Pseudodynamics vignette

David S. Fischer Anna K. Fiedler

February 23, 2018

Contents

1 Introduction

In this vignette, we describe how one can run the pseudodynamics parameter estimation on a simulated example data set provided in this repository. Moreover, we show a basic graphic interpretation of the model fits. This simple model represents a non-branching process.

2 Input to pseudodynamics

Pseudodynamics takes two groups of input observations: Cells (e.g. from single-cell RNA-seq) with cell state, time, experiment and branch label and population size estimators with a time label. We recommend preparing these two sets of input observations in two separate tables, one for the cell and one for the population size observations. Example data files can be found in *'./example/rawData/'*.

3 Run pseudodynamics

To run pseudodynamics, one has to adapt the following scripts and functions for the experimental and biological scenario in question. Pseudodynamics makes use of the Matlab toolboxes AMICI (https://github.com/ICB-DCM/AMICI/) and PESTO (https://github.com/ICB-DCM/PESTO/)

3.1 Scripts that you DO NOT explicitly call from a Matlab session

3.1.1 Objective function

Path: *'./example/llPseudodynamicsFvKS.m'*

Firstly, one has to define the objective function for the parameter estimation, in our case a negative log-likelihood function. The negative log-likelihood function defines the deviation of the simulated from the observed data. The evaluate this deviation, a forward simulation of the model given the proposed parameters has to be carried out, which is called from within the likelihood function.

3.1.2 Model definition

Path: *'./finiteVolume/models/pd_fv_syms.m'*

The model is defined in a separate function in the format of a standard input file to the Matlab toolbox AMICI. As the 'model' is the numeric approximation of the behavior of the partial differential equation (PDE) over time, this file contains all information necessary to forward simulate the system given a set of parameters and an initial condition.

We used the finite volumes approximation of the PDE through-out the manuscript and throughout the usage cases presented here. Accordingly, the model definition contains the finite volumes approximation of the partial differential equation, the resolution of the spatial grid, the parameter models and the right hand side of the ordinary differential equation that was obtained from the PDE by the finite Volume discretization. We used a spline interpolation to obtain continuous parameter fits across the cell state coordinate. This spline interpolation is also defined in this model function. Lastly, the output consists of the simulated concentration values in space and time and the simulated population size.

3.2 Scripts that you DO explicitly call from a Matlab session

3.2.1 Format data

Path: *'./example/extractDataInput.m'*

We recommend saving the input data in a standardized Matlab object before calling the function to run the parameter estimation. The input tables are loaded and preprocessed using the function `extractDataInput.m`. This file has to be adapted to the structure in the table at hand. The population size mean and squared standard error of mean is assigned to D.pop.mean and D.pop.var, respectively. The time points of population size measurements in assigned to D.pop.t. The single cell cell-state measurements corresponding to one experiment are collected in D.ind.histi and the corresponding measurement time point is saved in D.ind.tp(i). This is done in a way that the entries in D.ind.tp are ordered. Further, the relevant statistics of the area between individual sample cdfs and the cdf of the pooled samples for one time point are computed and saved in D.dist.mean and D.dist.var (see Suppl. Methods). The data struct D is then saved in a workspace, e.g. dataExample.mat, and used as input to the parameter estimation.

3.2.2 Model compilation

Path: *'./finiteVolume/models/pd_fv_wrap.m'*

The model has to be compiled using AMICI to be used within the parameter estimation with PESTO. Therefore, you have to run this model compilation script once before starting starting the estimation.

3.2.3 Run parameter estimation

Path: *'./example/mTe_fun.m'*

Lastly, the parameter estimation can be started. The function `mTe_fun.m` defines all details of this estimation such as the number of starts (initialization). The output of this function contains the set of parameter estimates for the pseudodynamics model and the likelihood of each initialization. To call this function one could for example use a Matlab script like the following:

```
for iN = 1:numberOfStarts
    mTe_fun(in);
end
```

However, also straight forward parallelization is possible by distributing (some of) the starts to different cores.

