Using PeTTSy, Perturbation Theory Toolbox for Systems

Mirela Domijan, Paul Brown, Boris Shulgin & David Rand

Warwick Systems Biology Centre

It will help the reader to understand the theory behind this software if these notes should are read in conjunction with the following papers:

(1) D. A. Rand. Mapping the global sensitivity of cellular network dynamics: Sensitivity heat maps and a global summation law. *J. R. Soc. Interface* (2008) **5** S59-S69 doi:10.1098/rsif.2008.0084.focus

(2) D. A. Rand *et al.* Uncovering the design principles of circadian clocks: Mathematical analysis of flexibility and evolutionary goals, *J. Theor. Biol.* (2006) **238** 616-635.

Table of Contents

1. Installing PeTTSy

The programs require Matlab R2008a or later, plus the Symbolic Maths Toolbox is required in order to generate the model files. Place the distribution file in the directory in which you wish to install the program and unpack it. This should re-create the original directory structure, making directories docs, force, java, lcycle, models, sbml, SDS, shared, symbolic, theory and xpp.

You must now add the directory tree under the installation directory, referred to as <Install_Dir>, to your Matlab PATH. The easiest way to do this is as follows

```
>>cd <Install_Dir>
>>addpath(genpath(pwd))
```

Our software has the ability to export model definitions to XPPAUT format. This will need to be installed separately. See section 3.7.

Importing PeTTSy models from SBML requires the SBML toolbox to be on your MATLAB path. See section 2.8.

2. PeTTSy ODE models

PETSSy comes with a number of models supplied. The files defining these models are found in the directory <Install_Dir>/models/definitions. A complete model definition consists of the following files

- *modelname*_model.m containing the model ODEs, plus optional metadata.
- *modelname*.par containing a list of parameter names and default values.
- *modelname*.varn, an optional file containing a list of variable names and default initial conditions.

Before a model can be used, it must first be installed. These files are read by the `make` function, called via the GUI, in order to generate the files needed to run PeTTSy, which consist of the ODEs plus the derivatives of the model variables with respect to variable and parameter, see section 2.7 for details. In addition, the user can add their own model by creating the above required model files in this directory, see section 2.6.

2.1. The model equations

The file *modelname*_model.m defines the model ODEs. For example, this is how the three variable model 'neurospora', based on Leloup *et al* (1999), is represented.

```
function dydt = f(t, y, p)

   eval(p);

   dydt = [

      (vs + amp * force) * ki^n / (ki^n + y(3)^n) - vm * y(1) / (km + y(1));

      (ks + 0) * y(1) - vd * y(2) / (kdn + y(2)) - k1n * y(2) + k2n * y(3);

      k1n * y(2) - k2n * y(3);

   ];
```

`p` is an expression that will create all the model parameters appearing in the equations as symbolic variables. These are required by the MATLAB Symbolic Math Toolbox to calculate variable derivatives during the `make` process, see section 2.7.

Next a variable called `dydt` is created. For a model with *n* variables, this is a vector of length *n*, where the *n*th element is the derivative of the *n*th model variable. The derivatives can be functions of the following

- The state of the model variables. This is a vector `y`, with elements `y(1)` to `y(n)`.
- The model parameters, which are referred to by their symbolic names.
- The external force(s).

2.2. The external force

The model ODEs can include one or more force functions, which represent external force(s) applied to the model. Examples of forces include the addition of a signaling factor at a specified timepoint, or the external environmental cycle applied to a circadian oscillator. Force is either a constant value or a function of time, but always evaluates to a value between zero and one. There are a number of different force functions predefined by PeTTSy, in addition you can create your own, see section 2.9. The GUI allows any of the installed force functions to be selected before running a model.

Force is defined in the model ODEs using one of the reserved symbolic names. These reserved names consist of the name '`force`', plus '`force1`', '`force2`', up to '`force9`'. These are then assigned a selected force function via the GUI when the model is run.

Each external force requires two parameters to be defined, which is done via the GUI when the model is run. These are termed '`dawn`' and '`dusk`'. This is for historical reasons as the first force function, '`photo`', is a square wave representing photoperiod applied to a model of the circadian clock. For other forces one or both of these parameters may have no effect. The table below summarises the built in forces that are initially available.

Force	Description	Dawn	Dusk
`photo`	a square wave representing an on-off signal.	time of force = 1	time of force = 0
`impulse`	signal switched briefly from 0 to 1 at dawn.	time of impulse	NA
`Hoffman`	signal goes from 0 to 1 sharply at dawn and back to 0 at end of the cycle period (or the end of the run, in the case of signal models with no cycle period).	time of force = 1	NA
`sinewave`	sine wave with period equal to the length of the cycle period (or the length of the run, in the case of signal models with no cycle period)	Determines phase of wave	NA
`noforce`	force is zero throughout	NA	NA
`cts`	force is 1 throughout	NA	NA
`60`	Force switched on for 300 time units at t = 0, t = 3 600, t = 7 200, and t = 10 080 time units.	NA	NA
`100`	Force switched on for 300 time units at t = 0, t = 6 000, t = 12 000, and t = 18 000 time units.	NA	NA
`200`	Force switched on for 300 time units at t = 0, t = 12 000, t = 24 000, and t = 36 000 time units.	NA	NA

2.3. Model metadata

The model definition file may optionally finish with some parameters which provide metadata about the model to PeTTSy. Each line must begin with three % symbols and take the form

```
%%%name value
```

The following table shows which parameters are recognized by the `make` process.

Name	Possible Values	Meaning
`info` Several lines can begin with this	Any text	Comments that are for information only

These three settings define the model and cannot be changed		

`orbit_type`	`oscillator`
	
	(Default)
	
	`signal`
		Whether or not the system is periodic. PeTTSy will attempt to find a limit cycle when running an oscillator model

`positivity`	`non-negative`
	
	(Default)
	
	`allow_negative`
		Tells the ODE solver whether or not to allow any model variables to take negative values. Note that only the Matlab solvers support this property. CVode will ignore it.

`plotting_timescale`	Numerical value (Default = 1)	Scaling factor required when plotting time series, eg if the computation is in minutes, but the plot is required in hours, then this value is 60. The default assumes both are in hours
Other values can be altered in the GUI at runtime, but for convenience the values here provide defaults which will be selected initially		
`tend`	A positive numerical value (Default = 100)	The end time for a signal system. Ignored for an oscillator

`method`	The default solver to use. Options are one of Matlab's built in solvers or CVode. You can also specify a stiff or non-stiff problem. Possible values are
	`matlab_stiff`
	`matlab_non-stiff`
	`cvode_stiff`
	`cvode_non-stiff`
		User can select which solver to use in the gui, but this one is chosen by default. If the Matlab solvers are selected, `ode45` or `ode15s` will be used for non-stiff and stiff problems respectively.

`cycle_period`	A positive numerical value (Default = 24)	The default cycle period of periodic forces for oscillator models. Does not apply to signal models.
`force_type`	*`forcetype dawn dusk`* (Default = '`photo 6 18`')	The type of force to be applied at runtime, together with the parameters dawn and dusk, Models with multiple forces have them all defined on a single line in a comma separated list. See section 2.6.

Note that if the model equations do not contain a force term, then it is important that `force_type` is not defined here.

2.4. The parameter values file

The file *modelname*.par contains a list of all the symbolic names of the model parameters appearing in the ODEs above, and their corresponding default values, and optionally a brief description of each parameter. If parameter descriptions are omitted, then the symbolic name for the parameter is used by the GUI functions where a description is specified. There should be one line for each parameter, thus the file should have the form

Param_name Param_value Param description

For example, for the *neurospora* model above the first line might read

vs 1.6 max rate of mRNA transcription

2.5. The variable names and initial conditions file

The file *modelname*.varn contains a list of the names of the model variables, with their order corresponding to the order of the elements of `dydt` defined as above. Each line should also contain a default value for the initial state of that variable, and optionally, a brief description of the variable. If the variable descriptions are omitted, then the symbolic name for the variable is used by the GUI functions where a description is specified. There should be one line for each variable, thus the file should have the form

Variable_name initial_value Variable description

For example, from the above model the first line might read

mFRQ 0.0 FRQ mRNA level

This whole file is optional, and if omitted a default one will be created by naming the variables `y1` .. `yn`, and setting the initial conditions to all zeros.

2.6. Creating your own models

To create a new model the user must create the files described in sections 2.1 to 2.5 and place them in <Install_Dir>/models/definitions. Create an m file called *newmodelname*_model.m. An outline for the new model's ODEs would look like this

```
function dydt = f(t, y, p)

   eval(p);

   dydt = [

      % your model ODEs go here



   ];
```

When writing the equations, note the following restrictions

- The model parameter names must consist only of alphanumeric characters and underscores, and begin with an alphabetic character. The vector y is the model variables. The indices of this vector should correspond to the order of the variable names in the *modelname*.varn file if one is supplied.
- Parameters cannot be called 'p' or 'y' as these names refer to parameter and variable vectors respectively, which are input arguments to the function. They cannot be called 't' as this represents time; or 'force', 'dawn' or 'dusk'; as these refer to the external force.
- Do not use symbols which have a special meaning to Matlab. Parameters cannot be called 'i' or 'j', as these names represent imaginary numbers and are not considered to be parameters by Matlab's symbolic maths functions. If you accidentally use them, you will probably find that they are replaced with `sqrt(-1)` when the model is created. Other examples with special meaning include 'alpha', 'eps' or 'pi'.
- If the model has only a single external force, (meaning only a single force function, though this may appear more than once in the equations), it should be named '`force`'. If the model has more than one force, name them `force1`, .. `force`*`n`*. Names up to `force9` are valid. Once you install the model, it does not matter in which order the forces appear in the model ODEs, they are sorted alphabetically, ie '`force`' coming first and '`force9`' last. This is the order in which the forces will be presented in the GUI. The symbolic terms for force in this order are replaced in the model equations by the force vector `force(n)`. For this reason, unless the model has only one force, its best not to use '`force`', but to call the first force '`force1`', otherwise symbolic names will not correspond to vector indices, ie '`force`' would be `force(1)`, `force1` would be `force(2)` etc…
- At runtime, force will always be a value between 0 and 1. Thus where it appears in the ODEs it is often multiplied by a parameter representing a scaling factor or amplitude.

Note see Section 3.7 Launching XPP, for additional restrictions if you wish your model to be compatible with XPPAUT

After the equations you can add any metadata required, as described in section 2.3. All of these parameters have default values, and so any can be omitted if the default is appropriate for your model. The default model is an oscillator, whose variables cannot have negative values, with a period of 24 hours.

When defining the default force type, if your model has multiple forces, define them all on one line as a comma separated list. Each definition includes default for type, then default dawn and dusk values. For example

```
%%%force_type photo 6 18, cts 10 100
```

would define the first force for this model as type 'photo' by default, and the second as type 'cts'. The symbolic names appearing in the equations are not included here, but the order in which the default force types are defined needs to correspond to the order into which the symbolic names will be sorted. For example, the above line would cause the GUI to initially set `force1` to 'photo', and `force2` to 'cts'.

The default definitions are optional. It is not necessary to define all, or even any of them. If there are fewer defined than are found in the equations, then extra forces of type 'photo 6 18' are appended to the list to make up the correct number.

Finally, you need to create the parameter values and variable name and initial conditions files, *modelname*.par and *modelname*.varn respectively, as described in the previous sections. The parameters file must contain all the parameters used in the model equations, and the variables file must contain precisely the same number of variables as there are equations.

Note that when adding initial condition values or parameter values to these files, you can use exponential notation of the form 1e-4, but do not use expressions such as 1*10-4 as they will make the file invalid.

2.7. Installing a model

Installation refers to processing the above files to generate the model derivative matrices so that the model can be used by PeTTSy. Go to the Model->Install existing model menu. From the box that opens, select a model form the drop down list and click Install. The list will contain all the models found in the definitions directory.

The PeTTSy directory tree contains two directories, or these will be created if they don't exist

`models/oscillator`

`models/signal`

A new directory with the same name as the model will be created in one of these, depending on the model type. If this directory already exists, you will be prompted to overwrite the existing model installation.

The installation process will create the following files in the *modelname* directory.

modelname.m

This file contains the model equations. These are copied from *modelname*_model.m, with the parameter symbolic names being replaced by the vector `p`. The order of the elements of this vector will correspond to the order in which the parameters are listed in *modelname*.par. The symbolic names for force are replaced by the vector `force`, with the order of the elements of this vector corresponding to the alphabetically sorted order of the symbolic names for force. For a single force model, this will produce the single term `force(1)`, for multiple force models, `force(n)` should correspond to the symbolic variable '`force`*n*'. See section 2.6 for an explanation of this.

*modelname*_model.*m*

This file is simply a copy of the corresponding file in the model definitions directory. It is for information only and not used by PeTTSy. If you ever wish to edit and re-install the model, you must edit the file in the definitions directory. Editing this one will have no effect

modelname.par

This file is a copy of the corresponding file in the model definitions directory. It is used by PeTTSy to provide parameter names.

modelname.pv

This file lists the default parameter values, in the order corresponding to *modelname*.par, and also `p(n)` in the model equations. The purpose of this type of file is to provide parameter values at runtime when solving the equations. The GUI allows the user to create their own pv files to use in addition to this default set.

modelname.varn

This file is a copy of the corresponding file in the model definitions directory. It is used by PeTTSy to provide variable names. If it does not exist in the definitions directory, then a default version will be created with variables named '`y1`', …'`yn`'.

modelname.y

This file lists default initial conditions in the order corresponding to *modelname*.varn, and also `y(n)` in the model equations. The purpose of this type of file is to provide initial conditions at runtime when solving the equations. The GUI allows the user to create their own y files in addition to this default set. If the file *modelname*.varn does not exist in the definitions directory, then this file will consist of all zeros.

modelname.fv

The default force values. There will be one line for each external force found in the model, taking the form

name type dawn dusk

The first value on each line, 'name', is the symbolic name for the force, ie `force`, `force1` etc…, and the second value 'type' is one of the force functions as described in section 2.2. The purpose of this type of file is to provide force values at runtime when solving the equations. The GUI allows the user to create their own fv files to use in addition to this default set.

modelname.info

This file contains values for all the metadata described in section 2.3, either read from *modelname*_model.m, or the default values for any undefined properties. This file is used by the GUI at runtime to provide information about the model.

derivatives/*modelname*_jac.m

The model's Jacobian matrix, the derivatives of the ODEs with respect to the model variables. This matrix takes the following form

$$\begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial y_1} & \cdots & \frac{\partial f_n}{\partial y_n} \end{bmatrix}$$

where $f_i(y) = \frac{dy_i}{dt}$.

