bioSim Glocal Toolbox – version 1.1

Version 1.1, November 2009

Marc Hafner, EPFL-IC-LANOS, Switzerland

Help and comments to marc.hafner@epfl.ch

When using this toolbox, please cite the article
Hafner M, Koeppl H, Hasler M, Wagner A (2009) **'Glocal' Robustness Analysis and Model Discrimination for Circadian Oscillators.** PLoS Comput Biol 5(10): e1000534.

Main functions and quick start

The toolbox does not require installation; only the directory of the toolbox has to be added to MATLAB through the **`addpath`** function. A model is created by calling the function **`model`**. The equations of the system have to be written in the corresponding files. With this object, deterministic simulations can be performed using **`simulation_deterministic`** that provides a **`data`** object as an output. This result can be plotted with the function **`plot`** (overload of the standard MATLAB plot function). The function for the parameter search can be called with **`multiMC_along_PCA`**. And the final uniform sampling is performed with the function **`volumeMC_along_PCA.`**

Introduction

This MATLAB toolbox was developed for a fast and optimized computation of the robustness of biological circuits such as those responsible for circadian cycles. It samples the parameter space searching for parameter vectors for which a model fulfills some specific criteria. Its main ideas and algorithms can easily be expanded to other biochemical networks. The advantage of this toolbox over others is the ability to perform efficient batch analysis. All major parameters can be entered as inputs. The source code may require changes if one wants to optimize some specific tasks.

Type **`help bioSimToolbox`** to display this information in MATLAB and the links to the help file of other functions.

Features

Object oriented programming

Models are stored as MATLAB objects and contain all necessary information for the algorithms to run. They contain the original parameter vector, the initial conditions, the path of the working directories and additional internal values as, for example, the number of state variables.

Directories

The user has to set the working directory (default: **`C:\Users\bioSim\)`.** If this directory cannot be created or is not freely accessible, the user has to define another folder manually. Folders "**`backup`**", "**`temp`**" and "**`models`**" are created by the toolbox. They contain backups for long simulations, temporary files for stochastic simulations, and the specific integrators for each model, respectively.

The library folder contains all the algorithms, but no model. Subdirectories contain classes and some specific functions like frames for integration functions.

After being created, the models can be saved. This operation creates a new folder which contains a MATLAB m-file that allows reinstalling the model (in a new session or on other computers).

Languages

Algorithms for parameter sampling are written in MATLAB.

Auto backup

Some of the long algorithms have auto backups implemented. The user has to specify a number for the simulation in order for the program to save backup information in a file specific to the simulation. If crashed or stopped, the simulation can be restarted when the function is called with the same simulation number on the same computer.

Display mode

A global variable **`DisplayMode`** can be set from 0 to 6. These values control the written outputs as follows:

- 0, nothing is display from functions
- 1, major errors are displayed
- 2, all errors are displayed
- 3, some information and all errors are displayed
- 4, detailed information and all errors are displayed
- 5, everything is displayed
- 6, some functions stop to check for displayed information. Not valid for simulations.

Prompt mode

A global variable **`PromptMode`** can be set to 0 or 1. When set to 0, the functions and scripts use the default setting and do not ask for inputs.

Functions

Model creation

A new model can be declared with using the class "**`model`**" creator. Required parameters are: **`oModel(tModelName,vk,vX0, …)`** where **`tModelName`** is a string, **`vk`** is a parameter vector, and **`vX0`** is some initial conditions. One can specify:

- **sconstrained**, the number of mass constraints of the system (last components of the parameter vector).
- **sT0**, the expected period for an oscillatory system, or any time scale for other purposes.
- **tOutputExpression**, a string expression of the output of the system as a function of the state variable. It could be expressed as function of the state variable (with generic name **X1, X2** …). It should be evaluable with a vector following MATLAB formalism (for example **X1.^2** instead of **X1^2**).
- **unitconcentration**, factor for the concentration of the molecular species relative to unit molar concentration M. For example if concentrations are expressed as μM, this value should be 10^{-6}.
- **unitvolume**, factor for the concentration of the species relative to liter. For example if concentrations are expressed as μm^3, this value should be 10^{-15}.
- **vlegend**, cell containing a string for each state variable.

By generating a new model, MATLAB automatically creates a new folder for the integrators in **[bioSimworkingpath '\models\']**. MATLAB creates the frame for the integrators, but equations need to be entered manually. Frames can be created using **Create*(oModel).**

A model can be saved with the function **savemodel(oModel)**. This will save the model and its integrator in a specified folder. A **installmodel** file will be created, which can reinstall the model, and copy the integrators on the same or a different computer.

Simulations

Different m files can be called to use one of the simulators:

- **simulation_deterministic** , for using MATLAB ODE integrator.
- **simulation_parameter_variational** , for using parameter variational integrator.
- **simulation_variational** , for using variational integrator.

The simulators accept the following inputs: **simulation_*(oModel, tspan, …)**. Depending on the simulator, these inputs are different (see the help files).

Simulators return a **data** structure containing the time points (**data.T**), the state variables (**data.X**) and some additional information. X should be called as **X(*time point, state variable*)**. For stochastic simulations: **X(*simulation #, time point, state variable*)**.

Plotting

plot(data), will plot the output of a deterministic simulation or the mean of the output of multiple stochastic simulations. **plottrajectories(data)** plots the multiple trajectories of stochastic simulations. The plot functions can accept expressions for the x and y axes as a cell, such as: **plot(data,{'X3', 'X1+X2', 'legend'}).** Multiple cells can be entered. Options for the MATLAB plot function are also accepted, such as: **plot(data,{…},'.-','Markersize',12 …)**

plotks(ks) can be used to plot a set of parameters with projections on all axes. It accepts the same options as the MATLAB **plot** function.

Scripts

Parameter search

multiMC_along_PCA is a script for the parameter search with an iterative Monte-Carlo method. Parameters for simulations are (if present in the workspace, used as defaults):

- **simulationnumber**, number for auto backup and recovery
- **kcheckfunction** , handle to the condition function **kcheck(oModel,vk[,mainX0])** returning a logical value. Name of the function should be **kcheck_modelname*.m** where **modelname** is the model used and * can be anything.
- **mainX0** , initial conditions for **kcheckfunction** (not necessary if kcheck accepts only 2 arguments).
- **nbsets** , number of parameters tested per step
- **nbhits** , number of maximal viable parameters per step
- **maxmajor** , maximal number of steps

Parameter volume integration

volumeMC_along_PCA is a script for the integration of the parameter volume with a Monte-Carlo method. Parameters for simulations are (if present in the workspace, used as defaults):

- **simulationnumber**, number for auto backup and recovery
- **kcheckfunction** , handle to the condition function **kcheck(oModel,vk[,mainX0])** returning a logical value. Name of the function should be **kcheck_modelname*.m** where **modelname** is the model used and * can be anything.
- **mainX0** , initial conditions for **kcheckfunction** (not necessary if kcheck accepts only 2 arguments).
- **nbsets** , number of parameters tested
- **nbhits** , number of maximal viable parameters

volumeMC_along_PCA2 is the same script but does not save rejected parameters. It is therefore faster and allows testing of more parameters.

For further details on the algorithm, refer to the article on the method (Hafner M, Koeppl H, Hasler M, Wagner A (2009) 'Glocal' Robustness Analysis and Model Discrimination for Circadian Oscillators. PLoS Comput Biol 5(10): e1000534.**)**.