ByoDyn Tutorial and Quick Start Guide for ByoDyn version 4.8

Adrián López García de Lomana, Alex Gómez-Garrido, Miguel Hernández, Pau Rué Queralt and Jordi Villà-Freixa

— December 17, 2008 —

This document gives short description about the functionalities of ByoDyn as a quick start guide. On the tutorial we will show a set of examples on the use of ByoDyn for the study of a dynamic system. From a computational point of view we describe and analyze here a bunch of relevant biochemical systems. We give examples about metabolic networks, engineered gene networks and multicellular pattern formation. All the examples will be illustrated with the help of the computational tool ByoDyn.

Computational Biochemistry and Biophysics Laboratory
Research Unit on Biomedical Informatics
Universitat Pompeu Fabra - Institut Municipal d'Investigació Mèdica
c/ Dr. Aiguader 88, 08003, Barcelona, Spain
http://cbbl.imim.es

Contents

1 Quick Start Guide **3**

1.1 Starting from command line version . . . 3

1.2 Starting from `ByoDyn.web` server . . . 3

1.2.1 Access . . . 3

1.2.2 `ByoDyn.web` Home . . . 3

1.2.3 First Analyses: the Quick Start Examples . . . 4

1.3 Common functionalities . . . 6

1.3.1 Exporting . . . 6

1.3.2 Simulation . . . 6

1.3.3 Stochastic Simulation . . . 7

1.3.4 Optimisation . . . 8

1.3.4.1 Two Phases Hybrid Optimisation . . . 8

1.3.5 Fitness Function Calculation . . . 10

1.3.6 Trajectories Reconstruction . . . 10

1.3.7 Fitness Function Surfaces . . . 10

1.3.8 Sensitivity Analysis . . . 11

1.3.9 Identifiability Analysis . . . 13

1.3.10 Optimal Experimental Design . . . 13

2 Tutorial **14**

2.1 *Amadori* Model . . . 15

2.1.1 Biological Background . . . 15

2.1.2 Simulation . . . 17

2.1.3 Sensitivity Analysis . . . 17

2.1.4 Optimisation . . . 18

2.2 *Repressilator* Model . . . 19

2.2.1 Biological Background . . . 19

2.2.2 Simulation . . . 20

2.2.3 Sensitivity Analysis . . . 21

2.2.4 Optimisation . . . 21

2.2.5 Fitness Function Surfaces . . . 21

2.3 *Notch-Delta* Model . . . 24

2.3.1 Biological Background . . . 24

2.3.2 Simulation . . . 24

1 Quick Start Guide

1.1 Starting from command line version

In order to run `ByoDyn` from command line version, two different files are required: the *runner* file and the *model* file.

- *runner* file: This file contains all the running options of `ByoDyn` as for example the path to the *model* file, the running type when you call `ByoDyn`, a simulation, an optimisation or a study of the parameter sensitivities, for example.
- *model* file: This file contains the topology of the network we want to study. Two formats are valid, a *homemade* tags format and the standard format of systems biology markup language (SBML).

As the first time you use `ByoDyn`, the first step is to type the command

```
byodyn --examples
```

on the terminal in order to create a folder called `examples` with (1) a systems biology model of metabolism and (2) different *runner* files necessary to launch those studies. Go to the new created folder and you should be able to list a *runner* file for exporting called `exporting.rn`, for simulation called `simulation.rn`, for optimisation called `optimisation.rn`, for fitness function calculation called `fitnessFunctionCalculation.rn`, for model trajectories reconstruction from a given set of parameter values called `trajectoriesReconstruction.rn`, `surface.rn`, to build fitness function surfaces, another one called `sensitivity.rn` for launching a sensitivity analysis, `identifitiability.rn` for identifiability analysis and finally `oed.rn` for optimal experimental design.

1.2 Starting from `ByoDyn.web` server

1.2.1 Access

You can access to the `ByoDyn.web` server at http://cbbl.imim.es/byodyn-web. The initial front page manages the access to the `ByoDyn.web`. In the case that it will be the first time that you are accessing to `ByoDyn.web`, you can go clicking in the button "Help" for instructions for guest access. Otherwise, if you already have an user and password fill each one of the fields and click in the "Log in" button. Figure 1 shows the page you should see.

1.2.2 `ByoDyn.web` Home

The first page that you will see after the front page is called *Home*. We also display it, in Figure 2. The page shows the user workspace, called "My workspace", which is a virtual place in the application where your work files (i.e. models, analysis files, results) are stored . The user workspace is divided into three main subsections:

- **My models:** Here, you can find your models. The models are the computational representations of a biochemical system.

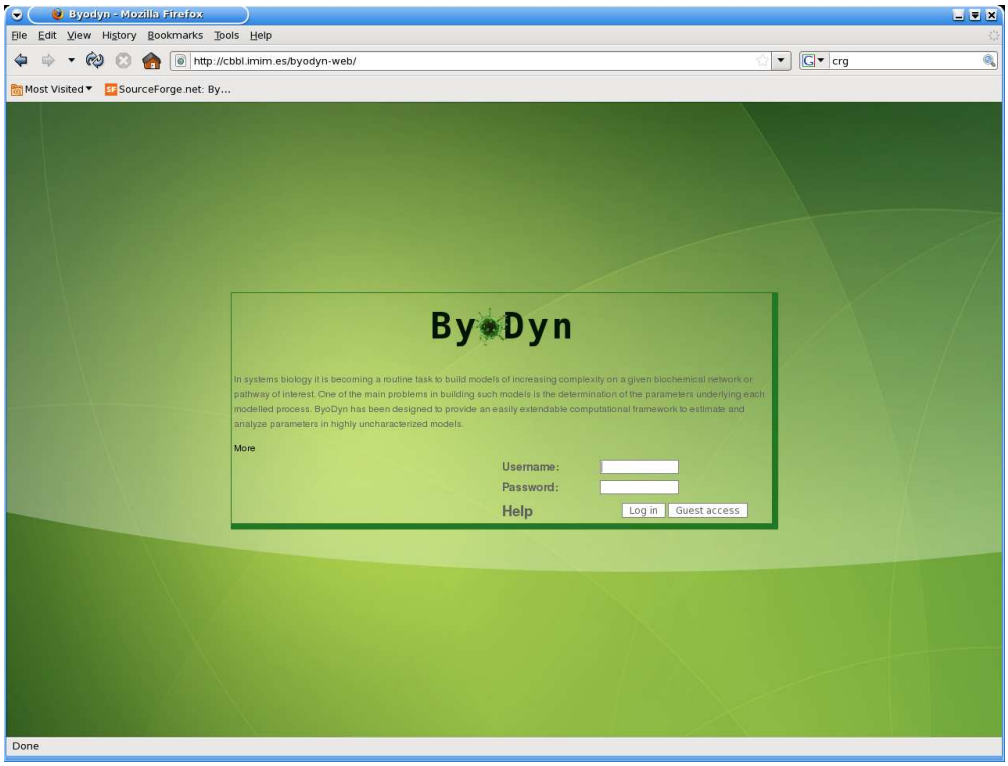

Figure 1: Screenshot of the `ByoDyn.web` front page.

- **My analysis:** This subsection stores your analysis files. The analysis files are the input files for `ByoDyn`, which define all the details and instructions in order to run the analysis.
- **My results:** The results of each analysis resulting from the execution of the analysis files will appear and will remain stored in this subsection.

1.2.3 First Analyses: the Quick Start Examples

In Section 1.3 we explain the different funcionalities `ByoDyn` offers via a bunch of example files. That section gives detailed explanations about the files and the results of those examples files of your workspace.

In order to find the example files you should browse along the menu bar at the left side of the screen. There you will find the **Examples** folder with two subfolders inside:

- **Option files from documentation:** In this subfolder you can find some analysis files, explained farther in this document, specifically in Section 1.3.
- **Models:** Here, you will find biochemical models in SBML format from public repositories.