It is used during the analysis of the time series.

derivatives/*modelname*_jac_pattern.m

A matrix to indicate the positions of the non-zero values in the model Jacobian. This is used by the ODE solvers to speed up calculations.

derivatives/*modelname*_dp.m

The model's vector field derivative matrix with respect to the parameters, including the dawn and dusk values for each model force. This matrix takes the following form.

$$\begin{bmatrix} \frac{\partial f_1}{\partial k_1} & \cdots & \frac{\partial f_n}{\partial k_1} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_1}{\partial k_n} & \cdots & \frac{\partial f_n}{\partial k_n} \\ \frac{\partial f_1}{\partial force_1} \cdot \frac{\partial force_1}{\partial dawn_1} & \cdots & \frac{\partial f_n}{\partial force_1} \cdot \frac{\partial force_1}{\partial dawn_1} \\ \frac{\partial f_1}{\partial force_1} \cdot \frac{\partial force_1}{\partial dusk_1} & \cdots & \frac{\partial f_n}{\partial force_1} \cdot \frac{\partial force_1}{\partial dusk_1} \\ \cdots & & \cdots \\ \frac{\partial f_1}{\partial force_N} \cdot \frac{\partial force_N}{\partial dawn_N} & \cdots & \frac{\partial f_n}{\partial force_N} \cdot \frac{\partial force_N}{\partial dawn_N} \\ \frac{\partial f_1}{\partial force_N} \cdot \frac{\partial force_N}{\partial dusk_N} & \cdots & \frac{\partial f_n}{\partial force_N} \cdot \frac{\partial force_N}{\partial dusk_N} \end{bmatrix}$$

This matrix includes expressions for the derivatives with respect to all dawn and dusk values. As the force equation is not determined until runtime, these expressions are represented by the derivative of force with respect to dawn/dusk, calculated at runtime, multiplied by the derivative of the variable with respect to force, which is known at 'make' time. There will be one row in this matrix for each model force, with the order being determined alphabetically, as described in section 2.6

xpp/*modelname*.eqn

This file contains the model equations in symbolic form. It is used to create an ode file for XPPAUT, which defines initial conditions, parameter values, force type, and the length of time to run the simulation for. See section 3.7 for an explanation of how PeTTSy is integrated with XPPAUT.

2.8. Importing a model from SBML

PeTTSy allows the import of SBML models, as long as you have the SBML Toolbox present on your Matlab PATH. This can be downloaded from http://sbml.org/Software/SBMLToolbox. The toolbox requires libSBML , which can be downloaded from http://sbml.org/Software/libSBML. Follow the installation instructions for your platform and ensure that the directory containing the libsbml library file (libsbml.so, libsbml.dll or libsbml.dylib, depending on your platform) and associated m files is also present on your Matlab PATH.

Go to the Model->Install model from SBML menu. A series of screens will then take you through the import process. At the end of this process the model will be installed and ready to use.

There are a number of restrictions on the SBML that can be imported and these are outlined at the

start of the process. SBML level 2 and later are supported. If you have a model in level 1 format and a Matlab SimBiology toolbox license, you can convert to level 2 format as follows:

```
mdl = sbmlimport(level1filename.xml);
notes = mdl.Notes;
%Without this, final sbml file has extra notes tags within
%<notes><body> block
notes = regexprep(notes, '<notes>(.*)</notes>', '$1');
mdl.Notes = notes;
sbmlexport(mdl, level2fname.xml);
```

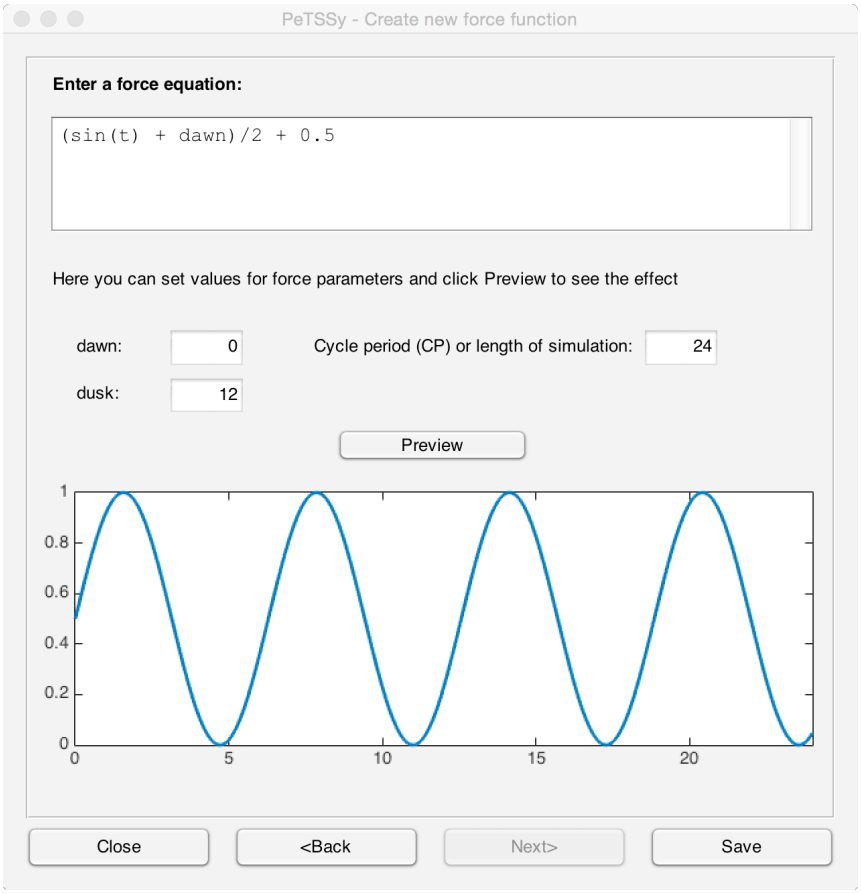
2.9. Adding a new force definition

If you require a model to have a new type of force not included in the default PeTTSy installation the process is simple. Go to the Force->Install new force menu. A screen will open that explains how to define a force equation.

The force equation can be either a constant value or a function of time, but must always evaluate to between zero and one. If a function, it must be a continuous function whose second derivative exists at all time points over which the simulation will be run. You can use literal values, Matlab functions such as `sin` or `exp`, plus the following symbolic names only: `dawn`, `dusk`, `CP` (for cycle period) and `t` to represent time. No model parameters for variables can appear in the force equation.

It is very important for oscillator models that PeTTSy is able to determine whether a force is constant or varying. Such a model run with a varying force (which will be repeated with an interval equal to the cycle period) is considered to be a forced oscillator, and if run with a constant force, is considered to be an unforced oscillator. The force expression will be examined to determine if it is a function of time, ie it includes the variable `t`. A force which is not a function of time is considered to be constant. It is important that this is accurate so please bear this in mind when creating your own force expressions.

On the first screen enter a name for the new force and click Next. Here you can enter your equation and enter dawn, dusk and cycle period values in order to test it. Click Save when complete.



2.10. Re-installing models

You may wish to edit a model by changing the model equations or metadata. To do this edit the files *modelname*_model.m, *modelname*.par, *modelname*.varn in <Install_Dir>/models/definitions as required. These files provide the source of the model definition used by `make()`, not the copies found in the model installation directory. Then re-install the model as before and answer yes when prompted to overwrite the old model.

2.11. Exporting models to SBML format

PeTTSy models can be exported to SBML Level 2 format. To export the current model, go to the Model->Export to SBML menu. A series of screens will then guide you through the export process. Model variables will become SBML species objects and parameters will become SBML parameter objects. You will be able to specify units for time and species amount or concentration. The model equations will be converted into SBML rateRule objects, with calls to the `force` function being converted to calls to a MathML function. The force function(s) will be defined as SBML functions, with the selected force being used to generate the function body.

Note that java error messages sometimes appear when generating MathML. This is caused by the format of some equations and PeTTSy can usually fix this by altering the equation into something mathematically equivalent. If the export process succeeds, these messages can be ignored.

3. Using the Software

The software suite consists of two GUIs

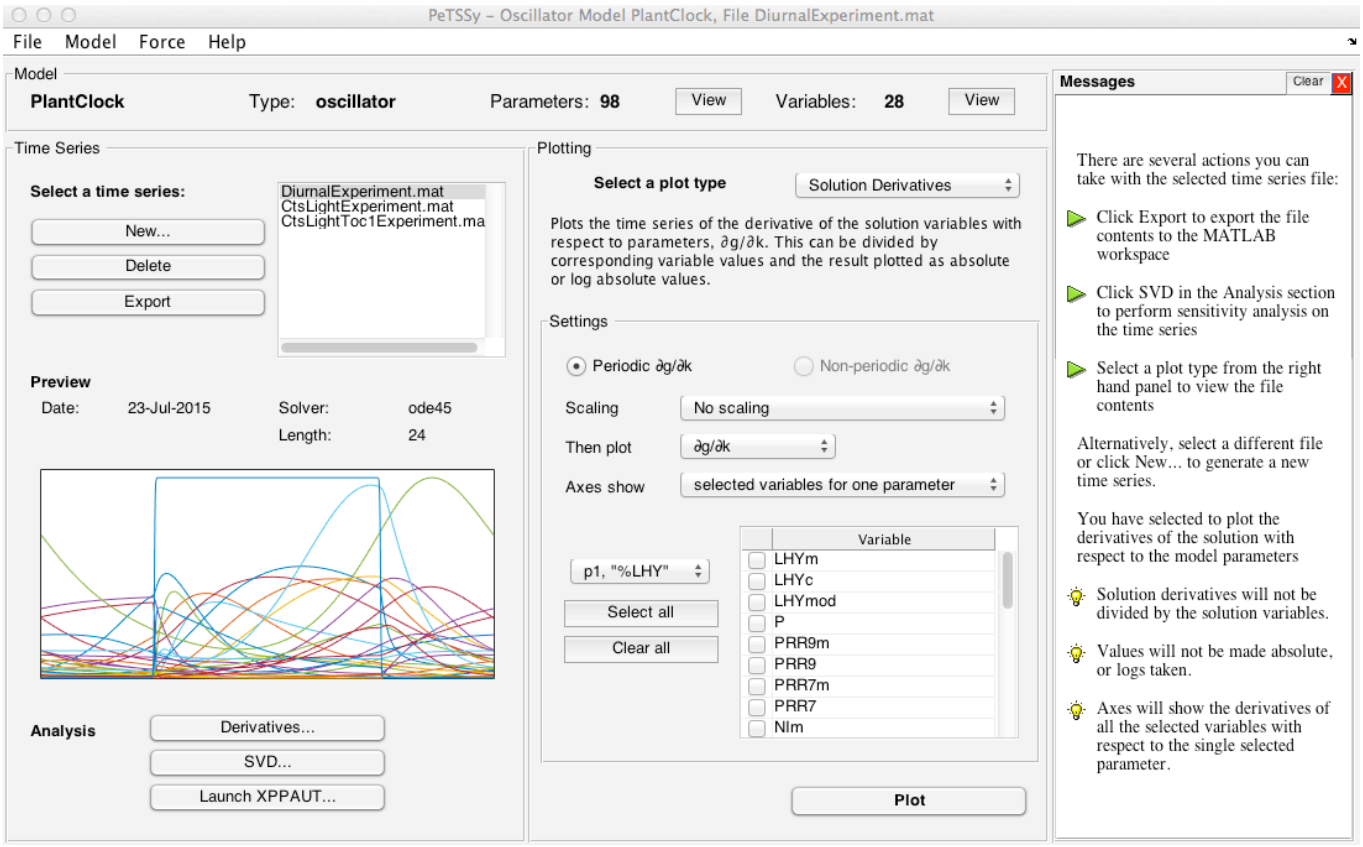
- `pettsy`, the main GUI, which is used to generate model time series, and generate the following analysis
    - Time series derivatives with respect to parameter
    - Derivatives of variable phase (peak and trough time) with respect to parameter
    - Infinitesimal response curves for oscillators
    - Period derivatives with respect to parameter for unforced oscillators
- `sagui`, which is used to perform singular value decomposition of the time series derivatives, and generate the following
    - Principal components with respect to parameter
    - Sensitivity of the time series solution to the principal components
    - Strength of the effect of varying the parameters on the principal components
    - Singular values
    - Derivatives of the time series amplitude/period/phase with respect to parameter and principal component

First you must generate some time series data for a selected model. From the installation directory, run the `pettsy` command to launch the main GUI.

At the top you will see details of the currently selected model. Click on the buttons to see lists of the model variables and parameters. The Model menu allows you to change the current model. The 'Time Series' panel on the left side of the GUI shows the available time series files. Select the one you are interested in or click New to launch a new simulation.

The Preview area displays some basic information about the selected time series, together with a preview plot.

You can obtain help or suggestions at any time by going to the Help menu and displaying the Tip window either at the foot or right hand side of the main window, depending on your screen dimensions. This window contains a summary of the current situation, together with a list of possible actions you may wish to take. It is written to continually while you use the PeTTSy GUI, even when not visible, but you will find the most recent information by scrolling to the end of this window.



3.1. Running a simulation

To generate a new time series, click the New… button to launch the New Time Series window. This allows you to specify parameters, initial conditions and the external force for your model.

To select a set of model parameters, choose a parameters file from the drop down list. Initially there will be just a single default file named after the model. Select this and the contents will be used to fill the grid below. From here the values can be edited. If you wish to save a new parameter set, edit as required and then click Save. If you change the file name, a new parameters file will be created. You can change the initial conditions required in exactly the same way.

You can also select a model force file from the available options. You will see the content of the file in the force table below. The row title(s) are the force symbolic names as appearing in the model equations. Force type can be changed by selecting one of the available types from the drop down list. Dawn and dusk values can be edited here. Again edits can be saved and new force files created. The force time series that would appear in the new simulation is also plotted.

Note that when the simulation is launched, it is the parameter, initial condition and force values appearing in the table that are used, and not necessarily the content of the selected files if you made edits without then saving them.

What happens after selecting the model force(s) depends on the mode and selected force type(s).

