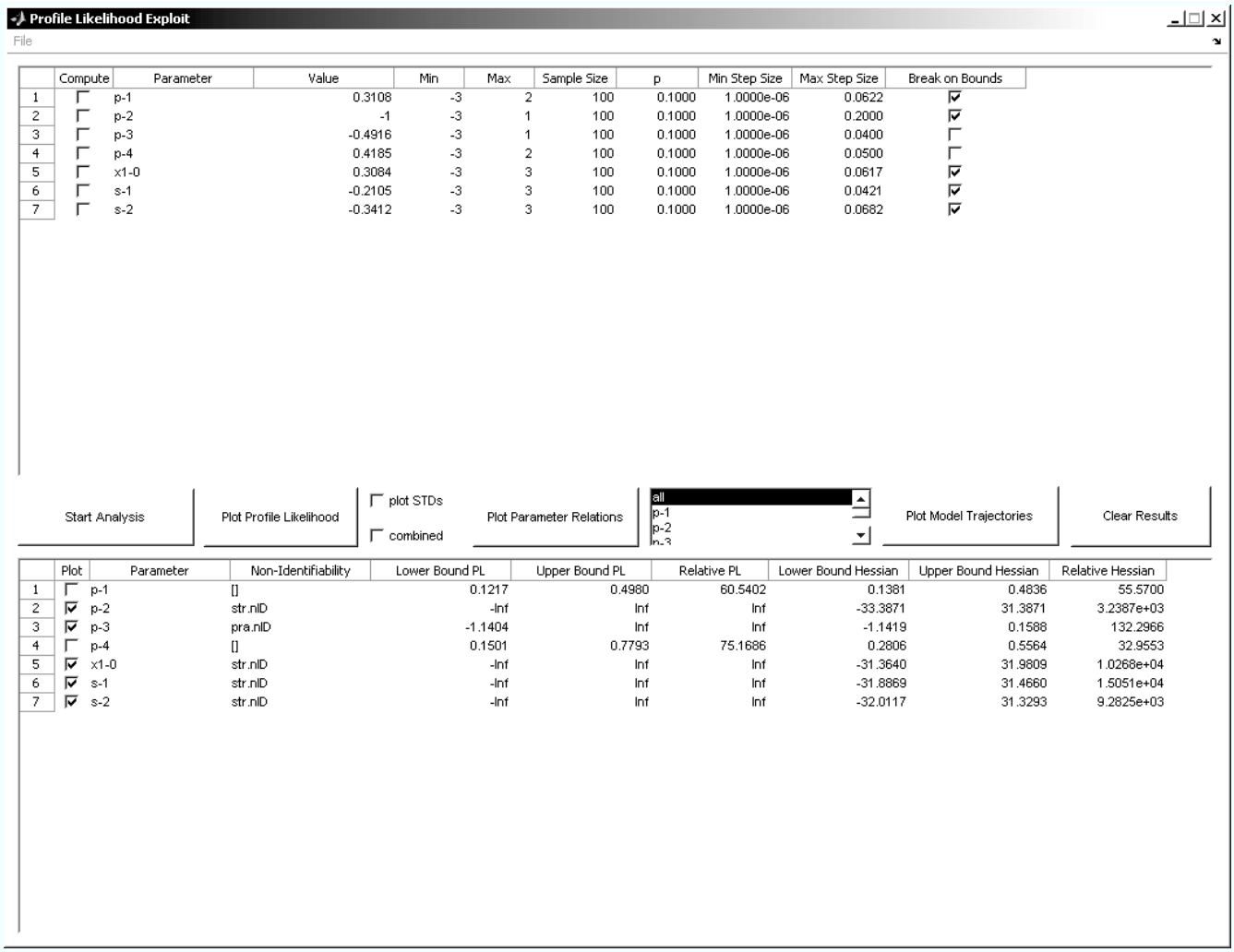

Fig. 3. The Graphical User Interface. The upper table contains information about the parameters and allows to specify tuning values for the algorithm. All necessary function calls can be accessed via buttons in the center. The lower table displays the results of the analysis. All values are given in orders of magnitude using $\log_{10}$.