Specifically, you need to do the following steps in order to work with the examples shown in Section 1.3:

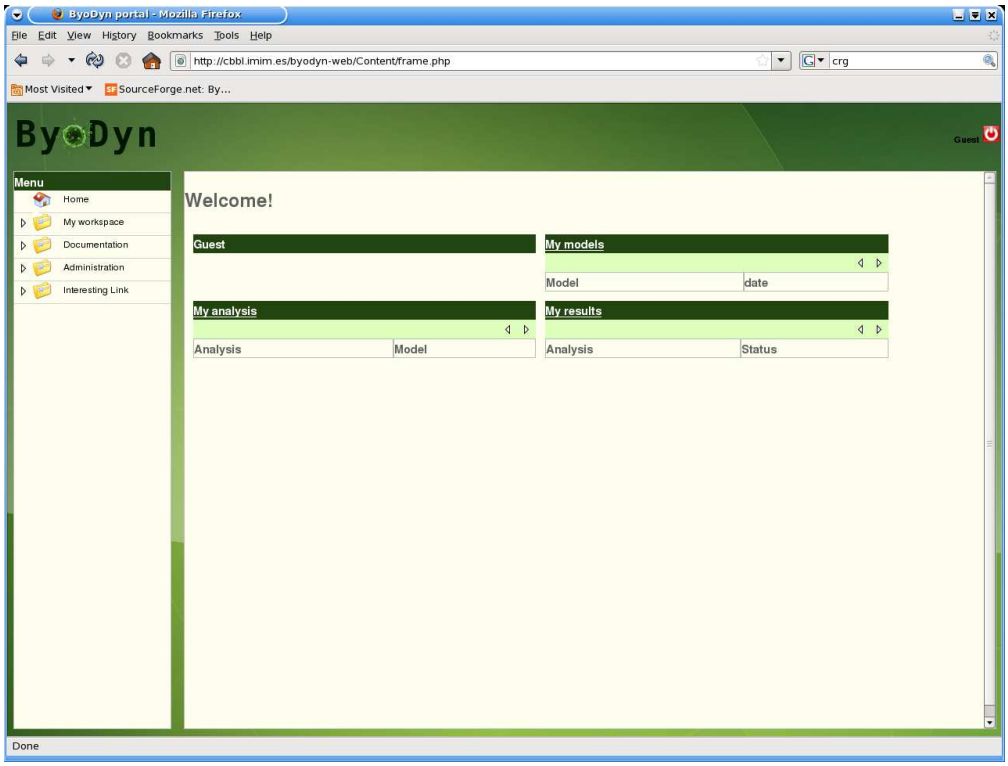

Figure 2: Screenshot of the `ByoDyn.web` home page.

- Click on Examples/Option files from documentation/Quick Start of menu bar.
- Now, in the main screen there are a list of files with a green arrow at the left. You need to click on the green arrow in order to copy the specific file on your workspace.
- Click on `Home` in order to return to the home page. Now, you can see in `My analysis` and in `My models`, the example files.
- Click on the specific analysis file and the details of this file will appear. Here, you could see the analysis file using the `Visual edition` tab or the `Text editor` tab.
- To execute the analysis please click on the fourth icon from the left, at the top, the one with the *Play* symbol.
- To see the state of the calculation and the results, please click on the last icon at top, the one with the folder symbol.

This protocol is valid for any of the functionalities explained at the next section.

1.3 Common functionalities

1.3.1 Exporting

Run the command `byodyn` and the name of the *runner* file, `exporting.rn` in this case. It will create an `output` directory with the results. Change to the `output` directory and you will list a file named `Kinetic_modelling_of_Amadori_degradation.xml` containing the new parameter values as defined on the runner file: parameter `k4v4` takes now 0.07 as value in the new SBML file.

1.3.2 Simulation

Run again the command `byodyn` and the name of the *runner* file, `simulation.rn`. Change to the `output` directory and you will list:

- `Kinetic_modelling_of_Amadori_degradation.description.txt`: A short description of the input model. We state the name of the model, the number of nodes, the number of constant species and the number of constant and non-constant parameters.
- `Kinetic_modelling_of_Amadori_degradation.out`: it is called that way because the given name of the model. More important is the extension of the files. This `.out` is a file with the concentration of each node (in columns) along time (first column).
- `Kinetic_modelling_of_Amadori_degradation.ps`: is equivalent from the former file but represented as a graph.
- `Kinetic_modelling_of_Amadori_degradation.veloc.out`: the same as `Kinetic_modelling_of_Amadori` but in this case the columns represent the rate of concentration change for the corresponding node.
- `Kinetic_modelling_of_Amadori_degradation.veloc.ps`: the corresponding graph for the file `Kinetic_modelling_of_Amadori_degradation.veloc.out`.
- `Kinetic_modelling_of_Amadori_degradation.tex`: it is a file with the system of ODEs and the equations of kinetic laws in latex format. Just use the program `pdflatex` to obtain the formulae in pdf format.
- `scratch` directory: this is a directory that contains necessary files for `ByoDyn` but of secondary interest for the user. We can list the following files:
  - `Kinetic_modelling_of_Amadori_degradation.0.integ.py`: this file contains the commands necessary to integrate the system of ODEs using SciPy. In the case of using Octave as the program to numerically solve the ODEs, you will find an equivalent `Kinetic_modelling_of_Amadori_degradation.0.oc`. The number refers to the rank of the processor in parallel machines.
  - `Kinetic_modelling_of_Amadori_degradation.gnu`: the `gnuplot` commands necessary to build the `.ps` files from the `.out`.
  - `Kinetic_modelling_of_Amadori_degradation.veloc.gnu`: the same but for the velocities instead of the concentrations.

If you have a look to the to the *runner* file (Fig. 3), here is an explanation about the different option fields:

```
# $Id: simulation.rn,v 3.9 2008/04/24 17:14:01 alglomana Exp $
##############################################
# created by Adrian L. Garcia de Lomana 300507
# this is an example of runner file to run a simulation
##############################################
modelFile BIOMD0000000050.xml
modelFormat     SBML
integrationMethod       python  default
runningType     simulation      velocity
time&timestep   170             .17
```

Figure 3: Snapshot of a *runner* file example for a deterministic simulation.

- `modelFile`: you need to specify the name of the *model* file. If the file is not at the same directory that the one from where you are running `ByoDyn`, set the complete path, for example: `/home/myself/myfolder/examplesOfByoDyn/myinput.xml`
- `modelFormat`: this defines the format of the *model* file. There are two possibilities, either `SBML` or `tags`.
- `integrationMethod`: this variable sets the program you want to use to solve the system of ordinary differential equations (ODEs) created by `ByoDyn`. There are two main possibilities that are specified at the first field: either `python` to use SciPy or `octave` to use Octave. The second field refers to the integration method being as possibilities `default`, `adams`, `non-stiff`, `bdf` or `stiff`. See the Reference Manual for further theoretical information about what these methods do. However, if you do not have any preference, you can adjust the first field with the option `automatic` instead of `python` or `octave`.
- `runningType`: the different possibilities are `runningType` are `exporting`, `simulation`, `parameterEstimat` `calculateFunction`, `dynamicsReconstruction`, `scoreSurface`, `sensitivityAnalysis`, `identifiabili` or `optimalExperimentalDesign`. In the case of a simulation, another field has to be append: either `velocity` or `noVelocity` to choose if you also require information of the velocity of the concentration change of the nodes along time.
- `time&timestep`: clearly you have to set the simulation time and the time step used by the simulation.

1.3.3 Stochastic Simulation

If you want to run a stochastic simulation execute `ByoDyn` with the option file `stochastic.rn`. The option file is shown in Figure 4 and we highlight here the differences from a deterministic simulation:

- `stochasticMethod`: this variable replaces `integrationMethod` and should be specified in order to run a stochastic simulation. Two algorithms are available, Gillespie and τ-leap. The respective arguments are `ssa gillespie` and `tau-leap default`. A further more argument is required in both cases, an integer, defining the number of runs that should be launched.

```
# $Id:$
##############################################
# created by Adrian L. García de Lomana
# this is an example of runner file to run stochastic simulations
##############################################
modelFile       isomerisation.xml
modelFormat     SBML
runningType     simulation      noVelocity
stochasticMethod ssa gillespie 10
time&timestep   200       0
```

Figure 4: Snapshot of a *runner* file example for a stochastic simulation.

- `time&timestep`: for the case of the τ-leap algorithm the meaning of `time&timestep` is the same as for the deterministic simulations, but please consult the user reference manual for the interpretation of `time&timestep` in the case of the Gillespie algorithm.

The output of the stochastic simulations is very similar to the one of the deterministic simulations, however, there are some minor differences which are described again at the user reference manual.

1.3.4 Optimisation

One of the most appealing functionalities of `ByoDyn` is the model parameter identification from time course experimental data using optimisation techniques. The User Reference Manual is pointed for more details about the algorithms used, the minimizing function and other issues of theoretical interest. In this case we present an optimisation of the 5 most sensitive parameters. One order of magnitude up and down for each parameter is specified as possible range for parameter exploration. Hybrid optimisations are composed of a global optimisation and a local optimisation algorithm. The local optimisation routines are written in Fortran and have to be compiled first.

1.3.4.1 Two Phases Hybrid Optimisation In the case of a two phases hybrid algorithm, the global optimisation runs until a certain condition is accomplished. At this point, it is assumed that the parameter vector is at the region of the global minimum and a local search is initialized for a quick convergence. In order to launch the optimisation, we have to use the file `optimisation.rn`. The optimisation lasts for several minutes depending on the machine and the optimisation evolution. It is a stochastic algorithm and the computational cost may vary largely. In our tests on a 1.83 GHz Intel Core Duo the optimisation took around two minutes of CPU time.