- For signal models, it is just necessary to enter a value for the length of the simulation.
- For oscillators when one of more of the selected forces are varying, ie a function of time, then it is just necessary to enter a value for the cycle period of the force. The force(s) are then defined over a time window determined by this value. The values of ‘dawn’ and ‘dusk’ determine the phase of events within this time block, and therefore each should have a value less than the cycle period. Selecting an oscillator with a varying force will ask the routines to find a limit cycle with period determined by the cycle period value, this being

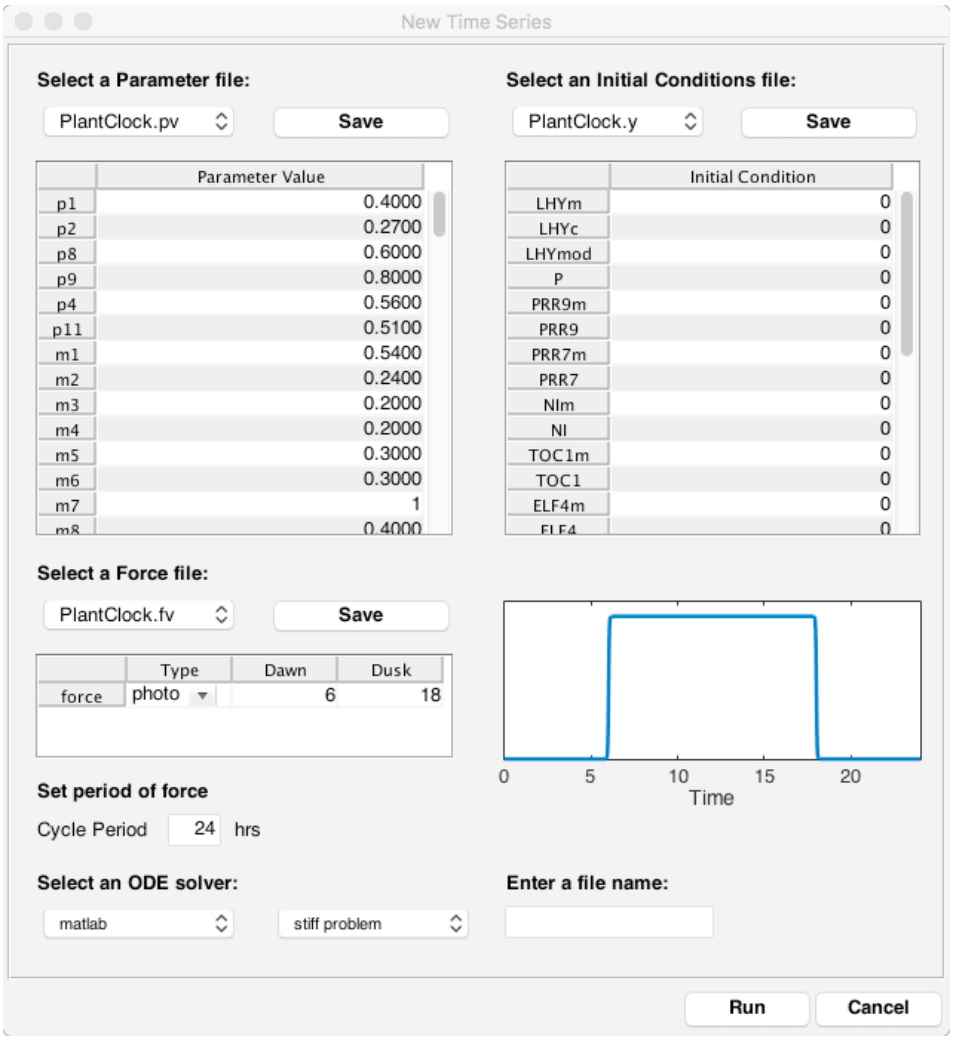
considered to be a forced oscillator system.

- For oscillators where all selected forces are constant, the force does not provide a phase reference or determine the period, and therefore the oscillator is unforced and the user must define phase in relation to the model variables themselves. This is done by specifying phase zero in terms of either the peak or trough of one of the model variables, plus or minus an offset. If 'Default' is selected for the variable to determine phase, then the routines will look for the variable with the longest time span between successive peaks during the initial entrainment of the system. This is to avoid selecting any variable which may have a multi-phasic waveform.

Finally you must select an ODE solver to use. The options are a built in Matlab solver, or CVode, a third party solver available from https://computation.llnl.gov/casc/sundials/main.html, though pre-installed in PeTTSy. You can specify a stiff or a non-stiff problem which will optimize CVode, or if a Matlab solver is selected, use `ode45` for non-stiff and `ode15s` for stiff problems. See Appendix 5.2 for advice on choosing a solver. Then enter a filename for the new simulation and click Run to launch the simulation and display the progress.

On completion, the new time series is saved as a structure called `theresults` in a mat file with the selected name placed in models/*modeltype*/*modelname*/results. See section 3.8 on exporting this structure to the MATLAB workspace for more information on the fields of this structure.

The New Time Series window is shown on the next page. In this example the model has one external force. The model is an oscillator, and the selected forces will produce a forced oscillator with a period of 24 hours. Enter a file name for the new simulation and click Run to begin. In the case of non-oscillator models, the simulation will simply run for the specified time. For oscillator models, the system will be allowed to relax to a stable limit cycle.



3.2. Possible problems when running a simulation

There are some potential issues that may cause the ODE solver to fail to find a valid solution. These can mostly be avoided by choosing appropriate values for your parameters. Problems arise when equations evaluate to either NaN (Not a Number) or Inf (Infinity).

The following expressions will evaluate to NaN:

- Raising a negative value to a non-integer power
- Zero divided by zero

The following expressions will evaluate to Inf:

- Zero raised to a negative power
- Division of non-zero by zero
- Taking log of zero

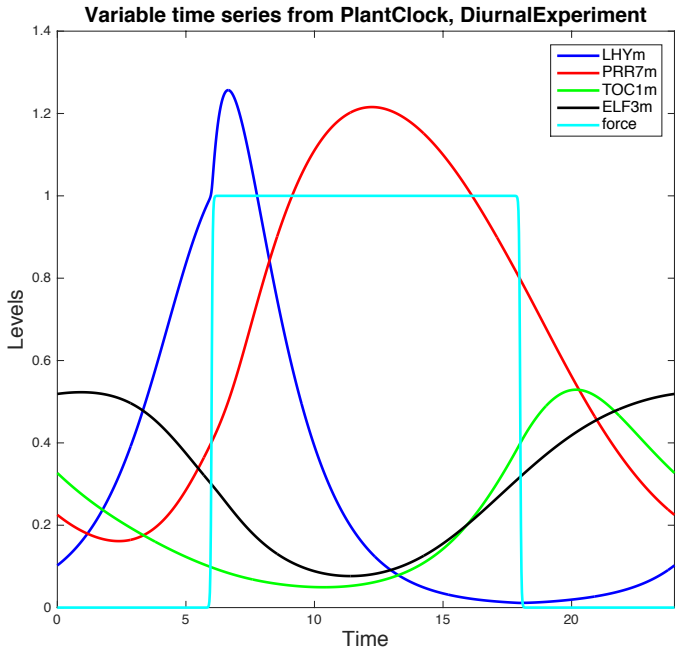
It should be possible to avoid these situations by avoiding setting parameter to zero or negative values. We note however, that there are some situations where the user may wish to have such parameters. For example, in creating mutant knock-out models, the user might need to set the particular relevant translation parameter(s) to zero. On occasions, this might cause the (aforementioned) problems when running the analysis, or later, when analyzing the time series generated (see Section 3.5 for further details). We refer the user to section 3.5 for an explanation of how to avoid these problems in this specific scenario.

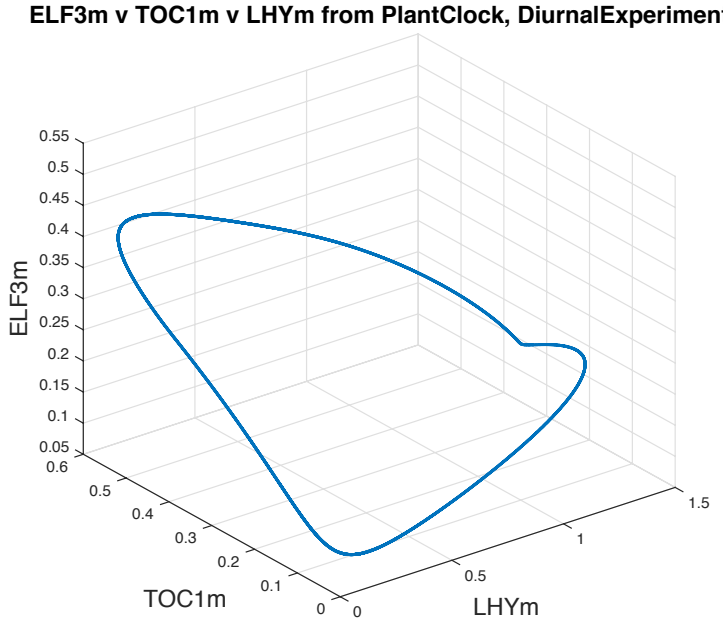
Should an equation evaluate to NaN or Inf, the simulation will fail and a form will open giving precise details of where the invalid expression occurred, and the parameter or variable values that were responsible. The idea is that the user can use this information to modify his/her model so that these errors do not occur. See for example the way that mutant models can be dealt with in Section 3.5.

Generally model variables are specified as non-negative (see section 2.3), however, even when this is the case, in rare circumstances, very tiny negative values can be produced by the ODE solvers, Negative solution values can also generate complex values (e.g when the ODE model has terms that require raising a negative value to a negative power or taking the log of a negative value). Once the solution is found, PeTTSy will simply discard the imaginary part, and if the model is specified as non-negative, it will set the negative real values to zero.

3.3. Plotting a time series

Once complete you can select the new time series file in the list on the main PeTTSy window. You will then see a summary plot below. However, it is possible to produce configurable plots in separate figure windows by using the Plotting panel on the right side of the PeTTSy window. First select 'Time series' from the list of plot types. The Settings panel then displays the times series plot controls. You choose either time, one of the model variables, or the external force for each axis. In the case of 2-D plots you can have multiple variables plotted on the y-axis. Checking the 'Normalise' box will scale each data series so that each has a mean value of one. Shown below are a 2-D plot of the three variables of the oscillator model *PlantClock*, (Pokhilko *et al,* 2012), versus time, and a 3-D plot of the same variables against each other, showing the limit cycle.





3.4. Analysing a time series

To calculate the time series derivatives etc…, select the file you are interested in and click the 'Derivatives…' button in the Analysis section. You will then be given a list of possible outputs. The options are as follows.

The following apply only to oscillator models.

- dy0/dpar in standard coordinate system (y0 is the initial conditions of the generated limit cycle)
- Curves under the integral for dy0/dpar
- Infinitesimal response curves
- Derivative of the period with respect to log parameter (unforced oscillators only)

The following apply to all models

- Derivatives of the time series with respect to parameter
- Derivatives of variable peak and trough times with respect to log parameter
- Derivatives of variables with respect to log parameter at peak and trough times

Checking only those you require can sometimes greatly speed up the analysis.

If you have a copy of the Matlab Parallel Computing Toolbox installed, then for oscillator models only, you will be given the option of running the analysis using one of your defined parallel configurations. This relies on the configuration being correctly set up to launch `matlabpool` or `parpool`. Choose the name of the configuration you want, or 'none' to run without the toolbox. You must also choose an appropriate number of workers from the drop down list. The maximum number that can be used is the greater of the number of model parameters or the number of model variables.

Note, at present PeTTSy only supports parallelization using Matlab R2013b or later.

Finally, you can opt to be given the opportunity to reject and re-calculate the fundamental matrices (X(s,t)) by increasing the time resolution. Here follows a description of what this means.

The time series solution is divided into a number of blocks, tb_i, tb_{i+1},…tb_N. By default, N= 70 blocks. The model Jacobian J(t) is integrated over each of these time blocks in both directions to give forward and reverse solutions for each time block.

Forward solution, fsols = X(s, t) where s is tb_i, t runs from tb_i to tb_{i+1} and is a solution of

$$\frac{dX(s,t)}{dt} = J(t) \cdot X(s,t)$$

with initial condition X(s,s) = Identity matrix.

Reverse solution, rsols = X(s, t) where t is tb_i, s runs from tb_i to tb_{i-1} are solutions of the adjoint equation:

$$\frac{dX(s,t)^T}{ds} = -J(s)^T \cdot X(s,t)^T$$

with initial condition X(t, t) = Identity matrix.

This is equivalent to solving,

$$\frac{dX(t-u,t)^t}{du} = J(t-u)^T \cdot X(t-u,t)^t$$

where u runs from 0 to tb_i-tb_{i-1}.

The results of this is that fsols and rsols are fundamental matrices which allow us to construct any solution, X(a,b), where if a < tb_i and b > tb_i, we can write the following

$$X(a,b) = X(tb_i,b) \cdot X(a,tb_i)$$

where X(a, tb_i) comes from $rsols_i$ and X(tb_i,b) comes from $fsols_i$. However, when a and b are inside the same interval, [tb_i, tb_{i+1}] then

$$X(a,b) = X(tb_i,b) \cdot inv(X(a,tb_i))$$

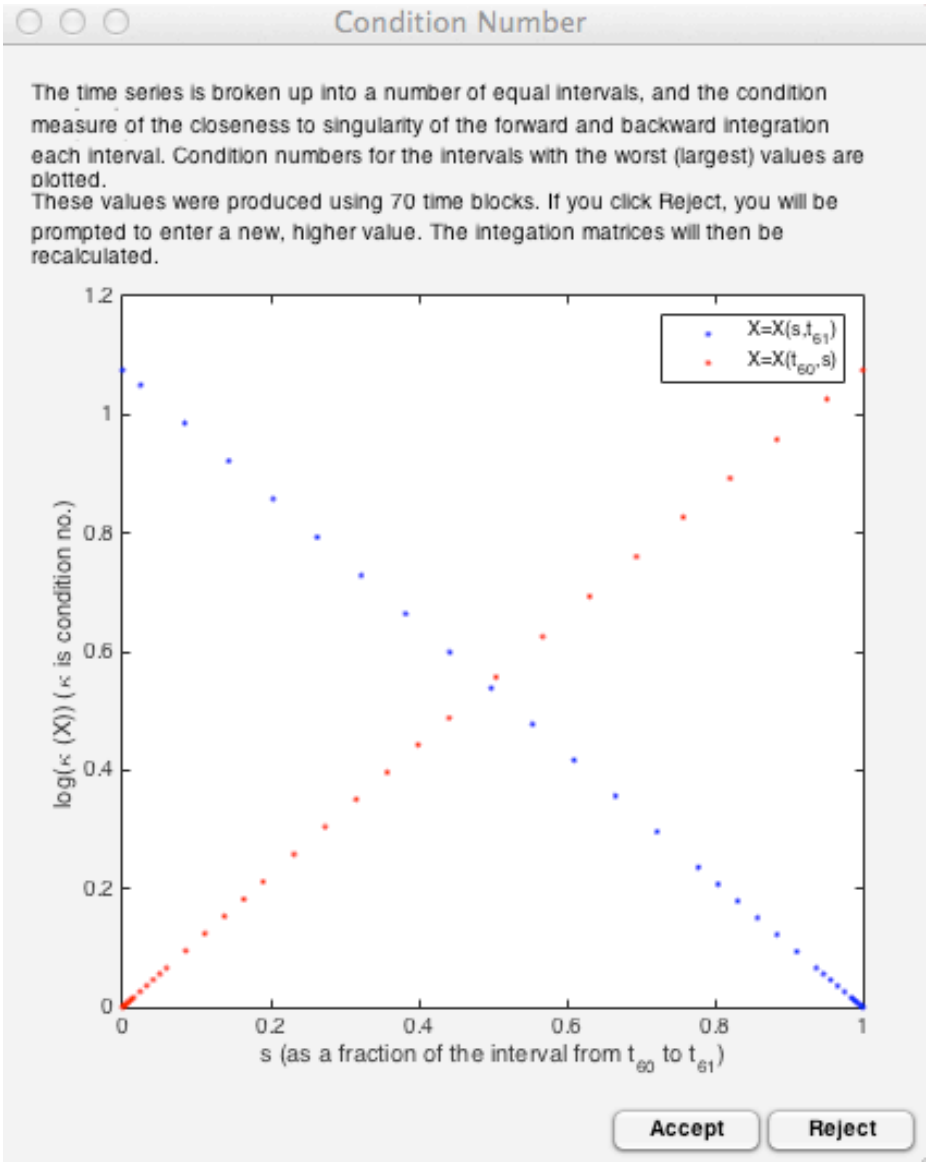
or,

$$X(a,b) = inv(X(tb_{i+1},b)) \cdot X(a,tb_{i+1})$$

namely, we have to invert either rsols or fsols in order to calculate X(a,b). Condition number is equal to one over the eigenvalue, and if it is too large, the matrix may be close to singular and then the inverse inaccurate. This means that fsols and rsols were not calculated finely enough, i.e., there were too few time blocks, and this will affect the accuracy of the results.