4 Output of pseudodynamics

4.1 Structure of output

The parameter estimation saves a parameter struct (and used options) in a mat-file. The parameter struct output from PESTO contains the initial user-defined fields (parameters.name, parameters.number, parameters.max, parameters.min, parameters.guess, parameters.constraints). The toolbox PESTO saves the estimation results in parameters.MS. As described in the documentation of `getMultiStarts.m`, the saved information about the multi starts is

- parameters.MS.n_starts: number of performed multistarts
- parameters.MS.par0(:,i): starting point yielding ith maximum a posteriori point (MAP)
- parameters.MS.par(:,i): ith MAP
- parameters.MS.logPost(i): log-posterior for ith MAP
- parameters.MS.logPost0(i): log-posterior for starting point yielding ith MAP
- parameters.MS.gradient(:,i): gradient of log-posterior at ith MAP
- parameters.MS.hessian(:,:,i): hessian of log-posterior at ith MAP
- parameters.MS.n_objfun(i): number objective evaluations used to calculate ith MAP
- parameters.MS.n_iter(i): number iterations used to calculate ith MAP
- parameters.MS.t_cpu(i): CPU time for calculation of ith MAP
- parameters.MS.exitflag(i): exitflag the optimizer returned for ith MAP

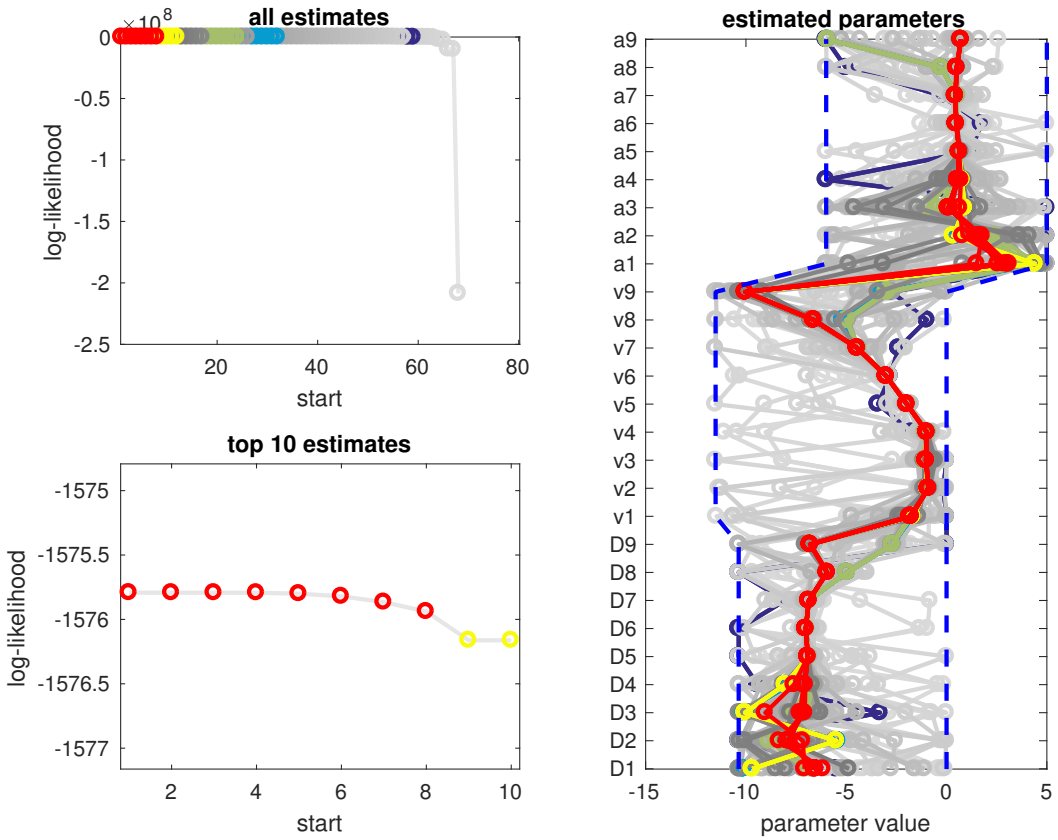

Figure 1: Log-likelihood waterfall plot created for the example using the PESTO function `plotMultiStarts.m`.

To bring the estimation results in the standard PESTO output structure, the individual saved multi start runs can be collected and put into a single struct. This can e.g. be done with a script in the form of `collectParameters.m` (Path *'./example/collectParameters.m'*). For the example at hand the collected parameters are provided in *'./example/results/parametersExample.mat'*.

Further the starts can be sorted according to the final negative log-likelihood value using `sortResultsExample` (Path: *'./example/sortResultsExample.m'*). Also, in this file the solution to the best estimate is computed. For the example at hand the results are provided in *'./example/results/resultsExample.mat'*.