We understand a successful calibration of the parameters when the values retrieved from the optimisation match with the ones used to generate the trajectories. In a realistic situation, where the nominal value of the parameters is not known *a priori*, the final value of the fitness function can be useful to rank the obtained solutions and discriminate the most plausible values as there are several orders of magnitude of difference between a local minimum and a global minimum. In our case local minima were in the range of 10^4 while the global minimum is at a value of 10^{-16}. Obviously the fitness function value for an experimental calibration is not known, but huge differences should be observed when ranking the solutions.

Let's have a look to the resulting new files at the `output` directory:

- `Kinetic_modelling_of_Amadori_degradation.st`: this file contains for each generation of the GA algorithm the mean value of the population and the value of the best individual.

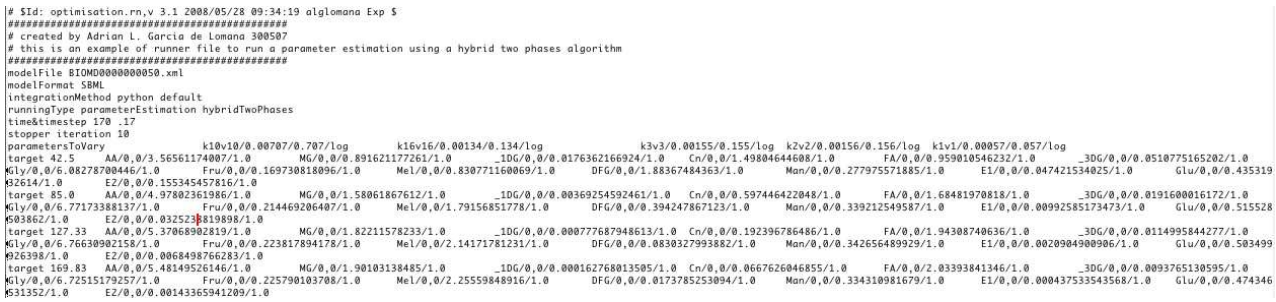

Figure 5: Snapshot of the *runner* file example to perform a parameter estimation.

- `Kinetic_modelling_of_Amadori_degradation` directory: when a solution has been found, a directory named as the input model is created. Two directories containing equivalent files will be found: `parametersFromGA` and `parameterFromHybridTwoPhases` directories. At these directories you an find several files named as the parameter you set to identify. The structure is the following: the first column is the value of the parameter, the other two the range in which the parameter space has been explored and the last one the value of the fitness function. Keep in mind that the starting points of the local search are the final values of the GA. It is worth to note that the fitness function of the results from the local search are orders of magnitude better than the ones of the GA. Also the resulting values of the parameters are very close to the nominal values (the values for which the fitting data was generated), in this case: $k10 = 0.0707$, $k16 = 0.0134$, $k3 = 0.0155$, $k2 = 0.0156$ and $k1 = 0.0057$.

- `scratch` directory: apart from the known `Kinetic_modelling_of_Amadori_degradation.0.integ.py` there is a directory called `localSearch`x being x a number of 12 digits. This directory contains several files for the local search. More information is given at the Reference Manual.

The runner file used is shown in Fig. 5. Compared to the *runner* file of the simulation (Figure 3), the new fields are the following:

- `stopper`: this variable controls when to stop the optimisation. In this case we set a number of generations for the GA. But it is possible also to set a value of the fitness function as threshold to have more control over the parameter space we are searching. More information is given at the Reference Manual.

- `parametersToVary`: Very similar to the field of `sensitivityParameters` from the sensitivity analysis, with two exceptions.

  - *A boundary has to be set*: the algorithm will explore the parameter space at that range of values. The first value has to be smaller that the upper value.
  - *Search scale*: When the searching range is wide (orders of magnitude) it has to be explored in a logarithmic scale so the distribution is not biased to the large values. If the parameter to be identify is an exponent, where the typical range of values is from 1.0 to 10.0, the linear scale is preferred. This is expressed by the tag `log` or `lin` respectively.

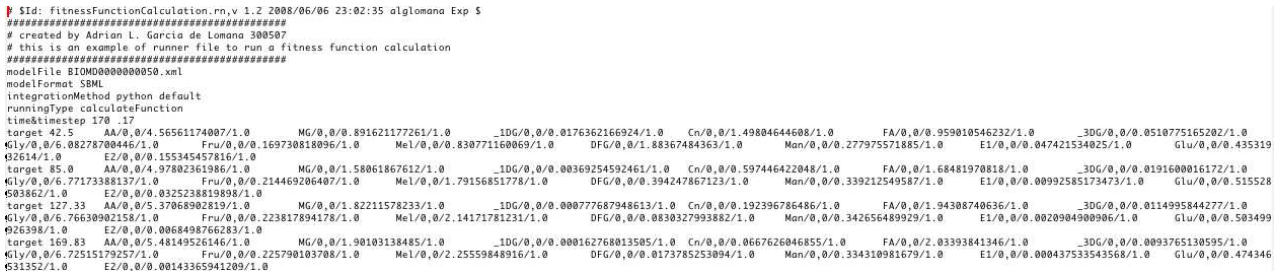

Figure 6: Snapshot of the *runner* file example to perform a fitness function calculation.

All of it separated by a / as shown in the Figure 5.

- `target`: this field refers to the (experimental) temporal expression data that we want to fit. The structure is the following: for a given time, you define the expression value by selecting a node, a cell, the concentration value and the standard deviation of the measurement. In our case the *experimental* data is produced *in silico* then each value comes from a single *measurement* and the value of the standard deviation is set as 1.0. The system is composed of a single cell so the cell index in the tissue matrix is the $0, 0$. As you can observe, several nodes can be fitted at a given time, but this is not a requirement.

1.3.5 Fitness Function Calculation

If the field `target` is defined as in Section 1.3.4, you can quickly know the distance from you model calling this functionality. The runner file you need is shown in Fig. 6.

1.3.6 Trajectories Reconstruction

After launching a parameter estimation, you end up with a bunch of solutions. In order to get an idea of how behave the new models you can define a directory where the files of the parameter solutions are stored and `ByoDyn` will reconstruct the trajectories corresponding to the parameter values. See Fig. 7 for a snapshot of the runner. The constrains defined on the `target` are also plotted. Furthermore, an interesting detail is that the trajectories reconstructed using the first value of the files is highlighted on the graph depicted as a dark thik line. We suggest to add as first line the original value of the model parameters to have an idea of the change of the model behaviour after the model calibration.

1.3.7 Fitness Function Surfaces

If we are interested to know the fitness function values in the surroundings of a given parameter point, the result of an optimisation for example, we can build a surface in the parameter space. In the `output` directory we have two new files not described previously:

- `k1.k2.txt`: this text file contains a matrix for the values of the fitness function surface.
- `k1.k2.ps`: this file is a postscript plot with the surface. The color indicates the fitness function value. The range of the parameters is shown.

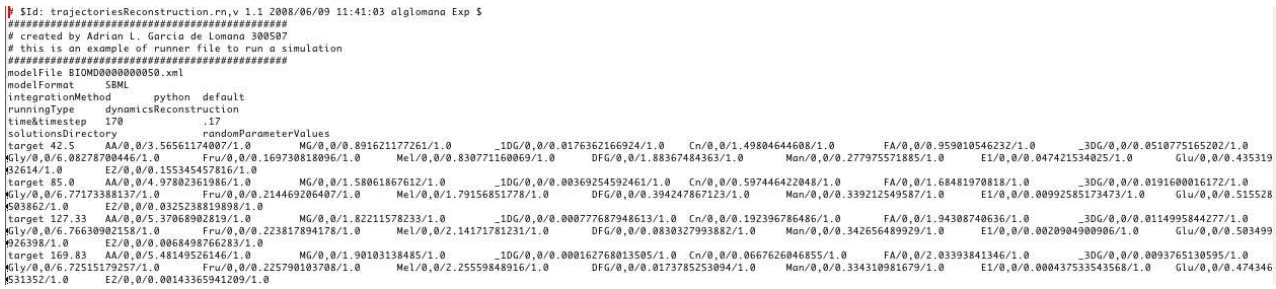

Figure 7: Snapshot of the *runner* file example to perform the trajectories reconstruction.