If you opt to check the accuracy of the X_{st} matrices, then after fsols and rsols have been calculated, you are presented with a plot as shown below for *neurospora* model (Leloup et al. 1999).

The time block with the worst (highest) condition number is shown, in this case tb_{26} – tb_{27}. The red plot is the forward solution, X(s, t_{27}), and the blue plot the reverse solution, X(t_{26}, s). Condition numbers are plotted against s as a fraction of the interval tb_{26} – tb_{27} .For the forward solution, condition number will increase as s gets closer to tb_{27}, and for the reverse solution, condition number will decrease as s gets closer to tb_{27}. As a guide, if condition numbers are above 1/eps, where eps is the floating accuracy of Matlab, the inverse is not accurate. On the above plot this corresponds to log(condition number) > 36.04, if eps = 2^{-52}.



The time block with the worst (highest) condition number is shown, in this case $tb_{26} – tb_{27}$. The red plot is the forward solution, $X(s, t_{27})$, and the blue plot the reverse solution, $X(t_{26}, s)$. Condition numbers are plotted against s as a fraction of the interval $tb_{26} – tb_{27}$ .For the forward solution, condition number will increase as s gets closer to tb_{27}, and for the reverse solution, condition number will decrease as s gets closer to tb_{27}. As a guide, if condition numbers are above 1/eps, where eps is the floating accuracy of Matlab, the inverse is not accurate. On the above plot this corresponds to log(condition number) > 36.04, if eps = 2^{-52}.

The selected outputs become a new field called '`theory`' of the time series structure. This is then saved back to the original file, with the variable now called '`thetheoryresults`'.

3.5. Possible problems when running an analysis

The situations described in section 3.2 (appearance of NaN and Inf values) may also occur when evaluating the model derivative matrices. As stated before, this should not happen when the parameters are chosen to be positive and the initial conditions are non-negative.

In section 3.2 we discussed the situation where the user may wish to create a knock-out mutant model, by just setting specific translation parameter(s) to zero. Here we illustrate the problems that might occur and also discuss how the user can get around them.

For example, consider the following ODE where transcription of protein A mRNA depends on a protein transcription factor B and is modeled by a Hill-type function,

$$\frac{dmRNA_A}{dt} = V_{max}\frac{\mathrm{Pr}_B^n}{\mathrm{Pr}_B^n + K^n} + (....).$$

Note that (…) in the above equation denotes other terms of the ODE that are not pertinent to our current discussion. The derivative of this transcription term with respect to the Hill coefficient (parameter *n)* will include the term *log* Pr_B. If the user wishes to model a protein B knockout mutation, where the parameter describing the translation of protein B is set to zero, then ultimately, the model solution should have no protein B, i.e. Pr_B =0. Having Pr_B =0 will result in a derivative with the *log 0 value* and the derivative will evaluate to Inf.

To avoid this problem, a better way to model the knockout mutation model above might be to re-make the model by removing the transcription term from the ODE equation. The user may also have the variable (s) representing protein B removed altogether from the model (i.e. perform a model reduction).

User should note that this problem may not be encountered when modeling all types of mutants, but if is it, then the above solution may be of use.

3.6. Plotting the analysis

To view the results of the analysis, select the time series file you are interested in and then choose a plot type from the Plotting panel on the right side of the PeTTSy window. Not all plot types apply to all analyses, it depends on which outputs were requested and whether the model is an oscillator, and if so, whether it is forced or unforced. If the selected plot type applies to the selected time series file, the plot settings controls are filled in and the Plot button is activated.

The plots described below will label axes with the names of the model parameters. These will include the specified dawn and dusk values for each model force, which are treated as model parameters for the analysis. These will be named *forcename*.dawn and *forcename*.dusk, where *forcename* is the symbolic name used in the model equation, ie '`force`', '`force1`', etc…

3.6.1. Plotting solution derivatives

This plot type applies to all models. It plots the derivatives of the selected model variables with respect to the selected model parameters over the course of the time series.

The derivative of the *i*th variable of a solution g(t,k) with respect to the *j*th parameter, at time t is given by the following term

$$\frac{\partial g_i}{\partial k_j}(t).$$

This derivative will be a periodic function of time in forced oscillators, but non-periodic in signal models and in general, non-periodic in unforced oscillators.

For signal models, the derivatives $\partial g_i / \partial k_j$ will always be non-periodic, and for forced oscillators they will always be periodic. For unforced oscillators $\partial g_i / \partial k_j$ will in general be non-periodic derivatives, since the period $\tau = \tau(k)$ can change with parameter changes. But we can rescale the model solution,

$$\gamma(t,k) = g(\bar{\tau}t,k)$$

where $\bar{\tau} = \tau(k)/\tau(k_0)$ with $\tau(k_0)$ as the period for initial parameter set values k_0, so that derivative $\partial \gamma_i / \partial k_j$ is a periodic function of time with period $\tau(k_0)$. This derivative can easily be calculated,

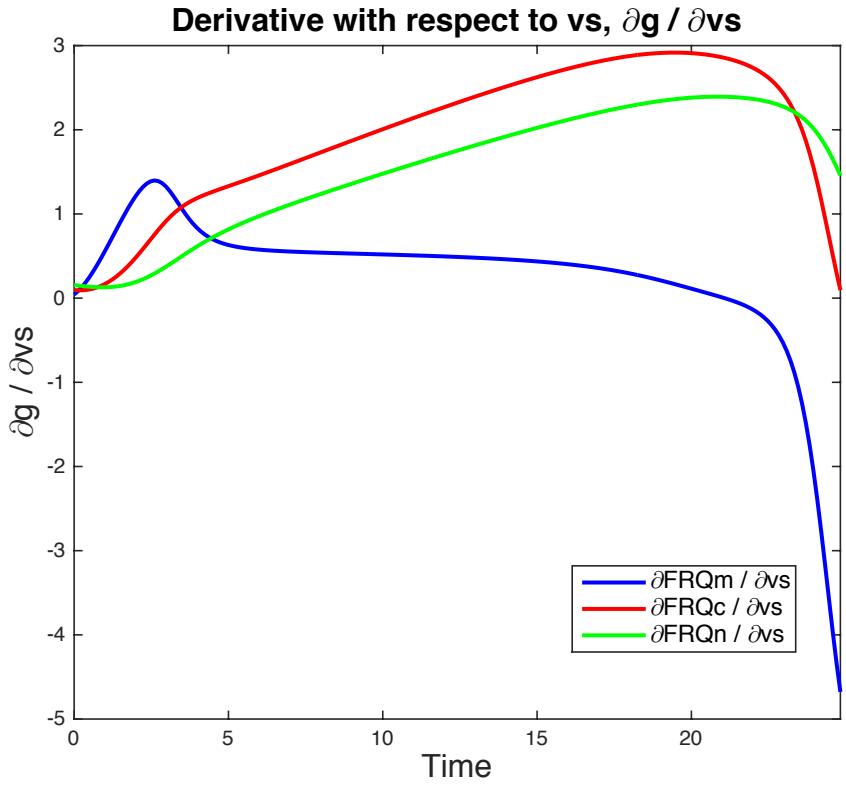
$$\frac{\partial \gamma_i}{\partial k_j}(t) = \frac{\partial g_i}{\partial k_j}(t) + \left(f_i(g) \cdot \frac{\partial \tau}{\partial k_j} \cdot \frac{t}{\tau(k_0)} \right).$$

For unforced oscillators the user is given a choice of calculating $\partial g_i / \partial k_j$ (non-periodic derivatives) and $\partial \gamma_i / \partial k_j$ (periodic derivatives).

Next you can opt to scale the derivatives by dividing by the variable time series. Each point will be divided by the corresponding variable value at that time point. You can then opt to take absolute values, or values of $\partial g_i / \partial \log k_j$. In the last case, if the parameter in question is zero, then the value of $\partial g_i / \partial k_j$ is given instead.

You then have the choice of creating a set of axes containing the derivatives of a single variable with respect to multiple selected parameters, or a set of axes containing the derivatives of multiple selected variables with respect to a single parameter. Choosing a parameter or variable from the drop down list will populate the table with a list of all the model parameters or variables as appropriate. These can be selected by checking the boxes.

The example below is taken from the *neurospora* model of Leloup *et al* (1999) and shows the non-periodic derivatives of all three variables from an unforced limit cycle, with respect to the parameter v_s.



3.6.2. Plotting period derivatives

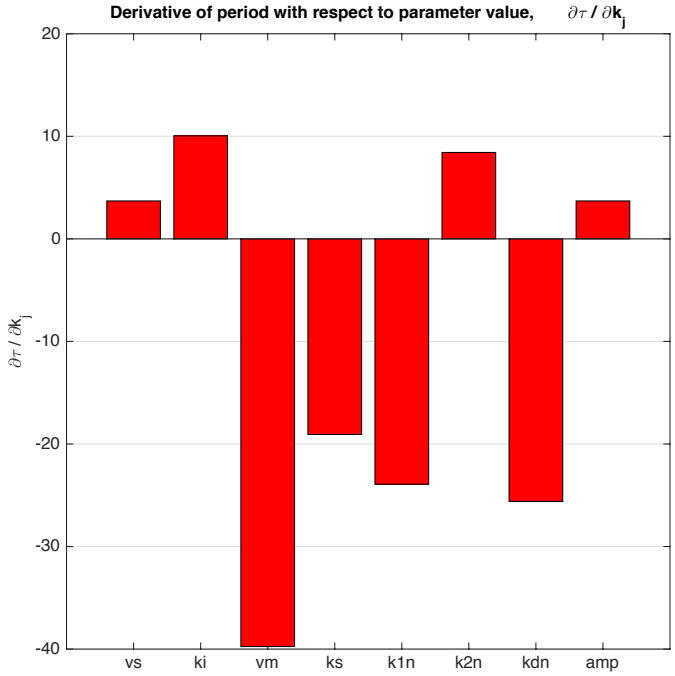
This plot type applies only to unforced oscillators, and plots the derivative of the oscillator period, **τ** with respect to the model parameters in one of three ways:

$$\frac{\partial \tau}{\partial k}, \frac{\partial \tau}{\partial \log k} \text{ or } \frac{\partial \log \tau}{\partial \log k}$$

There is also the option to take $\log_{10}$ of the absolute value of the above. The calculated derivatives are then displayed in a sortable table. Click on the column headers to sort by parameter name or derivative values, then select the required parameters. The header of the derivative column shows what has been selected to plot.

A special case arises for parameters having value zero. Here it is not possible to take logs, and so if the derivative with respect to log k is selected, these parameters are omitted from the plot.

The example below shows a plot of the period derivative for the most important eight parameters for the model *neurospora* (Leloup *et al*, 1999).



3.6.3. Plotting ircs

This plot type applies only to unforced oscillators and plots the infinitesimal response curves (IRCs) of the free running limit cycle in response to parameter perturbations. The Settings panel displays a table of model parameters with corresponding values for maximum phase advance (Max Adv, expressed as positive values), and maximum phase delay (Max Del, expressed as negative values) for an infinitesimally small perturbation of the parameter at a single time point during the period of the limit cycle. These values are in fact maximum and minimum (respectively) values of the IRC for that parameter. The last column shows the integral (i.e., the area under the curve) of the IRC over the whole limit cycle interval. The integral of the IRC is related to the period of the limit cycle. For more details the user is referred to (Rand *et al*., 2006). A value of zero would indicate that a permanent perturbation to that parameter (i.e. change of parameter for the whole of the limit cycle time) would cause no overall change to the phase (or period) of the limit cycle. A positive value would indicate an overall phase advance (and period shortening), and a negative value an overall phase delay (and period lengthening).

The table is sortable by clicking on the column titles, with plots for each selected parameter appearing in the same order in which they appear in the table.

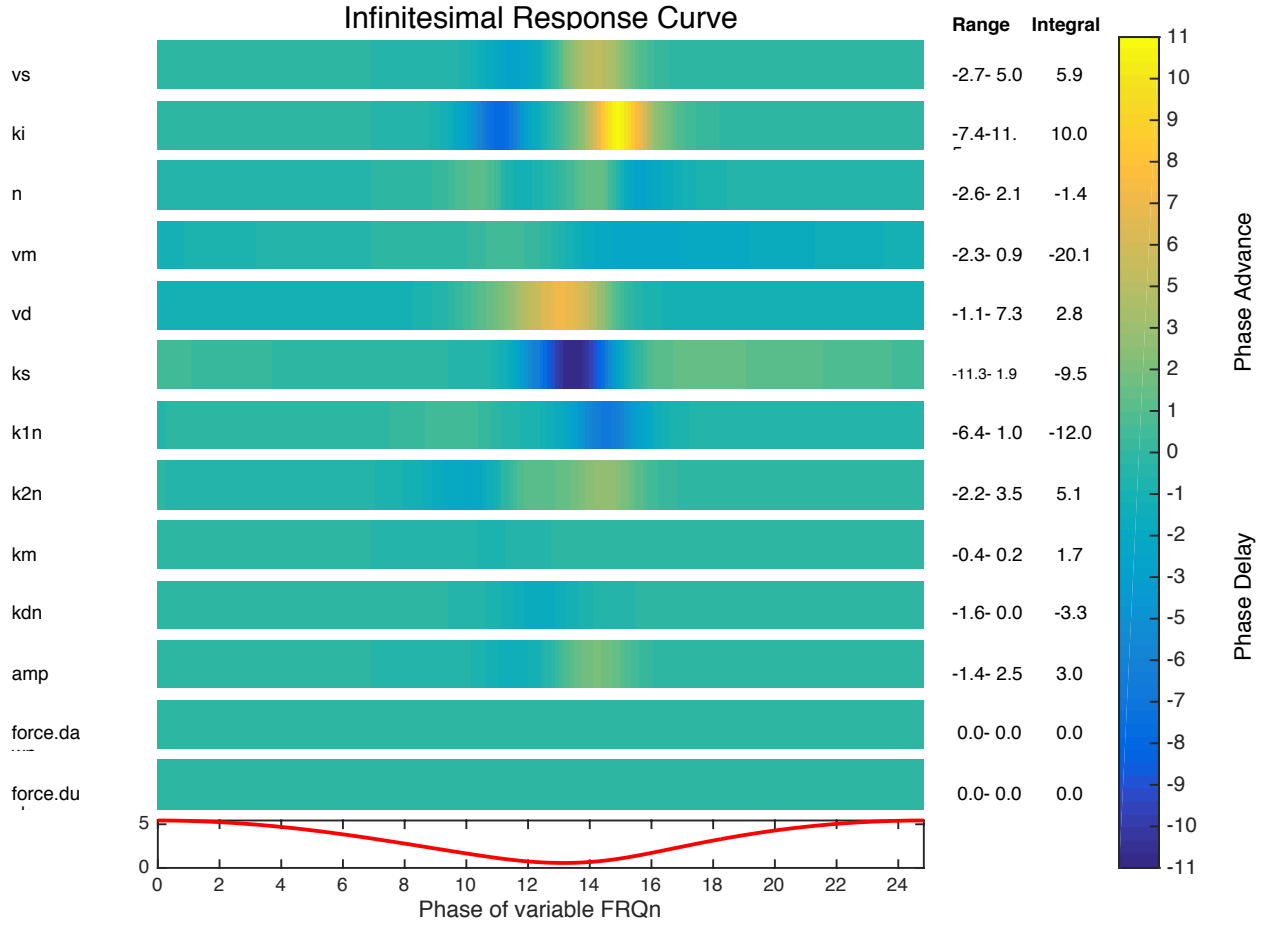
You can select from three plot types.