4.2 Analyse output

Path: *'./example/analyzeExample.m'*

4.2.1 Assess convergence of the parameter estimation

The first step in the interpretation of the pseudodynamics output should be the assessment of the convergence of the parameter estimation. We consider a parameter estimation as converged to the global optimum if multiple initializations yield a similar optimal parameter estimate, as judged by the likelihood. One can therefore assess convergence by the existence of a plateau of the order likelihood by initialization. A corresponding (log-)likelihood waterfall plot can be created in PESTO using `plotMultiStarts.m`.

4.2.2 Check uncertainty in parameter estimates

The standard deviation of the individual parameter estimates give an idea to the uncertainty of the overall model fit. Note that one has to compute likelihood profiles for each parameter to get exact confidence intervals. The approximate confidence interval using the hessian information (stored in parameters.MS.hessian) can be computed and plotted using the PESTO function `getParameterConfidenceIntervals.m`. (If a profile likelihood was computed, `getParameterConfidenceIntervals.m` will also compute profile-based confidence intervals).

4.2.3 Interpret maximum likelihood fits

To interpret the parameter fits, we plot the continuous spline interpolation in cell state space of each parameter (drift, diffusion, birth-death rate). This spline interpolation is the same as the one defined in the model function (3.1.2)

The function `computeParameters.m`(Path: *'./example/computeParameters.m'*) evaluates the splines for a vector of values at the nodes. Theses splines are saved together with the grid and the nodes in a struct. To visualize the parameter splines one can use as in `analyzeExample.m`:

```
figure;
plot(par.grid,par.D)
xlabel('cell state')
ylabel('diffusion parameter')
legend('estimated')

figure;
plot(par.grid,par.v)
xlabel('cell state')
ylabel('drift parameter')
legend('estimated')

figure;
plot(par.gridx,par.a)
xlabel('cell state')
ylabel('birth-death parameter')
legend('estimated')
```

5 Work flow

If all files are prepared, the subsequent work flow is

1. Extract data from raw input data (Sec. 3.2.1).
2. Compile model file using AMICI (Sec. 3.2.2)
3. Start parameter estimation using `mTe_fun.m`, e.g. by running the short script provided in Sec. 3.2.3.
4. Collect all estimation results using e.g. `collectParameters.m` and sort them using e.g. `sortResultsExample.m` (Sec. 4.1).

This output then has the standard form to be further analyzed by the Matlab toolbox PESTO.