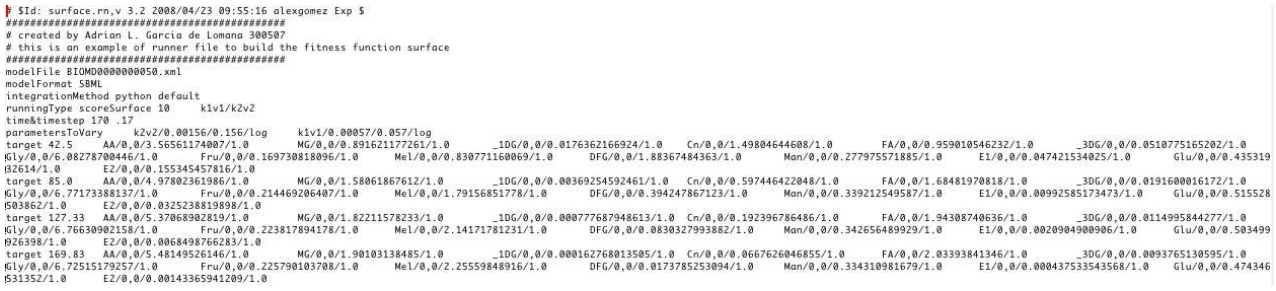

Figure 8: Snapshot of the *runner* file example to perform a fitness function surface.

The runner file looks as shown in Figure 8.

1.3.8 Sensitivity Analysis

In order to perform a sensitivity analysis you have to run `ByoDyn` using the *runner* file called `sensitivity.rn`. Again the results are in the folder called `output` and you could examine the following files:

- `Kinetic_modelling_of_Amadori_degradation.ps` and `Kinetic_modelling_of_Amadori_degradation.out` with the same meaning as the one explained in Section 1.3.2.
- `Kinetic_modelling_of_Amadori_degradation.sens.global.timeCourse.ps`: this graph represents the sensitivity of the system along time with respect to each of the model parameters. Check the Reference Manual for further information and the formula used.
- `Kinetic_modelling_of_Amadori_degradation.sens.global.txt`: this is a text file containing the global sensitivity of the parameters in order.
- `Kinetic_modelling_of_Amadori_degradation.sens.relative.ps`: this postscript file shows a matrix of the node relative sensitivity of each parameter. The intensity of the color is normalised against the maximum value for the model.

```
# $Id: sensitivity.rn,v 3.10 2008/04/01 16:22:45 alglomana Exp $
##############################################
# created by Adrian L. Garcia de Lomana 300507
# this is an example of runner file to perform a sensitivity analysis
##############################################
modelFile BIOMD0000000050.xml
modelFormat     SBML
integrationMethod       python  default
runningType     sensitivityAnalysis
time&timestep  170 .17
sensitivityParameters  k8v8     k4v4    k1v1    k6v6    k14v14  k5v5    k10v10  k2v2    k13v13  k16v16  k11v11  k3v3
```

Figure 9: Snapshot of a *runner* file example for running a sensitivity analysis. Note that for format reasons, not all parameters are shown but the sensitivity analysis includes all the parameters of the model.

- `Kinetic_modelling_of_Amadori_degradation.sens.relative.txt`: this text file holds the same information that the one shown in `Kinetic_modelling_of_Amadori_degradation.relative.sens.ps`.
- `scratch` directory: as at the simulation, the files called `Kinetic_modelling_of_Amadori_degradation.gnu` and `Kinetic_modelling_of_Amadori_degradation.0.integ.py` exist with the same meaning as before. Other files specific from the sensitivity analysis are:
  - `Kinetic_modelling_of_Amadori_degradation.`*`parameter`*`.out` where *parameter* is the name of the corresponding parameter of the system. These files store the values of the simulations in which the value of the corresponding parameter has been incremented infinitesimally.
  - `Kinetic_modelling_of_Amadori_degradation.sens.global.out`: this is the data for the corresponding graph `Kinetic_modelling_of_Amadori_degradation.global.sens.timeCourse.ps` at the `output` directory.
  - `Kinetic_modelling_of_Amadori_degradation.sens.global.gnu`: the commands for the plot `Kinetic_modelling_of_Amadori_degradation.global.sens.timeCourse.ps` using the data from `Kinetic_modelling_of_Amadori_degradation.sens.global.out`.

The *runner* file we have used (Fig. 9) is slightly different from the one of the simulation:

- The field of `runningType`, where you specify that you want to perform a sensitivity analysis by setting the variable to `sensitivityAnalysis`.
- `sensitivityParameters`: You have to define for which parameters you want to study the sensitivity of the system.

Finally, other additional files are created in the `scratch` directory.

1.3.9 Identifiability Analysis

It is also possible to run an identifiability analysis with `ByoDyn`. Apart from the `Kinetic_modelling_of_Amadori` and
`Kinetic_modelling_of_Amadori_degradation.ps` files described in Section 1.3.2, we have the following files specific of the identifiability analysis:

- `Kinetic_modelling_of_Amadori_degradation.criteria.txt`: A text file with the final value of the identifiability for the different criteria.
- `Kinetic_modelling_of_Amadori_degradation.FIM.txt`: A text file containing the Fisher information matrix. Please refer to the User Reference Manual for further information.
- `Kinetic_modelling_of_Amadori_degradation.COV.txt`: A text file with the inverse of the Fisher information matrix.
- `Kinetic_modelling_of_Amadori_degradation.correlation.txt`: The correlation matrix of the parameters in text format.
- `Kinetic_modelling_of_Amadori_degradation.correlation.ps`: This plot shows the correlation of the parameters. Blue colours show anticorrelation and red colour positive correlation. The intensity of the color is relative to the diagonal of the matrix.

The input file to run an identifiability analysis is shown in Figure 10. Note that in the `scratch` directory, several files named
`Kinetic_modelling_of_Amadori_degradation.xx.out`, where xx is the name of one of the parameters of the model, are created necessary to calculate the sensitivity of the model with respect to that parameter.

1.3.10 Optimal Experimental Design

Optimal experimental design (OED) tells you which would be the next temporal constrain so the identifiability problem becomes more tractable. Use a runner file similar to the one shown in Fig. 11. Apart from the `Kinetic_modelling_of_Amadori_degradation.out` and
`Kinetic_modelling_of_Amadori_degradation.ps` files described in Section 1.3.2, we have the following files specific of the OED analysis:

- `Kinetic_modelling_of_Amadori_degradation.oed.txt`: This is a text file where the most optimal values are stored. For each required criteria the best temporal point for each selected model species is saved.
- `Kinetic_modelling_of_Amadori_degradation.oed.`*criteria*`.ps`: For each OED criteria a postscript figure is created showing the value of the corresponding criteria for each selected node and time. Blue colours show anticorrelation and red colour positive correlation. The intensity of the color is relative to the diagonal of the matrix.

```
# $Id: identifiability.rn,v 3.4 2008/06/02 17:43:22 alglomana Exp $
##############################################
# created by Alex Gomez-Garrido 280907
# this is an example of runner file to perform an identifiability analysis
##############################################
modelFile BIOMD0000000050.xml
modelFormat     SBML
integrationMethod       python  default
runningType     identifiabilityAnalysis
time&timestep   170       .17
sensitivityParameters  k4v4    k1v1    k6v6    k14v14  k5v5    k10v10  k2v2    k13v13  k16v16  k11v11  k3v3    k9v9
target 42.5     AA/0,0/3.56561174007/1.0        MG/0,0/0.891621177261/1.0       _1DG/0,0/0.0176362166924/1.0    Cn/0
Gly/0,0/6.08278700446/1.0       Fru/0,0/0.169730818096/1.0      Mel/0,0/0.830771160069/1.0      DFG/0,0/1.8836748436
32614/1.0       E2/0,0/0.155345457816/1.0
target 85.0     AA/0,0/4.97802361986/1.0        MG/0,0/1.58061867612/1.0        _1DG/0,0/0.00369254592461/1.0   Cn/0
Gly/0,0/6.77173388137/1.0       Fru/0,0/0.214469206407/1.0      Mel/0,0/1.79156851778/1.0       DFG/0,0/0.3942478671
503862/1.0      E2/0,0/0.0325238819898/1.0
target 127.33   AA/0,0/5.37068902819/1.0        MG/0,0/1.82211578233/1.0        _1DG/0,0/0.000777687948613/1.0  Cn/0
Gly/0,0/6.76630902158/1.0       Fru/0,0/0.223817894178/1.0      Mel/0,0/2.14171781231/1.0       DFG/0,0/0.0830327993
926398/1.0      E2/0,0/0.00684987662831/1.0
target 169.83   AA/0,0/5.48149526146/1.0        MG/0,0/1.90103138485/1.0        _1DG/0,0/0.000162768013505/1.0  Cn/0
Gly/0,0/6.72515179257/1.0       Fru/0,0/0.225790103708/1.0      Mel/0,0/2.25559848916/1.0       DFG/0,0/0.0173785253
531352/1.0      E2/0,0/0.00143659412091/1.0
```

Figure 10: Snapshot of an example of *runner* file for running an identifiability analysis. Note that for format reasons, not all parameters are shown but the identifiability analysis includes all the parameters of the model.