- Heat maps will plot each selected parameter on a horizontal heat map bar. This uses Matlab's current colour map (Jet by default) so generally phase delays will be blue, and advances red, with green representing no change. A vertical bar will indicate the scale used. If the selected scale is too small, any phase change that goes off the range is shown in the highest or lowest possible colour.
- Line plots will plot a series of XY plots, one for each parameter
- Single line plot will plot all parameters on a single set of axes.

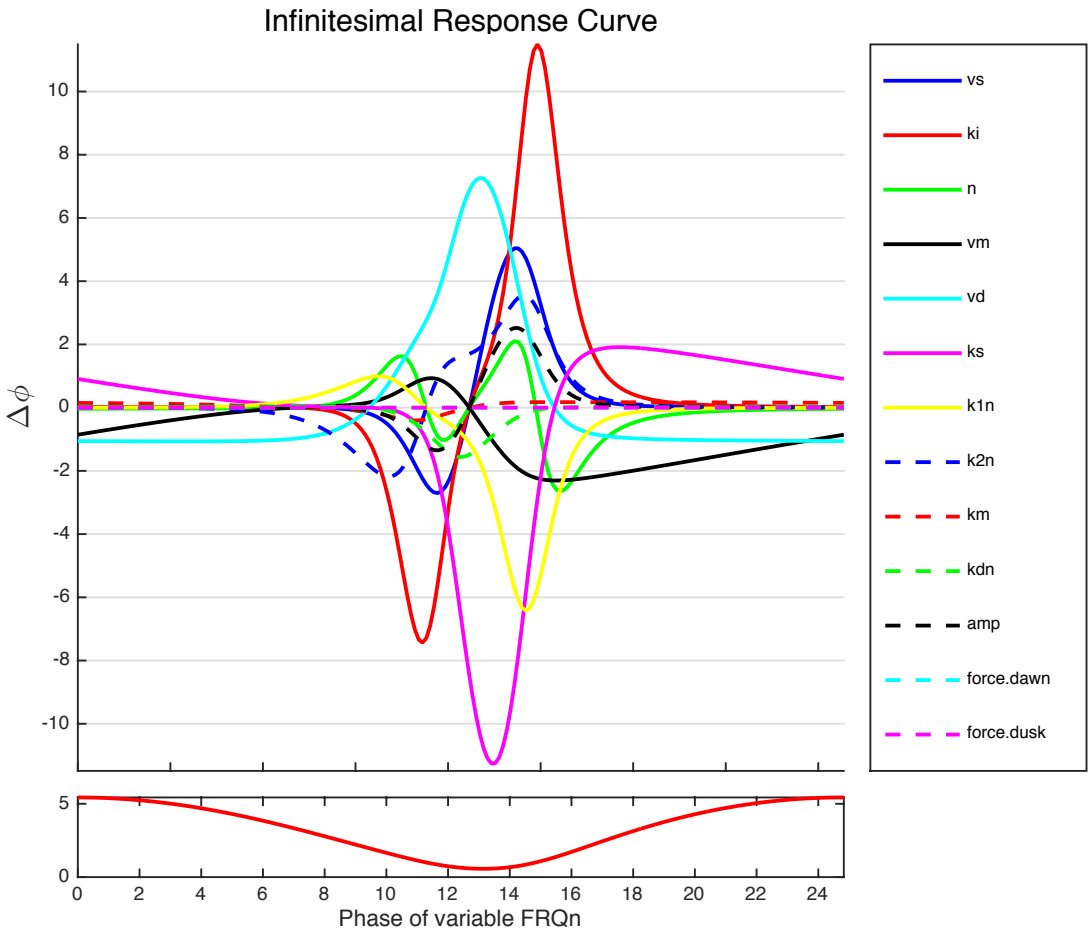
You can set the scale of the plots. In the case of heat map plots, the full colour map used will represent values from minus this value to plus this value, with zero phase change in the middle. In the case of line plots, the range of the y-axis will be +/- this value. If you select 'auto', then automatic scaling will be used. In this case the colour map or y-axis will run from +/- the largest of the absolute values of the maximum advances and delays taken from all selected parameters.

For the first two plot types, scaling will be the same for all plots, allowing easy visualisation of which parameters cause the largest phase changes. Each plot has the name of the parameter to the left and the phase change range and integral to the right. Moving the mouse pointer over one of these labels displays its contents as a tool tip. This is useful when there are a large number of plots and the labels are very small. Clicking on a plot brings up a larger line plot of the underlying data.

Additionally, a final set of axes shows one cycle period of time series data, represented by the variable used to define the start of the limit cycle. An example of a heat map plot is show below.



An example of a single line plot displaying exactly the same data is show below. In this case, clicking on an individual data series will bring up an individual plot.



3.6.4. *Plotting phase derivatives*

This plot type applies to forced oscillators only. It plots as a bar chart the derivatives of the phases of the model variables, measured either as peak or trough time, with respect to model parameters,

$$\frac{\partial \phi}{\partial k} \text{ or } \frac{\partial \phi}{\partial \log k}$$

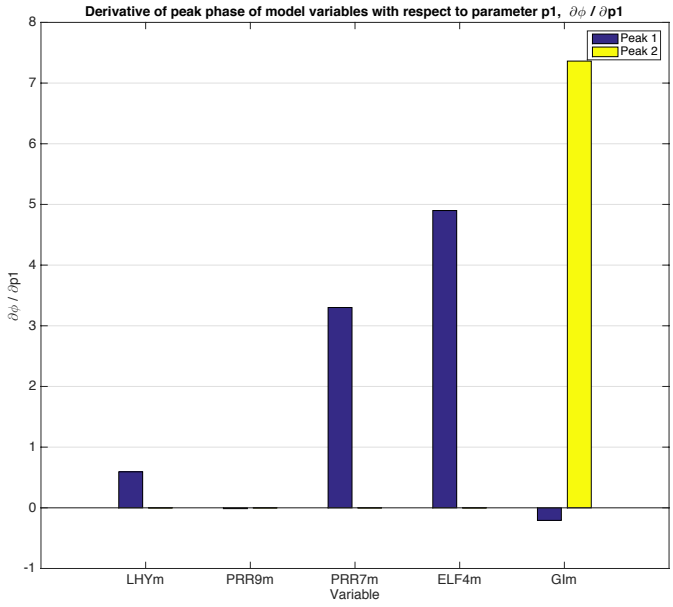
Firstly, you can select whether to use peak or trough time as a measure of phase, and whether to take derivatives with respect to parameter or to log parameter. If the latter, then a special situation arises for parameters of value zero. Logs cannot be taken and they will be omitted from the plot.

There is then an option to take $\log_{10}$ of the absolute value of the resulting derivative.

You have the choice of creating a set of axes containing the phase derivatives of a single variable with respect to multiple selected parameters, or a set of axes containing the phase derivatives of multiple selected variables with respect to a single parameter. Select the type of axes required, then choosing a parameter or variable from the drop down list will populate the table with a list of all the model parameters or variables as appropriate, together with the actual derivative values that will be plotted. The column title show the actual formula for the derivative that will be used.

In the case of variables with more than one peak or trough, they will be plotted on the same axes as separate data series. The table will show the largest derivative value across all peaks for that variable.

The example below is taken from the *PlantClock* model (Pokhilko *et al*, 2012). It shows the phase derivatives of selected variables with respect to one parameter, p1. Note that variable GIm is biphasic.



3.6.5. Plotting phase ircs

This plot type applies only to forced oscillators. The phase infinitesimal response curves (IRCs) represent changes to phases (peak times) of the model variables in response to parameter perturbations. Each curve has a discontinuity at the time of the peak in question.

Note that period of forced oscillators is fixed by the external force and so it will remain constant under parameter perturbations.

Change of phase is represented by a sum. First term of this sum is the integral of the phase IRC and the second term is additional value (a partial derivative, for details refer to (Rand, 2008)) that here we represent as a single point drawn at time of phase.

The rest of the interpretation of how phase changes as parameter is perturbed follows similar lines to the interpretation of the IRCs. Namely, if the selected parameter is perturbed over the whole limit cycle then change in phase is indicated by the area under the whole phase IRC and the single time point value. If the parameter is perturbed over the time interval that does not include the time of the phase, then the phase change is given only by the corresponding area under the phase IRCs curve.

For a permanent perturbation of the parameter, f the area under the phase IRC is positive and the aforementioned single point value is positive, then there will be a phase advance. Likewise, if both are negative, this will result in a phase delay. However, if they are found to be of opposite sign, then determining whether the perturbation will result in a phase advance or delay requires examination of their actual values. Note that the actual values can be exported (c.f. Section 3.8) if further analysis is needed.

Phase IRCs apply to a particular variable, and so firstly you must select the variable of interest. As variables can have more than one peak, you must then select the peak of interest from the drop down list. Usually there will only be a single choice, ‘peak 1’. However, with bi-phasic and multi-phasic variables you can select the required peak, or the final option, ‘all’. Note that the peaks are listed in chronological order, i.e., ‘peak 1’ is the peak closest to the start of the limit cycle.

Selecting a variable and peak will fill the parameter table with the appropriate values. This table displays the model parameters with corresponding values for maximum phase advance (Max Adv, expressed as positive values) and maximum phase delay (Max Del, expressed as negative values). These represent phase changes when the parameter is perturbed for a single time point point. The last column shows the total change to the phase when the parameter is perturbed permanently (.i.e., for the whole of the limit cycle).

In the case of multi-phasic variables, if you select all peaks, then the data shown will be the largest

values found across all peaks.

You can sort the table by clicking on the column headings, in which case when choosing to plot parameters on separate axes (Line plot), they will be plotted in the order in which they appear in the table.

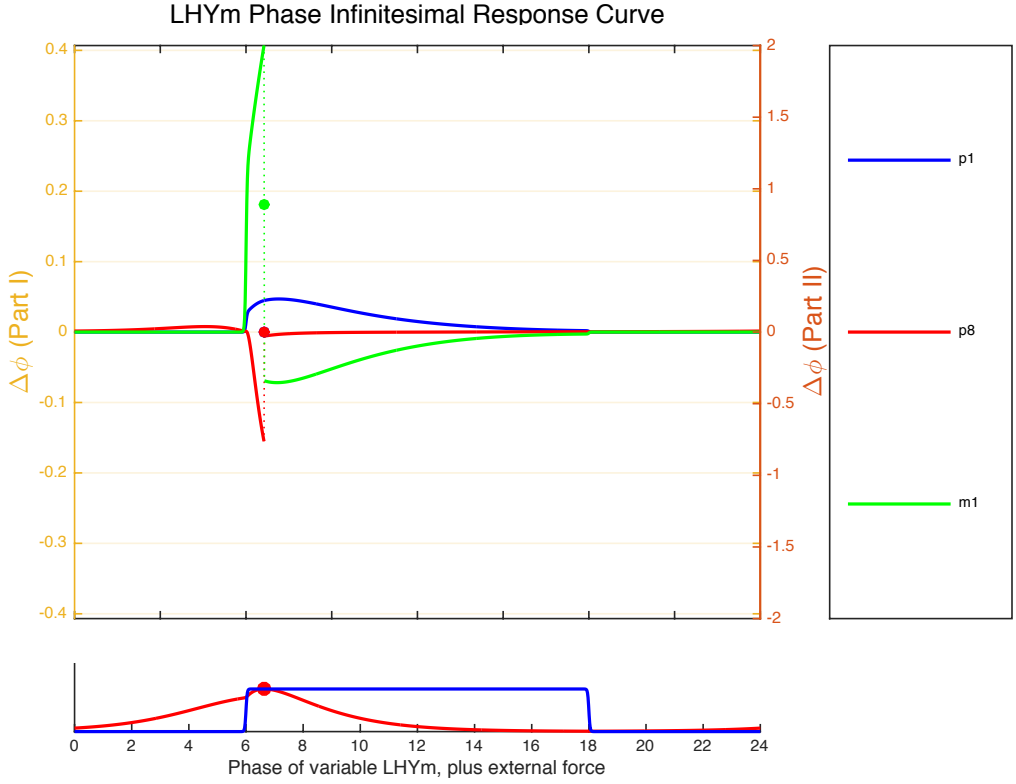
You can select from two plot types.

- Line plots will plot a series of XY plots. There will be one set of axes for each parameter, and one data series for each peak selected.
- Single line plot will create a set of axes for each selected peak, plotting all parameters on each.

You can set the scale of the plots, the range of the y-axis will be +/- this value. If you select ‘auto’, then automatic scaling will be used.

The scaling will be the same for all plots, allowing easy visualisation of which parameters cause the largest phase changes to which peaks. In the case of the first type of plot, each plot has the name of the parameter to the left and the range and integral to the right. Moving the mouse pointer over one of these labels displays its contents as a tool tip. This is useful when there are a large number of plots and the labels get too small to read. Clicking on a set of axes brings up a larger plot of the underlying data. In the case of the line plots displaying data from more than one peak, clicking on an individual data series brings up a plot of just that series.

An example of a single line plot is shown below. Phase IRCs are plotted with the discontinuities at the time of the peak represented by vertical dotted lines joining up the two halves of the curve. The partial derivative values added to the areas under the curves are shown as points plotted on the secondary y-axis at the time of the peak in question. Below, a final set of axes shows the time series of the selected variable. The variable times series is shown in red with peak selected indicated by a red dot with the plot of the forcing function overlaid in blue. In this case, clicking on an individual data series will bring up an individual plot.

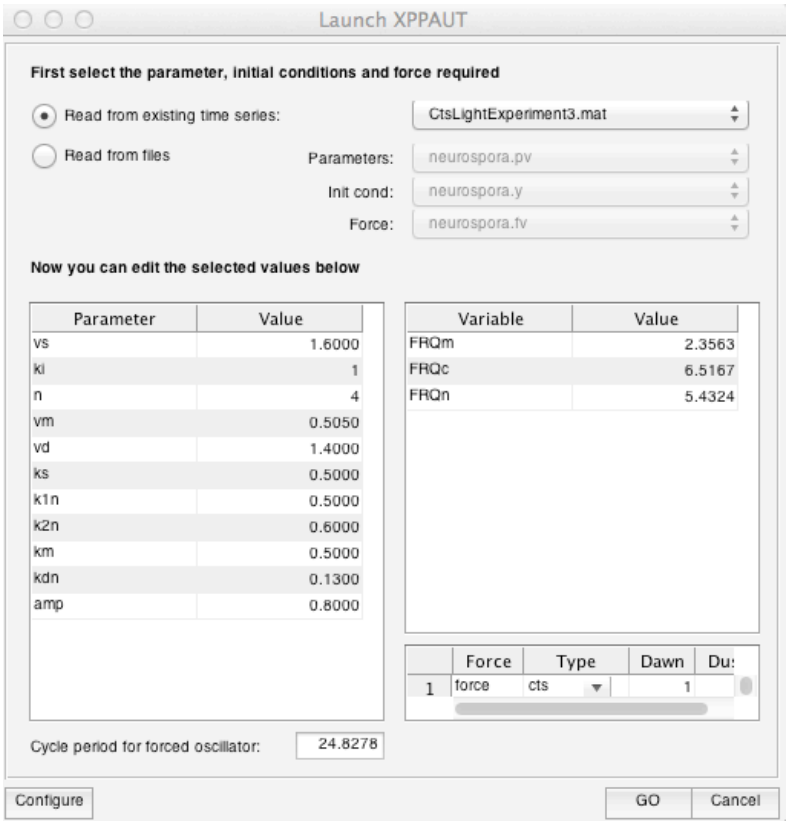


3.7. Launching XPPAUT

PeTTSy allows the models to be exported to XPPAUT by creating an ode file including the model equations, parameter values, initial conditions and force. Click the XPPAUT… button to open the form below.