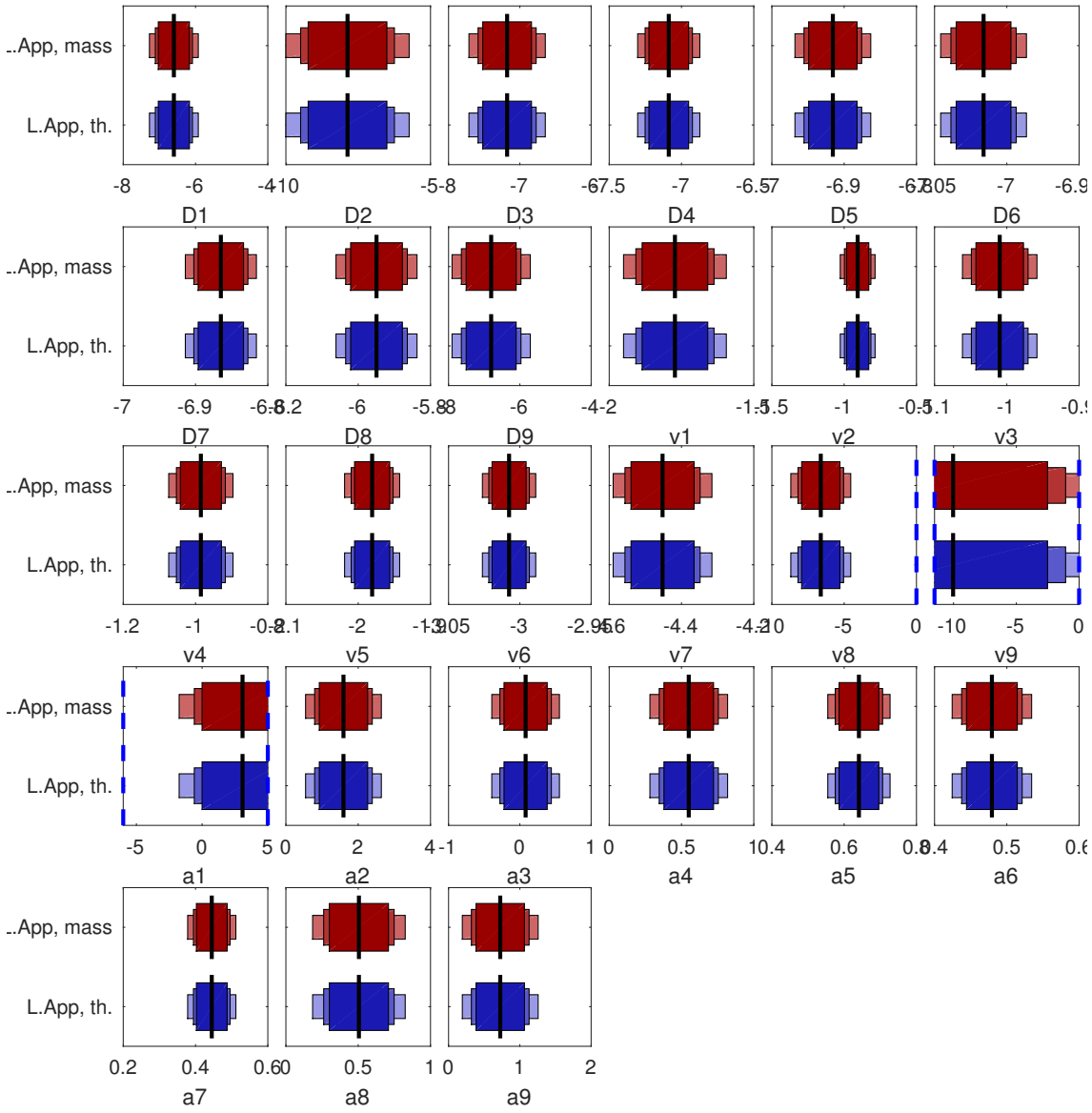

Figure 2: Approximate confidence intervals for the example computed and plotted using the PESTO function `getParameterConfidenceIntervals.m`.

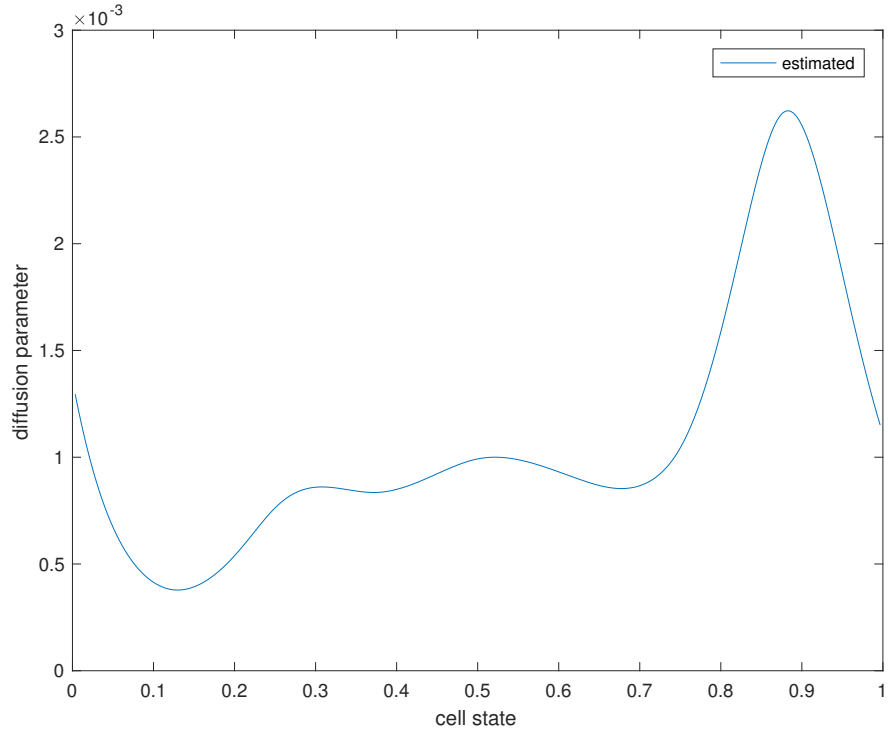

Figure 3: Estimated diffusion rate for the example data.