2 Tutorial

Mainly all the work of `ByoDyn` will be done from a terminal. A terminal is a tool to interact with the computer using commands. Locate the terminal on your operating system and launch it. All the text in `text font` corresponds to the commands you type on the terminal. First obtain the exercises from the directory where you installed `ByoDyn` and copy them into your home (or any other if you prefer) to start working:

```
cp -r $BYODYN_PATH/examples/tutorialExercises .
```

If nothing happens, everything went well. Now you can go to the new created directory. To do that type the following.

```
cd tutorialExercises
```

and check what is in there:

```
ls
```

Three directories will appear containing computational models of metabolism and gene regulation we will study. The first two models are publicly available at the *BioModels Database* http://www.ebi.ac.uk/biomodels/ and are in SBML format. SBML (Systems Biology Markup Language) is *de facto* format for computational models. More than 90 software tools support it and an active community is behind it http://sbml.org/index.psp. These models, involving gene regulation and metabolism, based on uni-cellular systems. The last one is the result of a multidisciplinary collaboration and simulates the early patterning and neural specification during early inner ear development. In this often, we aim at studying gene regulation in a multicellular environment to understand pattern formation.

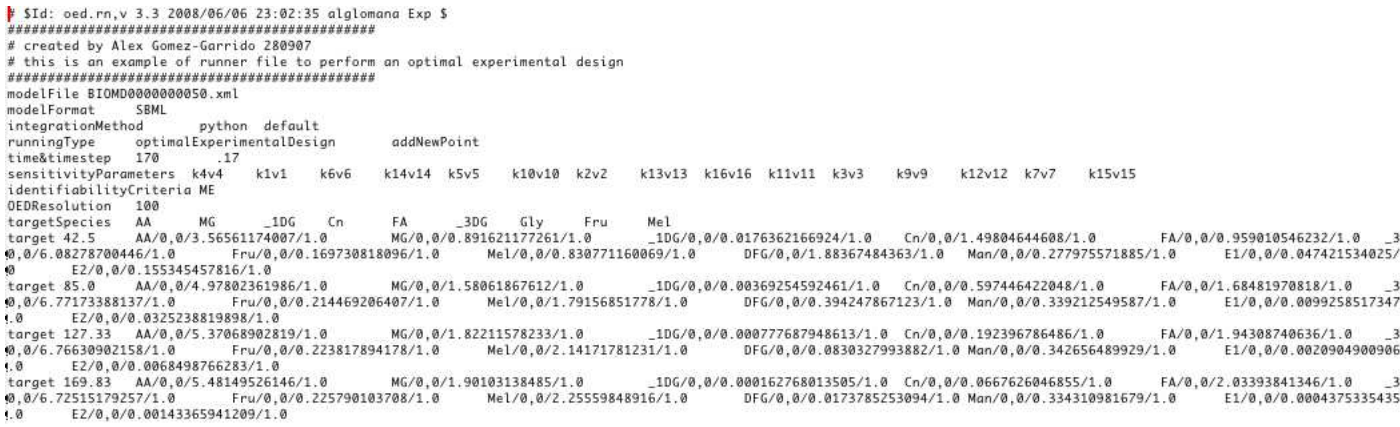

Figure 11: Snapshot of an example of *runner* file for running an optimal experimental design protocol.

2.1 *Amadori* Model

First change to the directory where the model and the options files for `ByoDyn` are:

```
cd amadori
ls
```

Among the different files, the file Amadori.xml contains the model description in SBML format. It is number 50 of the BioModels database ninth release. In addition, files with the .rn extension are the runner files for `ByoDyn` where the different tasks we want the program to do are specified.

2.1.1 Biological Background

This model accounts for sugar metabolism. Thermal decomposition of N-(1-deoxy–fructos-1-yl)-glycine (DFG) is a complicate kinetic pathway involving important metabolites as acetic acid, formic acid, glucose, fructose or mannose among others. From the metabolic network (please check Scheme 1 of Martins and Van Boekel [2003]) a kinetic model has been proposed in [Martins and Van Boekel, 2003]. The model leads to the following set of equations[1].

Kinetic laws:

$$v12 = k12 * Man \quad (1)$$

$$v13 = k13 * Glu \quad (2)$$

$$v10 = k10 * E1 \quad (3)$$

$$v11 = k11 * E1 \quad (4)$$

$$v16 = k16 * E2 \quad (5)$$

[1]the above set of equations were extracted by `ByoDyn` from the SBML file directly in the `.tex` format file `Kinetic_modelling_of_Amadori_degradation.tex` in the output directory

$$v14 = k14 * Cn * Gly \quad (6)$$

$$v15 = k15 * Cn \quad (7)$$

$$v1 = k1 * DFG \quad (8)$$

$$v2 = k2 * DFG \quad (9)$$

$$v3 = k3 * DFG \quad (10)$$

$$v4 = k4 * E1 \quad (11)$$

$$v5 = k5 * \_3DG \quad (12)$$

$$v6 = k6 * \_3DG \quad (13)$$

$$v7 = k7 * E2 \quad (14)$$

$$v8 = k8 * \_1DG \quad (15)$$

$$v9 = k9 * \_1DG \quad (16)$$

ODEs:

$$\frac{\mathrm{d}[DFG]_i}{\mathrm{d}t} = -v2 - v3 - v1 \quad (17)$$

$$\frac{\mathrm{d}[E1]_i}{\mathrm{d}t} = +v1 - v10 - v4 - v11 \quad (18)$$

$$\frac{\mathrm{d}[E2]_i}{\mathrm{d}t} = -v7 - v16 + v2 \quad (19)$$

$$\frac{\mathrm{d}[Cn]_i}{\mathrm{d}t} = +v3 - v14 - v15 + v5 + v8 \quad (20)$$

$$\frac{\mathrm{d}[Gly]_i}{\mathrm{d}t} = -v14 + v10 + v11 + v4 + v7 + v3 + v16 \quad (21)$$

$$\frac{\mathrm{d}[\_3DG]_i}{\mathrm{d}t} = +v13 - v5 - v6 + v4 \quad (22)$$

$$\frac{\mathrm{d}[FA]_i}{\mathrm{d}t} = +v6 + v15 \quad (23)$$

$$\frac{\mathrm{d}[\_1DG]_i}{\mathrm{d}t} = -v8 + v7 - v9 \quad (24)$$

$$\frac{\mathrm{d}[AA]_i}{\mathrm{d}t} = +v9 + v15 \quad (25)$$

$$\frac{\mathrm{d}[Man]_i}{\mathrm{d}t} = +v10 - v12 \quad (26)$$

$$\frac{\mathrm{d}[Glu]_i}{\mathrm{d}t} = +v11 + v12 - v13 \quad (27)$$

$$\frac{\mathrm{d}[MG]_i}{\mathrm{d}t} = +v15 \quad (28)$$

$$\frac{\mathrm{d}[Mel]_i}{\mathrm{d}t} = +v14 \quad (29)$$

$$\frac{\mathrm{d}[Fru]_i}{\mathrm{d}t} = +v16 \quad (30)$$

The study of the system computationally can give insights of the network parameters and reaction

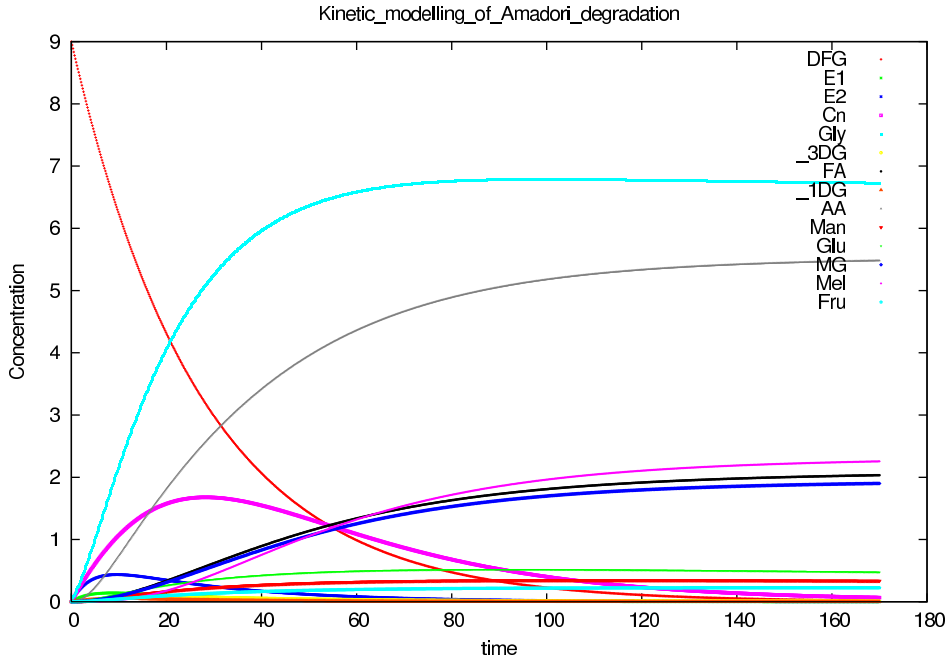

Figure 12: Numerical trajectories of the simulation the Amadori simulation.

mechanisms.