The first time you do this you will need to tell PeTTSy where to find XPPAUT. Click the Configure button to open a form that will allow you to enter the XPPAUT home directory, that is, the directory containing the `xppaut` (UNIX) or `xppaut.exe` (Windows) application file. You can also optionally enter the location of your web browser and XPPAUT help directory. Defining these will allow XPPAUT to launch its html help files. This is independent of PeTTSy, which just provides a convenient way to add this information to the XPPAUT startup script. Clicking OK on this form will then create the required script to launch XPPAUT.

When you have done this, return to the form below to define the simulation you wish to run. The parameters, initial conditions and force type to be exported appear in the tables. These can be edited by hand. Alternatively, they can be filled from a pre-existing time series simulation file, by taking those values used to create the time series, or they can be filled by selecting any of the models parameter values (.pv), initial conditions (.y) or force(.fv) files. You need to also enter a time to run the simulation for. For a forced oscillator, this will be the period of the force.



Clicking GO should bring up the XPPAUT application, with the generated ode file selected. This file will be found in the model's installation directory, at `xpp/modelname.ode`

If XPPAUT is working on your system, but still will not launch from this form, then you can examine the launch script generated, at the following locations

`<INSTALL_DIR>/xpp/runxpp.bat` Windows
`<INSTALL_DIR>/xpp/runxpp.sh` UNIX

This can be edited by hand to fix the problem.

3.7.1. Limitations when defining your own models

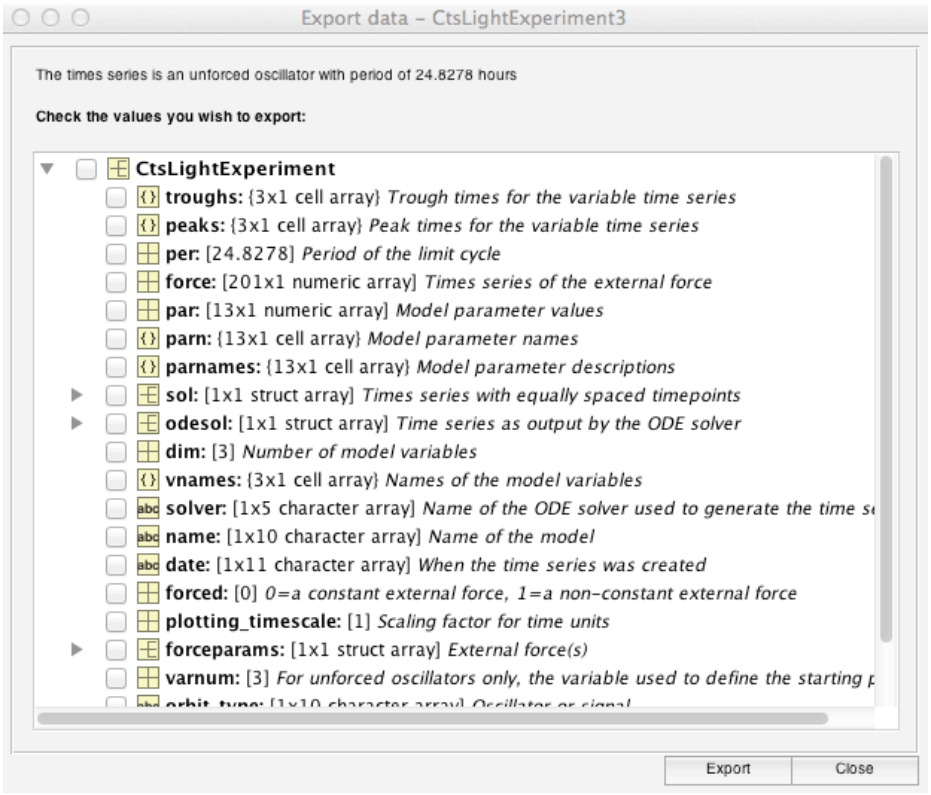
If you create your own model and you wish it to be compatible with XPPAUT, there appear to be some additional limitations on variable naming. This may not be an exhaustive list but these restrictions all apply to XPPAUT version 5.99

- Parameter and variable symbolic names cannot be more than ten characters long. This applies to the short names used in the equations for labeling plots, not the longer more descriptive names that are not used by XPPAUT.
- The following are XPPAUT key words and cannot be used in any name: `delay, ln, then, heav, flr, ran, normal, del_shft, hom_bcs, arg1 ... arg9, @ shift, not, sum, of`

3.8. Exporting data

PeTTSy allows you to export the data from the currently selected time series file to the MATLAB workspace. To do this select the File->Export data menu, or the Export button. The following window will appear, from which you can select the fields that to you wish to include in the export.

The expandable tree represents the hierarchy of the data structure. Any fields you check will be exported to a corresponding structure in the MATLAB workspace, which will be named after the original file, which is also the name of the top level entry in the tree. There now follows a brief explanation of the data fields. Not all of these will appear for all time series files of course. If you have not run the analysis to generate the derivatives and other output, then the 'theory' field will be missing altogether. The content of this field, which is a MATLAB structure, will vary depending on which outputs you selected when running the analysis. Some fields apply only to force or unforced oscillator simulations.



The following fields describe the model and are generally read from the definition and don't change.

Field	Description
name	The name of the model.
plotting_timescale	Scaling factor when plotting time. Hours are assumed by default, but if the model uses minutes, this value will be 60. If the model uses seconds, this values will by 3600.
orbit_type	Either 'oscillator' or 'signal'.
dim	The number of dimensions (variables) the model has
vnames	A cell array of variable names.
parn	A cell array of parameter names. These are the symbolic names used in the model equations. For each model force, parmeters called *forecename*.dawn and *forcename*.dusk will be appended to this list.
parnames	A cell array of longer, more descriptive parameter names.

The following fields describe the time series

Field	Description
myfile	The location of the file where this structure is saved.
date	The date it was created.
solver	The name of the ODE solver used to solve the time series and whether it was optimized for a stiff or non-stiff problem.
par	A vector of parameters values used. They correspond to the parameter names in parn and parnames.
odesol	The solution returned by ODE solvers. This will contain several fields. These will differ for oscillator and signal models, as oscillators will generally use the boundary value solver last to refine the solution. See the Matlab documentation for more information.
sol	The solution in odesol but with evenly spaced timepoints, returned by `deval` in the case of MATLAB solvers or `interp1` in the case of Cvode. It contains the following fields: x, the timepoints; y, variable values; dy, variable derivatives.
force	The time series of the external force(s).
forced	For forced oscillators, this wil be non-zero, otherwise it will be zero.
forceparams	An array of structures, one for each external force. Each structure has the following fields: force, the name of the force in the model equations; name, the name of the force equation used; dawn and dusk values for this force.
per, tend	The length of the time series. For oscillators this field is called per, as it represents the period, for signal models it is called tend.
varnum	This applies only to unforced oscillators and it's the number of the variable that was used to define the starting point of the limit cycle.
peaks, troughs	These are cell arrays with one element for each variable. Each element is a scalar or vector representing the timepoints where the variable peaks or troughs are found.

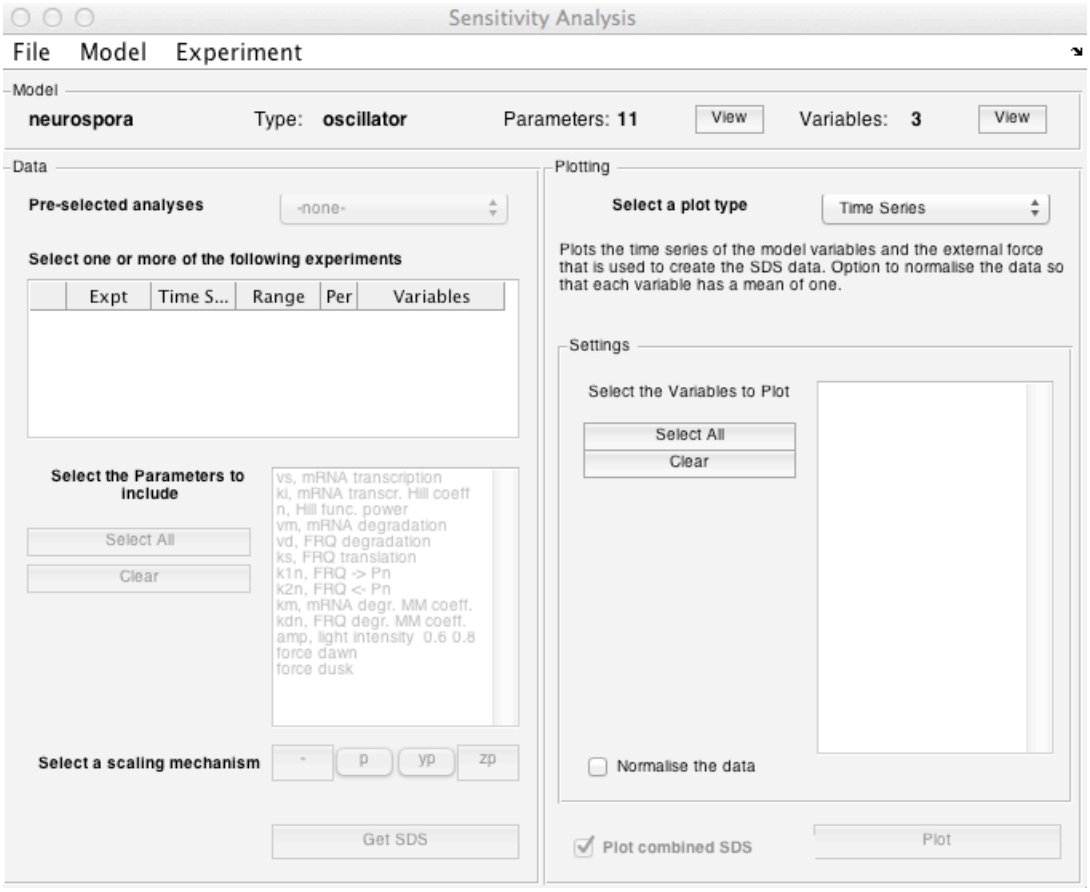
If the analysis has been performed the files 'theory' will be present. This is a structure and will contain some or all of the following fields

Field	Description
date	The date the analysis was run.
t	A vector of time points, copied from sol.x. These are the time points for those other fields which have a time dimension
dy0dpar	Applies only to oscillator models. $\frac{\partial y_0}{\partial \log k}$ in the standard coordinate system. If k=0, $\frac{\partial y_0}{\partial k}$ Is given instead. A parameter by variable matrix.
dxdm	Applies only to oscillator models. Curves under the integral for $\frac{\partial y_0}{\partial \log k}$ (if k=0, then $\frac{\partial y_0}{\partial k}$). A time by variable by parameter matrix.
dperdpar	Applies only to unforced oscillators. A vector of period derivatives with respect to parameter. Generally these values represent $\frac{\partial \tau}{\partial \log k}$ But where k = 0, they represent $\frac{\partial \tau}{\partial k}$ This is the data plotted in section 3.6.2.
irc	Applies only to unforced oscillators. A structure representing the infinitesimal response curves. It contains the following fields: data, the irc curves, a time by parameter matrix; integrals, a vector of integrals for each irc curve (parameter); maxAdvances and maxDelays, the maxima and minima respectively of each irc curve. This is the data plotted in section 3.6.3.
ircphi	Applies only to forced oscillators. The phase infinitesimal response curves. A cell array with one element for each model variable. Each element is itself a cell array, with one element for each peak found for that variable. Each element of this array, which corresponds to a specific peak, is a structure with the following fields: data, the phase irc curves, a time by parameter matrix; integrals, the total change to the phase when the parameter is perturbed permanently (.i.e., for the whole of the limit cycle); maxAdvances and maxDelays, the maxima and minima respectively of each irc curve; bs, the partial derivatives that correspond to parameter perturbations at the time of the peak. This last field is a structure containing a field y, which is a vector a values for each parameter, and a field t, which is the time of the peak. This is the data plotted in section 3.6.5. The integrals, maxAdvances abd maxDelays fields are used to fill the table, with the integral column being named 'Phase Chng'. The bs structures are the values plotted as single points. Generally, each cell array corresponding to a variable will contain only a single element as most variables are monophasic. However,

	for variables with multiple peaks there will be an element for each peak, plus one extra that contains the integrals, advances and delay values which are the largest across all variables. This structure is used for filling the table when selecting 'all peaks' in section 3.6.5 Note that the 'integrals' are not actually the areas under the curves, but these values plus the corresponding 'bs' points. As such they are identical to the $\frac{\partial \phi}{\partial \log k}$ values in dpkdpar for the corresponding peak.
ircphi_t	A vector of time points for the phase irc curves. This contains additional points compared to the model variable time series time point vector as points are required for the exact times of the peaks
periodic_dgs, nonper_dgs	These fields are the periodic and non-periodic solution derivatives respectively, time by variable by parameter matrices. This is the data plotted in section 3.6.1
dpkdpar, dtrdpar	These are the derivatives of the peaks (dpkdpar) and troughs (dtrdpar) with respect to log parameter, $\frac{\partial \phi}{\partial \log k}$. An exception is where k = 0, where they represent $\frac{\partial \phi}{\partial k}$. These fields are cell arrays, with one element for each model variable. Each element is a parameter by peak matrix, which are generally column vectors for monophasic variables. This is the data plotted in section 3.6.4.
dirty, dypk	These are the derivatives of the model variables with respect to log parameter at the times of their peaks (dypk) and troughs (dytr). These fields are cell arrays, with one element for each model variable. Each element is a parameter by peak matrix, which are generally column vectors for monophasic variables.
model_jac	This is the model's Jacobian matrix, the derivatives of the ODEs with respect to the model variables. This is produced by evaluating the file *modelname*_jac at each timepoint in the structure sol.

3.9. Sensitivity Analysis

Once the main PeTTSy program has completed its analysis of the time series data, it is now possible to further analyse the time series derivatives (dgs) by performing a singular value decomposition. This sensitivity analysis produces a large number of different outputs, so they are created and viewed by their own GUI. To launch this either run the `sagui` command from the Matlab prompt, or click the SVD… button on PeTTSy. If doing the latter, Sensitivity Analysis will open with the model currently selected in PeTTSy preselected.