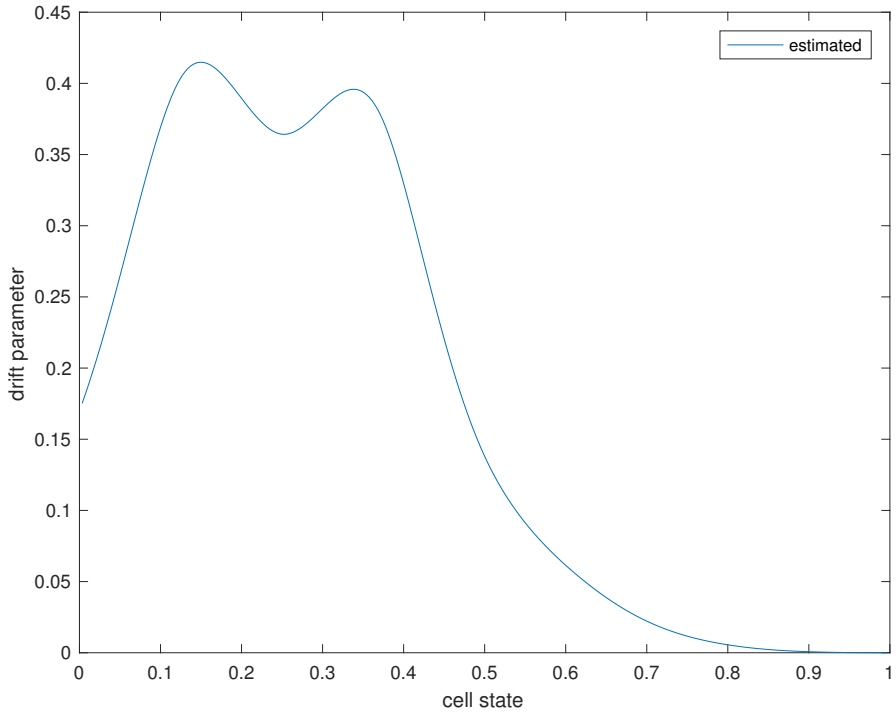

Figure 4: Estimated drift parameter for the example data.

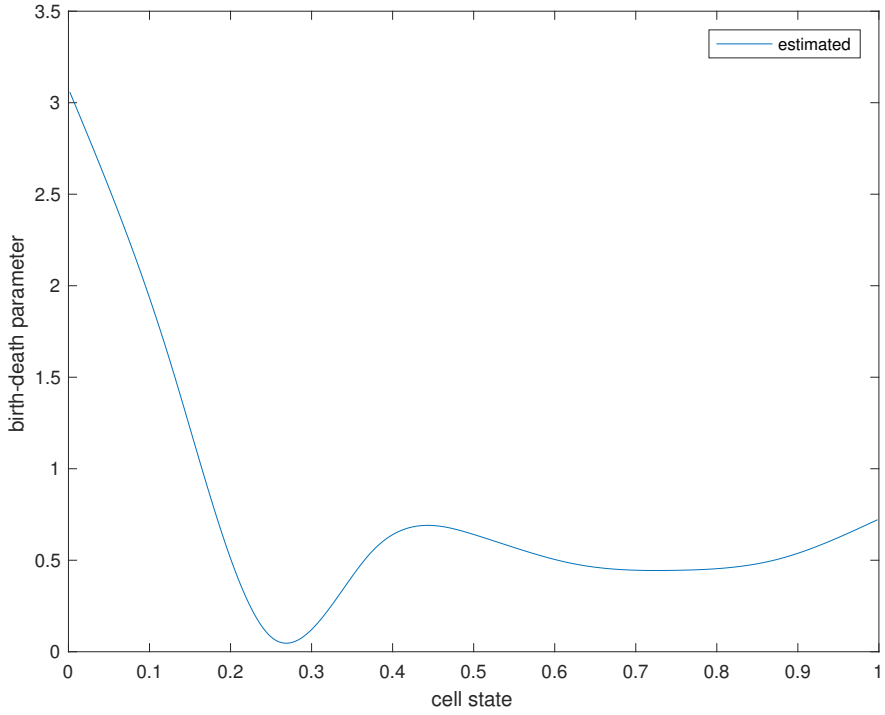

Figure 5: Estimated grwoth rate for the example data.