2.1.2 Simulation

We will call `ByoDyn` with the file `simulation.rn` as argument. In this file all the options for `ByoDyn` in order to run appropriately a simulation are stored. The default name of the option name for the output directory (`output`) can be changed to whatever other name by using the `-o` option as in:

```
byodyn -o simulation simulation.rn
```

The result of the simulation is the calculation of the concentration of the reacting nodes along time. You can see the graph by making use of your preferred postscript viewer. For example:

```
ggv simulation/Kinetic_modelling_of_Amadori_degradation.ps&
```

DFG is degraded giving raise of a variety of products, mainly glycine (Gly) and acetic acid (AA). Please have a look to Figure 12.

2.1.3 Sensitivity Analysis

Sensitivity analysis is a technique very informative to understand which is the key step on a reaction pathway. It is calculated as the change on the concentrations of a system while varying a parameter. Type the following for the study:

```
byodyn -o sensitivity sensitivity.rn
```

See the results at:

```
ggv sensitivity/Kinetic_modelling_of_Amadori_degradation.sens.global.timeCourse.ps
```

You will see the same results of Figure 13. It is not very clear which parameter is the most sensitive

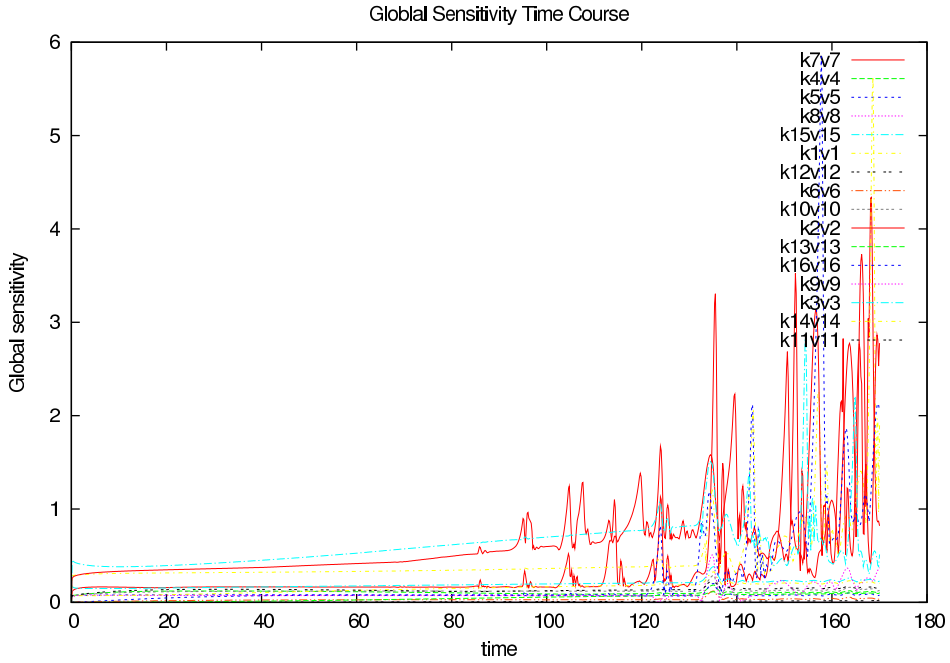

Figure 13: Time course global sensitivity of the nodes of the Amadori model.

in the model. We can have a better clue accessing the figure showing the matrix of sensitivities of nodes/parameters:

```
ggv sensitivity/Kinetic_modelling_of_Amadori_degradation.sens.relative.ps
```

As you can see in Figure 14, the parameters with highest sensitivity are k_2, k_1. These constants regulate the degradation of DFG as seen in Equations 8, 9 and 17. Indeed, the degradation of this compound is known to be the trigger of the complete metabolic network.

2.1.4 Optimisation

The set of the parameters (degradation constants, binding constants, formation constants, etc.) of this model are the following:

```
cat simulation/Kinetic_modelling_of_Amadori_degradation.description.txt
```

Now, we can use optimisation techniques to infer the nominal values using a temporal experimental data set. Type the following for the optimisation:

```
byodyn -o optimisation optimisation.rn&
```

Optimisation techniques are computationally expensive. Specially global methods that commonly are based on stochastic variables. In this case the optimisation will run in the order of minutes. In the meantime let's have a look to the experimental data we are using:

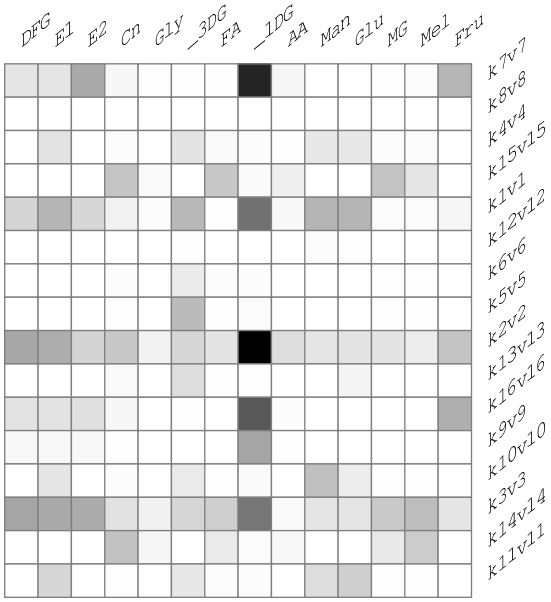

Figure 14: Relative sensitivity of the nodes with respect to the parameters of the Amadori model.

```
emacs optimisation.rn
```

Check the fitness function that represents a distance between the experimental data and the simulated results.

$$F(\mathbf{p}) = \frac{1}{T_k} \sum_{i=1}^{T_k} \frac{\left(y_i^{\text{obs}} - y_i^{\text{calc}}(\mathbf{p})\right)^2}{\sigma_i} \tag{31}$$

Because the number of parameters is quite large, the final fitness function value for most of you will be around 10^{-3}. Are the optimal parameters close to the nominal ones? Could you explain what happened? If you are lucky the final value of your fitness function will be around 10^{-15}. Which are the values of the parameters? What happened instead? Could you compare the situation with the optimisation in Section 1.3.4? If you would like to find the nominal values for all the parameters, change the optimisation method, instead of `hybridTwoPhases` set it as `hybridOnePhase` and the convergence will occur. In our tests, it took less than 8 minutes the `hybridTwoPhases` optimisation and several hours for the `hybridOnePhase` calibration on a 1.83 GHz Intel Core Duo.

2.2 *Repressilator* Model

2.2.1 Biological Background

The model is composed of three genes and their genetic products. Each of the gene products inhibits the transcription of the other one (Figure 15). This system was artificially engineered on bacteria organism as a proposed oscillator. More information can be found at the original article [Elowitz and Leibler, 2000].

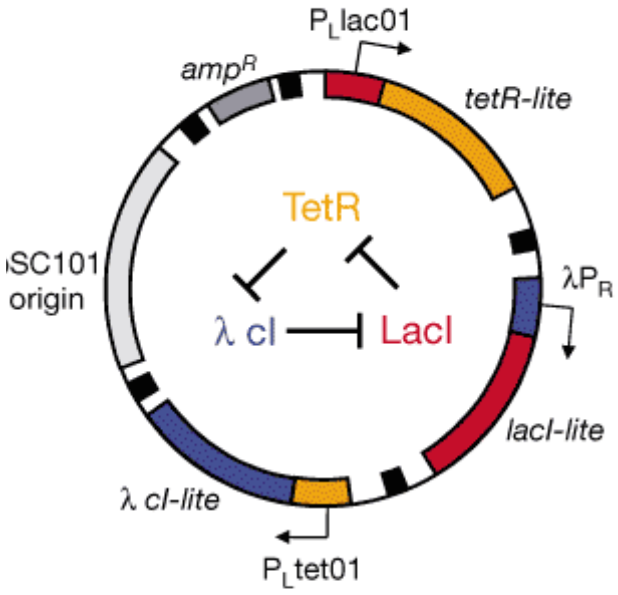

Figure 15: Figure obtained from Elowitz and Leibler [2000]. The proposed network was built on a plasmid.