You can select a different model using the Model menu item, and view the model variables and parameters as in the main PeTTSy window.

Note that models which do not yet have any time series files containing the dgs values will not appear in this GUI.

On this window the left panel contains controls used for defining your analysis, and the right panel contains controls used for plotting the results.

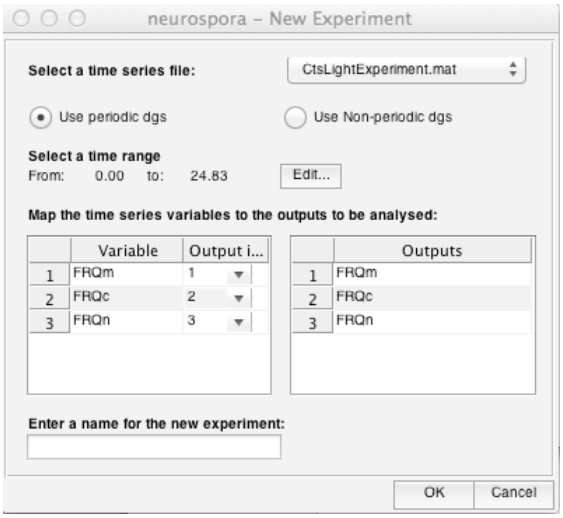
The first thing you must do for a model is to define an ‘experiment’. For this reason, when selecting a model that does not have any experiments defined, the New Experiment dialog box is brought up automatically. It can also be brought up later via the Experiment->New menu item. Multiple experiments can be combined to perform ‘experimental optimisation’, see section 3.9.3 below.

3.9.1. Defining an experiment

Sensitivity analysis uses the concept of an 'experiment'. An experiment represents part of a time series. It defines three things

1. The time points to use
2. Which variables to use, and whether to create composite variables by summing existing ones
3. For unforced oscillators, whether to use the periodic or non-periodic derivatives

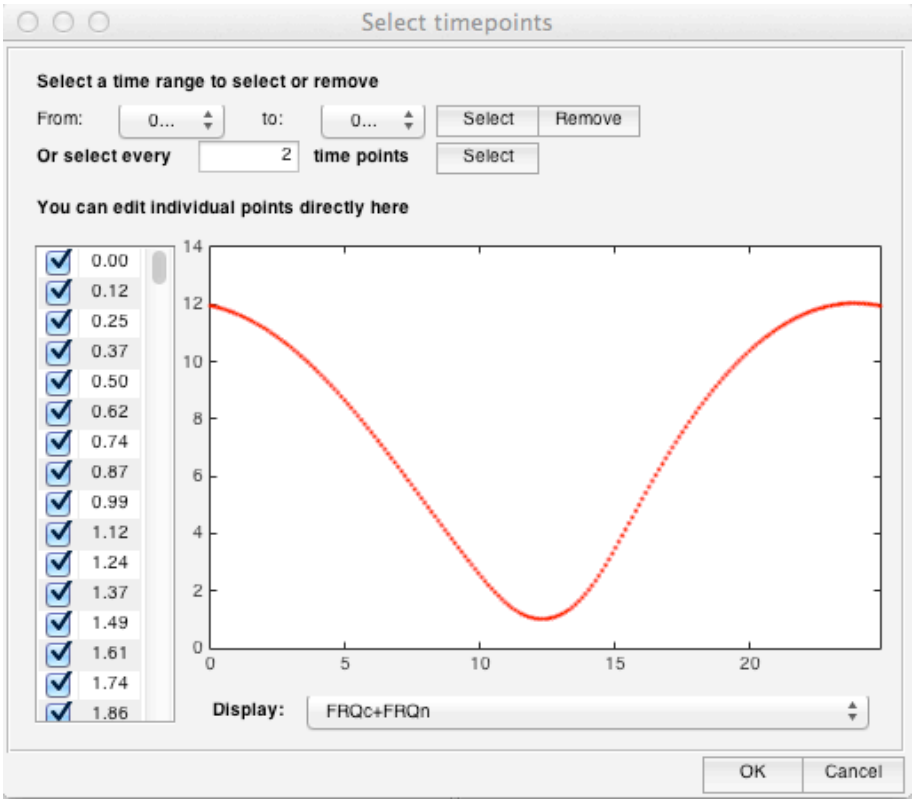
Before performing any analysis, you must create at least one experiment. Go to the Experiment->New menu item. This form then opens.



First, select the time series file upon which you want to base the experiment. Then select periodic or non-periodic dgs, if these buttons are enabled.

You can then define how the variables are to be treated. By default, all model variables will be used in the analysis. You can change this by changing the Output index value in the left hand grid. The right hand grid shows how the variables will be used in the analysis. To omit a variable, set its index to zero. It will then disappear from the outputs. To tell the program to use the sum of two or more variables in the analysis instead of using them all individually, set the index values of the required variables to the same non-zero value. This would be useful, for example, if a model treated cytoplasmic and nuclear pools of a particular protein separately, but you wanted to treat them as one pool in the analysis, perhaps because it would not be possible to measure the separate pools in an actual experiment. The analysis will use the variables/composite variables listed in the right hand grid, in the order in which they appear. Do not worry about blank lines in this grid, they are ignored.

You can also choose which time points to use from this form. Click the Edit button next to the displayed time range. A form like the one below will then appear. The plot shows one of the model variables from the time series. The drop down list below allows this to be changed. Only those selected on the previous form as outputs will appear in this list, including any composite variables, as seen here. The list to the left shows which time points will be used in the analysis, by default all will be selected. You can check or uncheck items in this list. When you do this, the colour of the corresponding points on the graph will switch between red (selected) and grey (not selected). You can also click on the points on the graph directly, which will toggle their selection in the list. There are three shortcuts to selecting or deselecting groups of points. Firstly you can define a range in the boxes at the top, then click to select or remove all of these. Secondly, you can opt to select every *n* points, starting at the first. Thirdly, you can right click on a point on the graph. A marker will then appear. Then when you left click again on another point, all points between this point and the marker will be reversed, ie selected if not already or removed if they were.



When you are done, click OK, then enter a name for the experiment back on the New Experiment form, and click OK again to save it.

Back on the main Sensitivity Analysis window, the new experiment will have appeared in the table of experiments towards the top of the left hand panel. This table contains columns displaying the new name, the name of the time series file it is based on, the first and last time points included, whether the derivatives are periodic or not, and the variables included. In the example below, two experiments have been created for the *neurospora* model (Leloup *et al*, 1999), one for a forced and one for an unforced limit cycle. In both cases the second and third variables (cytoplasmic and nuclear protein) have been summed.

	Expt	Time S...	Range		Per	Variables
☐	forced	forced	0.0	24.0	Y	FRQm,FRQc+FRQn
☐	unforced	ll	0.0	24.8	Y	FRQm,FRQc+FRQn

3.9.2. Running the analysis

Select an experiment by checking its check box. You can now choose which parameters to include in the analysis, and how to scale the derivatives before the analysis.

Analysis starts off with the solution derivatives described in section 3.6.1. These are processed according to the definition of the experiment, that is, unwanted time points and variables are removed, and any composite variables required are created by summing their components. Derivatives with respect to unwanted parameters are also removed. Then the resulting values are scaled according to the user's selection. The '-' option means no scaling, otherwise the following scaling is performed

- p scaling means that the derivatives with respect to the *j*th parameter are multiplied by the value of the *j*th parameter.

- yp scaling means that the derivative of the *i*th variable or composite variable is scaled by dividing by the amplitude of the same variable or composite variable in the original time series. Amplitude is calculated by considering only those time points included in the experiment. p scaling is then applied to the result.
- zp scaling means that the derivative of the *i*th variable or composite variable with respect to the *j*th parameter is scaled by dividing by the amplitude of the same derivative, considering only those timepoints included in the experiment. p scaling is then applied to the result.

To run the analysis, click Get SDS. A matrix, M, is then produced by stacking the variable time series so that the *j*th column represents the derivative of each variable time series with respect to the *j*th selected parameter. This is then divided by the square root of the number of selected time points. This removes the dependence of the result on this number. Singular value decomposition is then performed on this matrix, such that

$M = U\sigma V'$

Matrix U is the principal components of M, it will have one column for each selected parameter and a number of rows equal to the product of the number of selected time points and selected variables. Vector σ is the singular values, number equal to the number of selected parameters. Given that W is the inverse of V, a square matrix with dimension equal to the number of selected parameters, the following are also calculated:

- The sensitivity of the variables to the principal components, σU, or σUW
- The parameter sensitivity spectrum (strengths), that is, the effect of parameter perturbations on principal components, σW

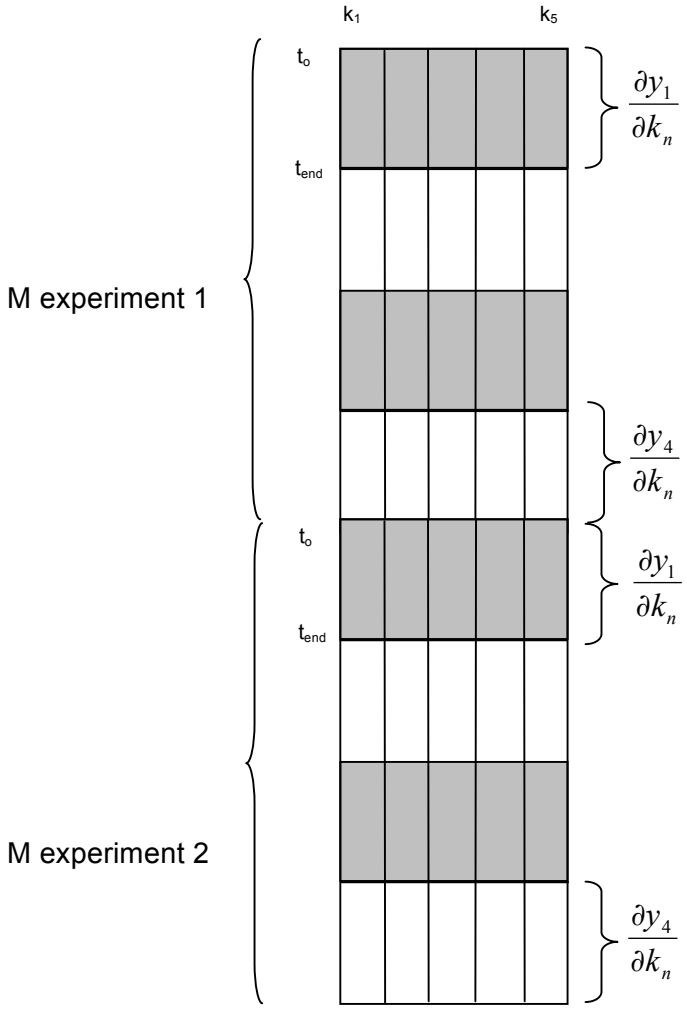
3.9.3. Experimental optimisation

It is possible to select more than one experiment in the table before performing SVD. This is known as Experimental Optimisation. When this is performed, each individual experiment is analysed as above. Then each individual matrix, M, is combined by stacking the columns as shown below, with an additional SVD being performed on the resultant matrix. Check the 'Plot Combined SDS' box to view the output of this analysis. In this case the matrix will be scaled by the square root of the total number of selected time points for all selected experiments.

Principal components corresponding to each experiment can then be extracted from analysis of the combined matrix. However, the number of singular values and size of the matrix V are determined by the number of columns (parameters) of this matrix, so these apply to the whole analysis, not individual experiments/variables. For this reason, some of the plot types, those that involve U, will produce two figures for each experiment, one for the experiment analysed on its own, and one for its component from the combined analysis (identified by 'combined SDS' in the title bar), whereas others will just produce one extra plot for the combined analysis.

When more than one experiment is selected, wherever variable names appear in the plots they will always be preceded by the name of the experiment.

In the example below, five parameters and two experiments have been selected. In each experiment, four variables or composite variables have been selected. This diagram shows how the derivatives are combined for analysis.



3.9.4. Saving analysis settings

Once you have decided upon the analysis you wish to perform, it is possible to save your current settings in order to recall them at a future time. Go to the File->Save Analysis Settings menu item and enter a name for the analysis. This will create a file which contains the following settings: selected experiments, selected parameters, scaling mechanism. If you select the name of this analysis in the 'Pre-selected analyses' list at a later time, these selections will be restored.

3.9.5. Plotting the result

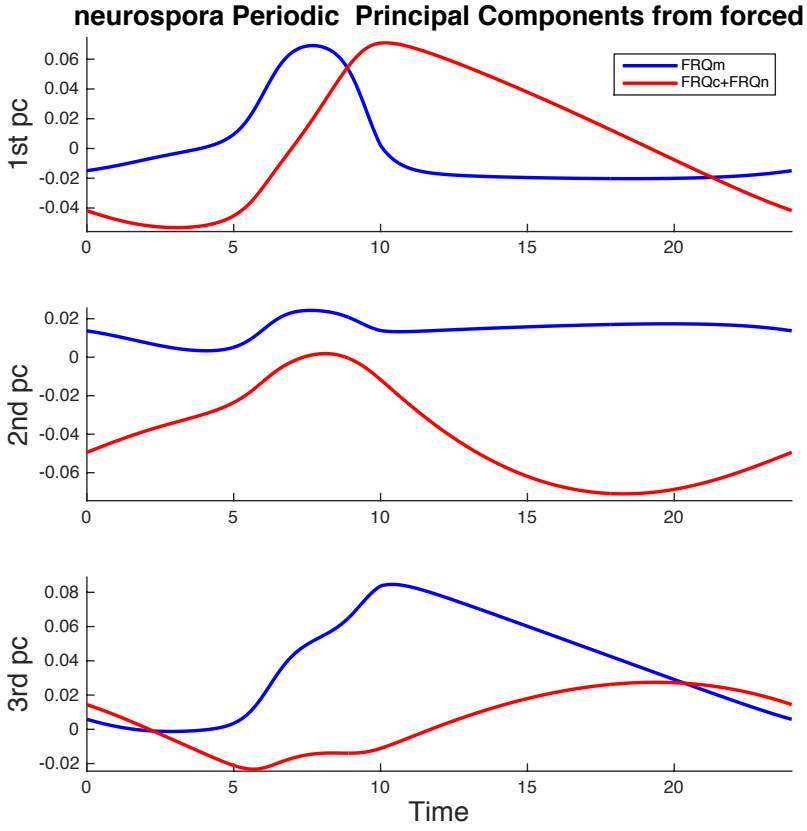
The right panel of the Sensitivity Analysis window contains all the controls for generating plots. Select the required plot type from the list at the top. The settings for the selected type then appear in the Settings panel below. The plot button becomes activated when the SVD analysis has been performed. The first two plot types, 'Time Series' and 'Solution Derivatives', are similar to the corresponding plots in PeTTSy, except that here they reflect the experiment defined. That is, they will show only the selected time points and variables and composite variables. In addition, unless scaling is set to '-', the solution derivatives are divided by the square root of the number of time points. This is to remove the dependence of the output on time series length. A separate figure will be created for each selected experiment. The following sections describe the additional plot types available.