2.2.2 Simulation

First of all change to the corresponding directory:

```
cd ../repressilator
```

And execute the corresponding command:

```
byodyn -o simulation simulation.rn
```

As you can see at the simulation results,

```
ggv simulation/repressilator.ps
```

the network behaves as an oscillator. These parameters account for the phenotype that has been checked experimentally. Now, we can change the value of the parameters and observe that the dynamics of the system differ. The parameters used for the former simulation were the original ones of the SBML as you can see in the file:

```
cat simulation/repressilator.description.txt
```

In order to change the value of the parameter of the model, you have to edit the file `simulation.rn` file adding the following line:

```
parameters name_of_parameter=value ...
```

We suggest to use `Emacs` as editor:

```
emacs simulation.rn
```

Let's vary the parameter n by adding the following line:

```
parameter alpha=5 beta=100
```

and run the simulation again. Can you explain the differences on the dynamical behavior from the last simulation? We suggest you to have a look to the system of ODEs (Ordinary Differential Equations):

```
cd simulation
pdflatex repressilator.tex
ggv simulation.pdf
```

and Figure 1b of the Elowitz and Leibler [2000] publication.

2.2.3 Sensitivity Analysis

Perform the sensitivity analysis:

```
byodyn -o sensitivity sensitivity.rn
```

As you can see the exponent n is the most sensitive parameter of the network. Also the dynamics are greatly affected by the half live of the mRNA.

2.2.4 Optimisation

We will perform the optimisation of the two most sensitive parameters on a range of two orders of magnitude:

```
byodyn -o optimisation optimisation.rn
```

The model calibration takes several minutes although the convergence is not achieved with fitness function values at the order of 10^5. To gain further insights about why the optimisation was not successful, we studied the fitness function in the surroundings of the nominal values of the parameters in the next section.

2.2.5 Fitness Function Surfaces

We can build the fitness function surface running the `ByoDyn` option file `surface60.rn`. The resolution is largely poor to determine the relative size of the well of the global minimum therefore we ran the same option file but for a resolution of 500 points for each variable. The execution lasted for several hours on a single PC, approximately 16 hours of CPU for each 500 points resolution figure. As it can be seen in Figure 16 the section of the well of the global minimum is so small that its finding is practically very difficult. An interesting study is the evolution of the fitness function in the surroundings of the global minimum well along the addition of new targeting data. The files `surface6.rn`, `surface60.rn`, `surface600.rn` and `surface6000.rn` were executed with a resolution of 500 points. The computational cost was in the order of days for each surface on a Intel Core 2 Duo. Figure 17 shows that adding experimental data points makes the surface smoother and the model calibration becomes much easier.

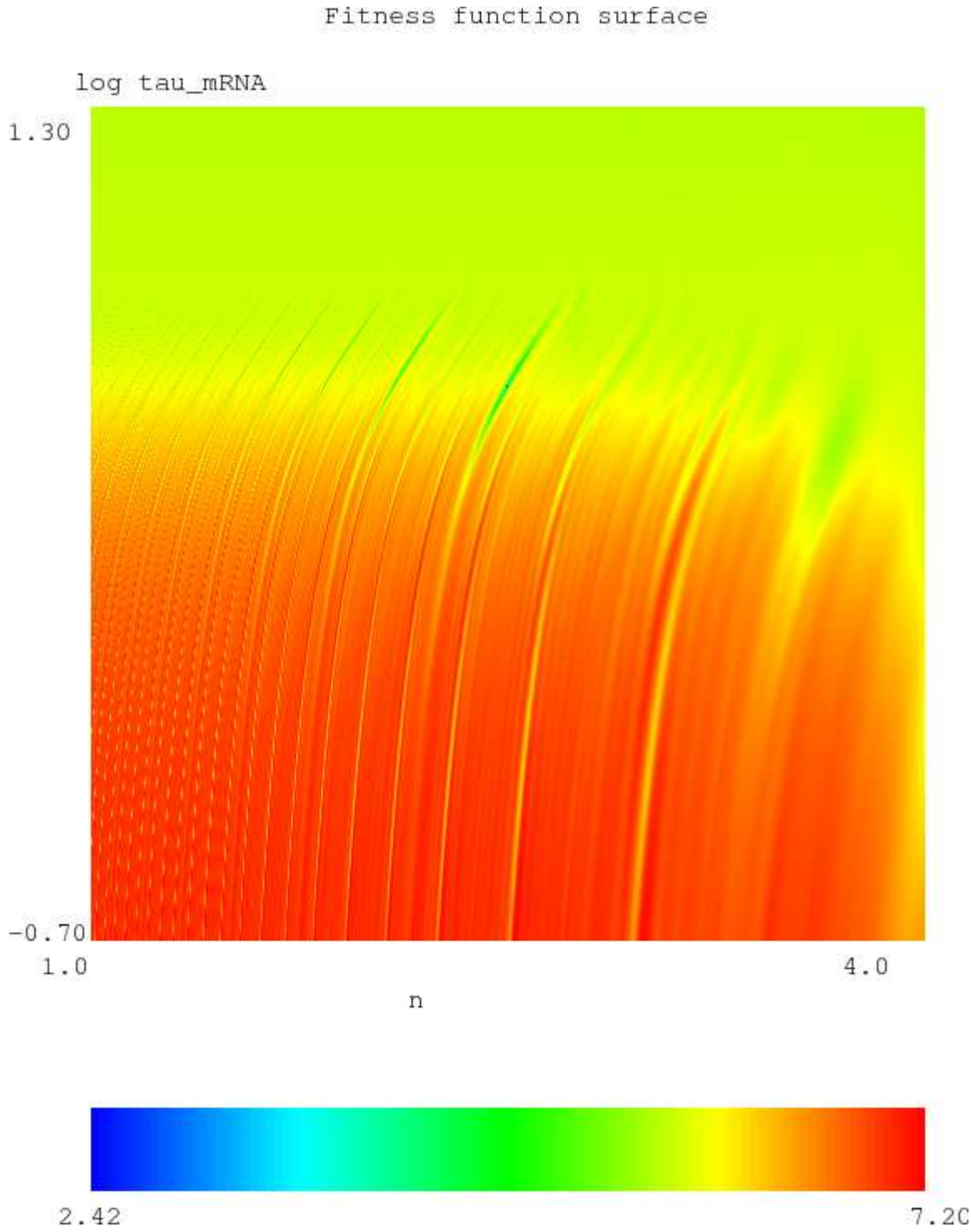

Figure 16: Fitness function surface of n and tau_mRNA for the optimisation problem of section 2.2.4.

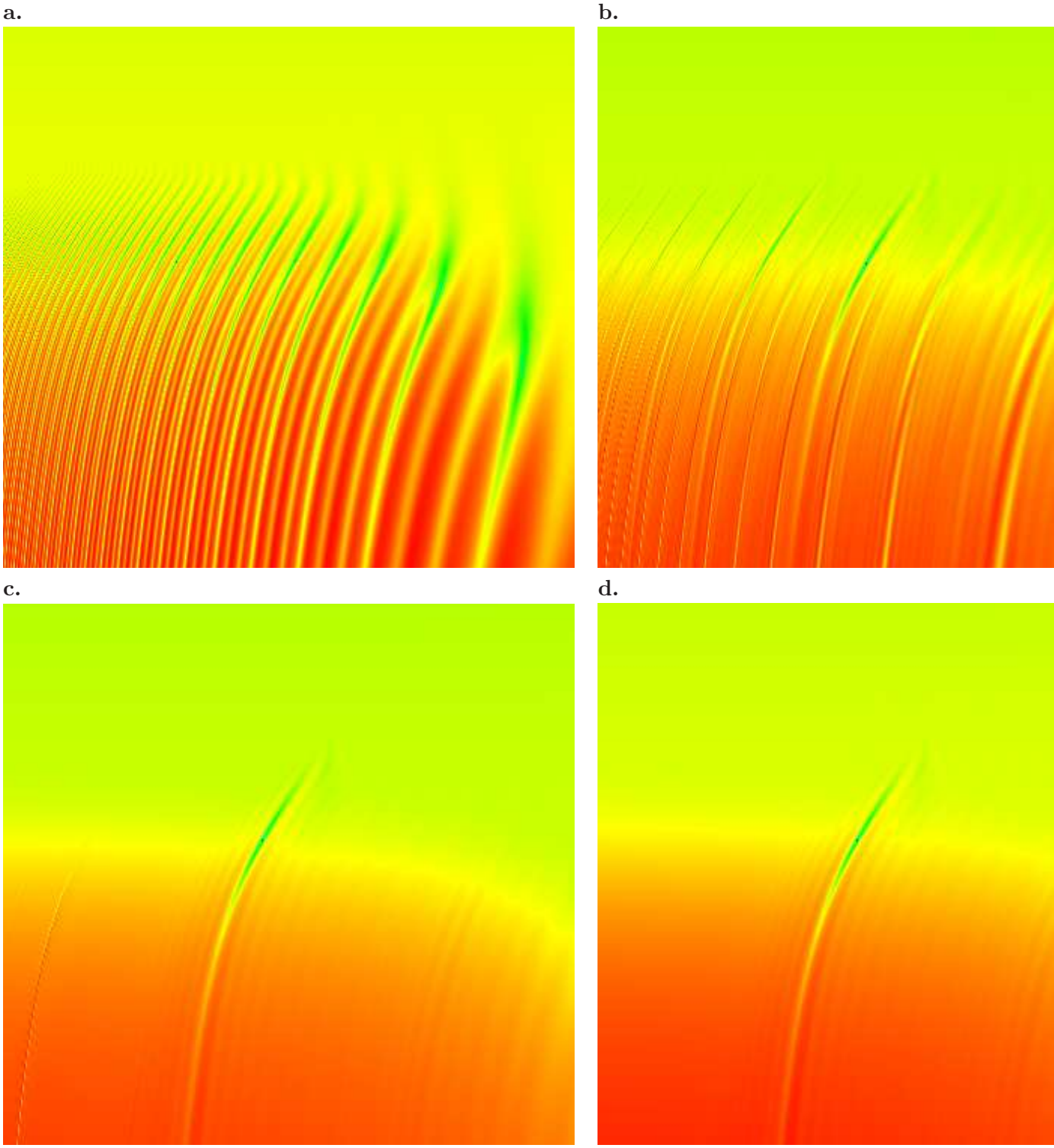

Figure 17: Fitness function surfaces for 6 (**a.**), 60 (**b.**), 600 (**c.**) and 6000 (**d.**) target points. Note that the section of the global minimum well maintains its section but the number of local minima reduces considerably.

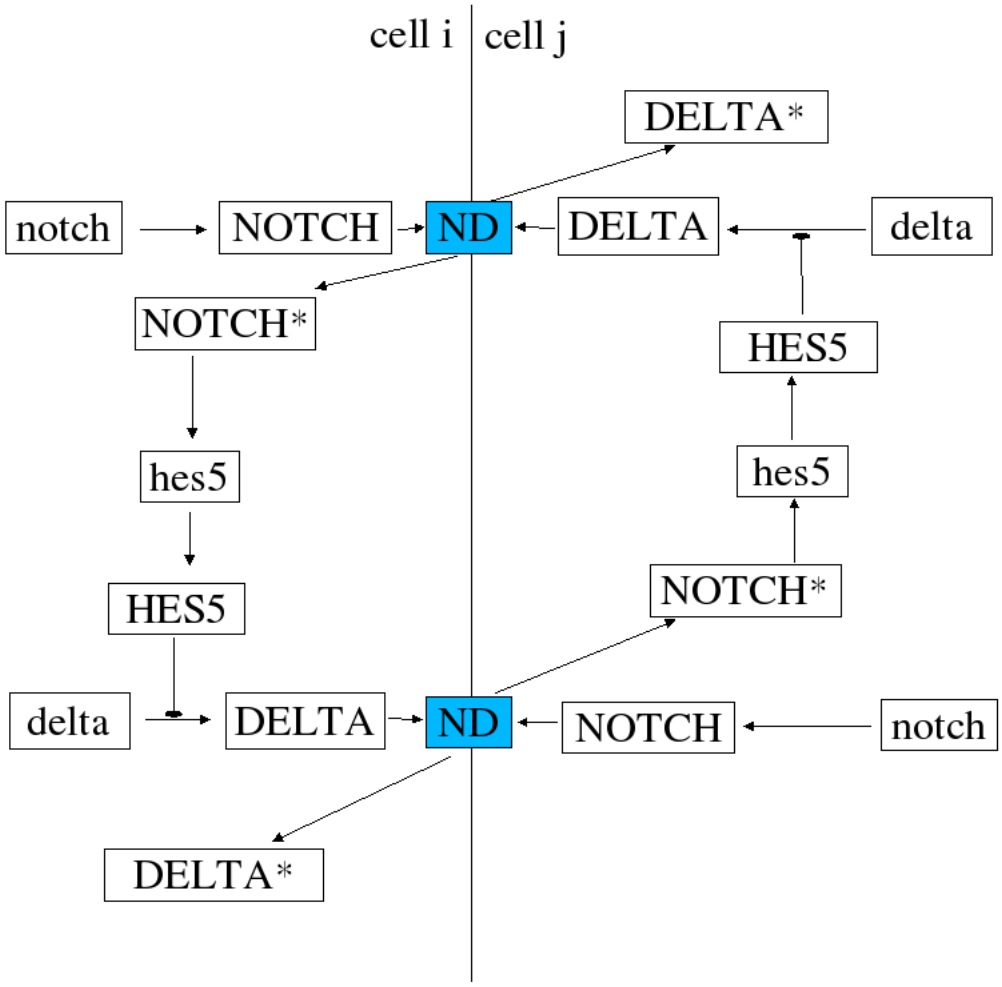

Figure 18: Lateral inhibition is based on the interactions of Notch and Delta.

2.3 *Notch-Delta* Model

2.3.1 Biological Background

The otic placode is a transitory structure that will give raise to the inner ear. During the its development cell fate is achieved by lateral inhibition and other biological pattern formation methods as boundary formation. Further information can be found at [Alsina et al., 2004]. On the computational work we have focus on the lateral inhibition mechanism (Figure 18).

2.3.2 Simulation

Run the simulation as usual:

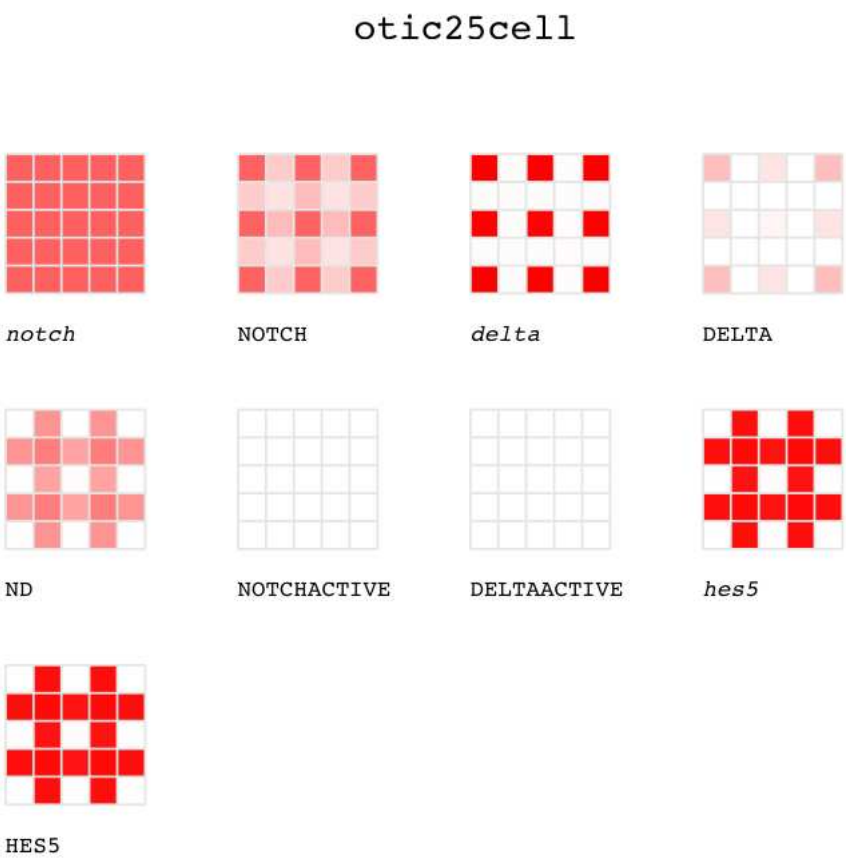

Figure 19: Multicellular plot of a process of lateral inhibition on an epithelium of 25 cells. The figure represents the steady state of the trajectories of the system at time $t = 3000$.

```
cd ../notch
byodyn simulation5.rn
```

Check the results using again the graphical program `ggv`,

```
ggv output/otic25cell.grid.ps
```

as we show in Figure 19. You can also run the models of 2, 9 and 49 cell matrices. Simply note that it takes around 7 and 25 minutes the simulation of models containing 25 and 49 cells respectively on an Intel Core Duo 1,83 GHz.

References

B. Alsina, G. Abellò, E. Ulloa, D. Henrique, C. Pujades, and F. Giráldez. FGF signaling is required for determination of otic neuroblasts in the chick embryo. *Dev. Biol.*, 267(1):119–134, 2004.

M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.

S. I. Martins and M. A. Van Boekel. Kinetic modelling of Amadori N-(1-deoxy-D-fructos-1-yl)-glycine degradation pathways. Part II–kinetic analysis. *Carbohydr. Res.*, 338:1665–1678, 2003.