3.9.6. Principal components

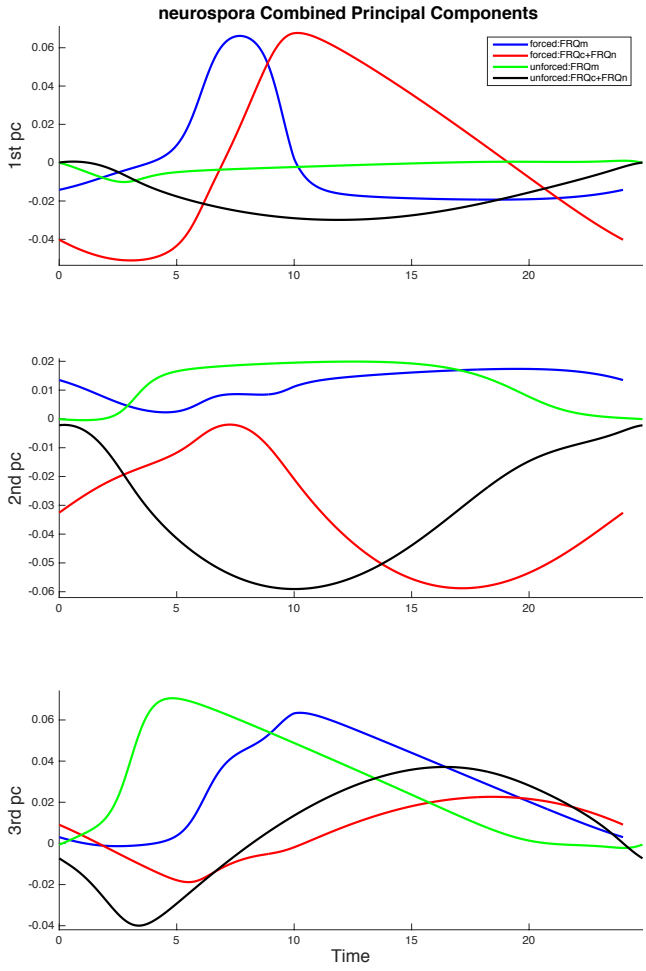
This plot displays the principal components of the analysis, U, versus time. The user can choose the following

- Which variable-pc combinations to include. Variables available are those defined by the selected experiment(s). When performing experimental optimisation, all variables from all selected experiments appear in the variable list, preceded by the name of the experiment.
- Whether to divide the pcs by the time series. Checking this box will divide each pc by the corresponding variable value at corresponding time points.
- Whether to plot the actual pc values, (or values divided by the time series if selected), or to plot the absolute values, or the log absolute values.

A figure is produced for each selected experiment. Each figure contains a set of axes for each selected principal component. Each set of axes shows the pc of all selected variables for that experiment. If the 'Plot Combined SDS' box is checked when more than one experiment is selected, then there is an additional figure. In this case each set of axes shows the pc of all selected variables across all experiments from the matrix U produced by performing SVD upon the combined matrix. The example below shows the first three principal components for two variables of the model *neurospora*, (Leloup *et al*, 1999), from a forced limit cycle.



The example on the next page shows the same experiment, combined with an experiment from an unforced limit cycle. This is the plot from the analysis of the combined matrix.



3.9.7. Sensitivity heat map

This plot displays a series of heat maps showing the sensitivity of the variables to the principal components, versus time. Sensitivity is defined as either the product of the *j*th singular value and the *j*th PC, $\sigma_j U_j$; or as the absolute value of $\sigma_j U_j$ multiplied by the largest absolute value of the *j*th row of W. This second definition if also referred to by the GUI as f(i,m).

Sensitivity plots are very useful for detecting variables which are sensitive to model perturbations around their peak and trough times, as these can be overlaid on the heat maps. This is useful to know if you wish to change a parameter to alter the phase of a particular model variable.

There are a number of custom controls available. The user must select the following

- The definition of sensitivity

  $\sigma_j U_j$ or $|\sigma_j U_j| \times \max|W_j|$

- Whether to plot the sensitivity itself, or the derivative. If derivative is checked, one of the following will be plotted

  $\frac{\partial \sigma_j U_j}{\partial t}$ or $\left|\frac{\partial \sigma_j U_j}{\partial t}\right| \times \max|W_j|$

- Which principal components to plot, ie which columns of U

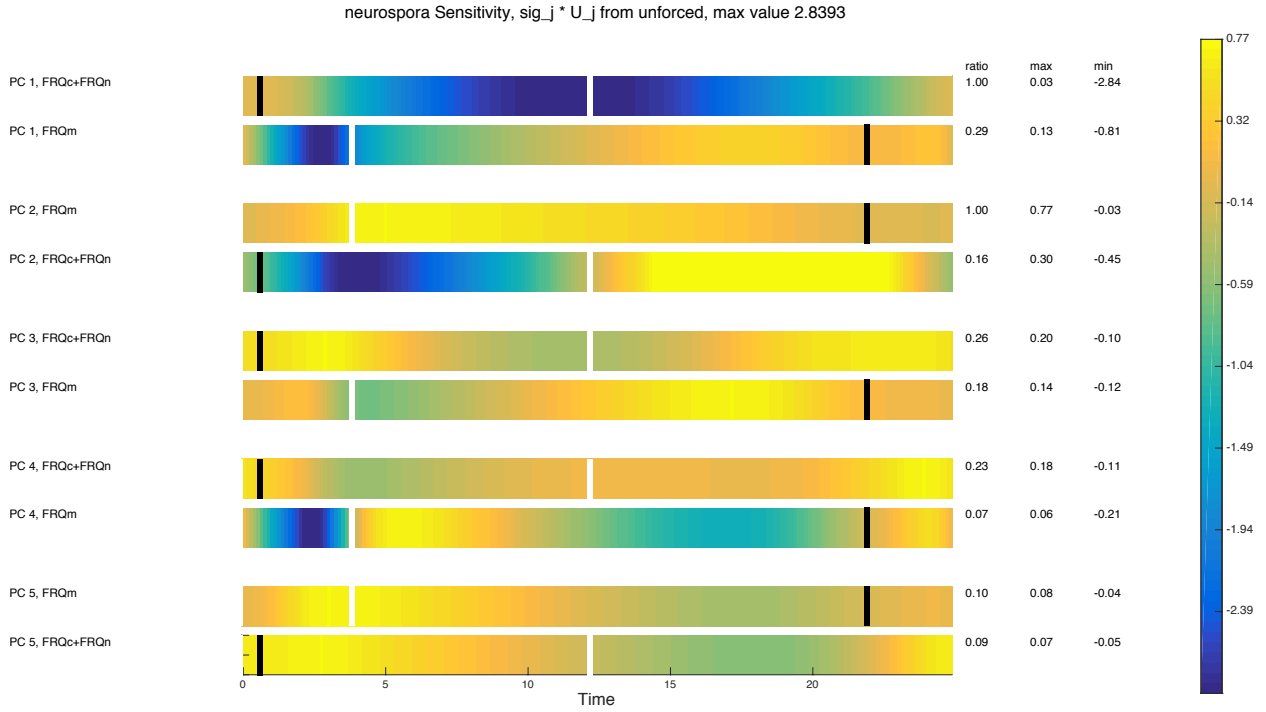
- Which variables to include. Options are to plot the sensitivity of all model variables from the selected experiment(s) to the selected principal components, or to only include those variable-selected pc combinations where the maximum absolute sensitivity value exceeds the entered proportion of the global absolute maximum of the sensitivity function across all pcs and variables for that experiment.
- Whether or not to sort the variables. Heat maps are grouped by principal component. Checking the sort box will result in variables being sorted within each pc into descending order according to the maximum absolute sensitivity value. If Scaling is set to 'by maxima' then sorting is slightly different. The maximum absolute value is divided by the global maximum, if it is itself a maximum value, or the absolute global minimum, if it is itself a minimum value, in order to produce a value to be used in the sort.
- Whether to highlight the most important regions (highest sensitivity) of each heat map. Checking this box will add a magenta coloured bar to the lower half of each heat map to indicate areas where the underlying absolute values exceed the specified proportion of the absolute global maximum of the sensitivity function across all pcs and variables for that experiment. Moving the mouse pointer over one of these bars will produce a tooltip string displaying the threshold value.
- Whether or not to superimpose the peak and trough times of the corresponding variable time series onto each heat map. If this box is checked then a white bar is added to represent the peak time of the variable and a black bar is added to represent the trough time. These can be useful when searching for a variable whose peak/trough time is very sensitive to a particular principal component
- How to scale the color range of the heat maps. There are four options available.
    - 'by maxima' scales each heat map so that the larger of its maxima or minima (absolute value) is equal to the global maxima or minima for that experiment, and this is shown as the highest or lowest possible colour. In this case the values on the colour bar span the global maxima and global minima, and do not apply to any one heat map, except those which include the global maxima and minima.
    - 'by amplitude' scales each heat map so that its absolute maximum equals the global absolute maximum for that experiment. The colour bar represents ± the global absolute maximum, and the values do not apply to any one heat map.
    - 'all equally' scales all heat maps by the same factor. The colour bar represents ± the global absolute maximum for that experiment, and its values apply to all heat maps.
    - 'to fill colour space' scales each heat map to fill the colour space. The colour bar represents ± the global absolute maximum for that experiment, and the values do not apply to any one heat map.

In the example below the user has elected to consider the first five principal components, and to show two model variables, which are grouped as a pair for each pc. Areas greater than 75% of the global maximum are highlighted.

The variable each plot represents is named to the left of plot. To the right of the plots are displayed the maximum and minimum values of the underlying data and the 'ratio'. Ratio is a scaling factor, the maximum absolute value of the data divided by the global maximum absolute value for that experiment. An exception is when Scaling is set to 'by maxima'. In this case, if the maximum absolute value of the plot is positive, ratio is this value divided by the global maximum. If the maximum absolute value of the plot is negative, ratio is this value divided by the global minimum. If any of these labels become too small to read, then moving the mouse pointer over them will result in a tooltip string displaying their value.

When performing experimental optimisation a separate figure is produced for each experiment. If the Plot Combined SDS box is checked when more than one experiment is selected, then the output of the SVD performed on the combined matrix is used to produce a second figure for each

experiment, this time representing the pc component corresponding to the experiment from the combined output.



Clicking on any of the heat maps will display the underlying data, with the time point that was clicked on marked.

3.9.8. Time series with sensitivity

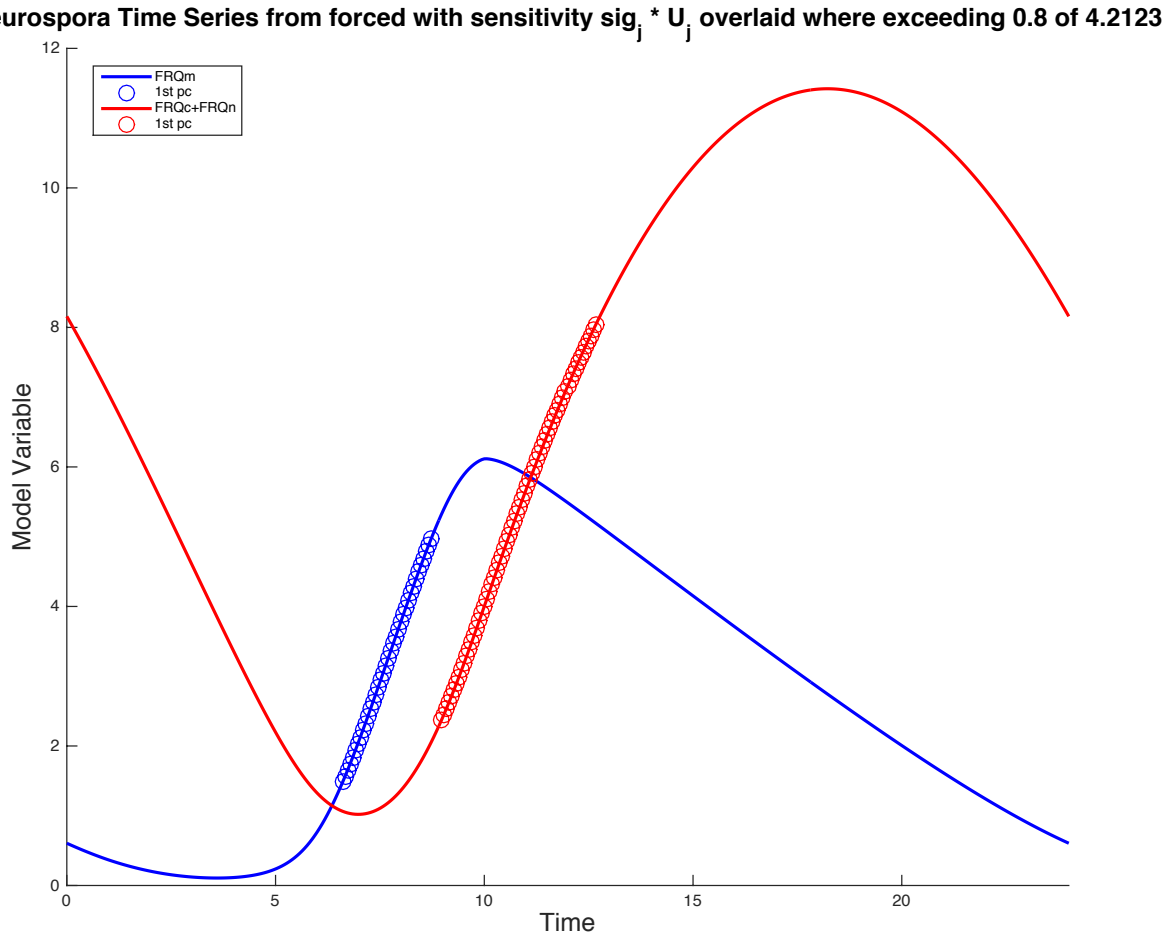
This plot type could be said to be the inverse of the previous type. Rather than plotting sensitivity, and then overlaying variable time series peaks and troughs, here it is the time series which is plotted, and regions of high sensitivity to selected principal components are highlighted on the plots. Sensitivity is defined in two ways, as above. Again this plot is very useful for detecting variables which are sensitive to model perturbations around their peak and trough times. This is useful to know if you wish to alter the phase of a particular model variable.

There are a number of custom controls available. The user must select the following

- The definition of sensitivity, as above.
- Whether to consider the sensitivity itself, or the derivative, as above.
- Which principal components to consider when looking for regions to overlay, ie which columns of U.
- The threshold for overlaying sensitivity, as a percentage of the global maximum of the selected sensitivity function across all pcs and variables for that experiment.
- Whether to plot all the model variables for each experiment, or only those which will have part of their time series highlighted, ie those which have a maximum sensitivity to any of the selected principal components above the threshold.
- Whether to normalise the time series so that each has a mean of one.

In the example below, regions of the variable time series where sensitivity to any pc exceeds 80% of the global maximum sensitivity across all pcs and variables have been highlighted. Circles are used to highlight areas where the sensitivity to the first principal component exceeds the threshold.

Other markers would be used for other pcs, though in this case none apply.



When performing experimental optimisation a separate figure is produced for each experiment. If the Plot Combined SDS box is checked when more than one experiment is selected, then the output of the SVD performed on the combined matrix is used to produce a second figure for each experiment, this time representing the pc component corresponding to the experiment from the combined output.

3.9.9. Singular spectrum plot

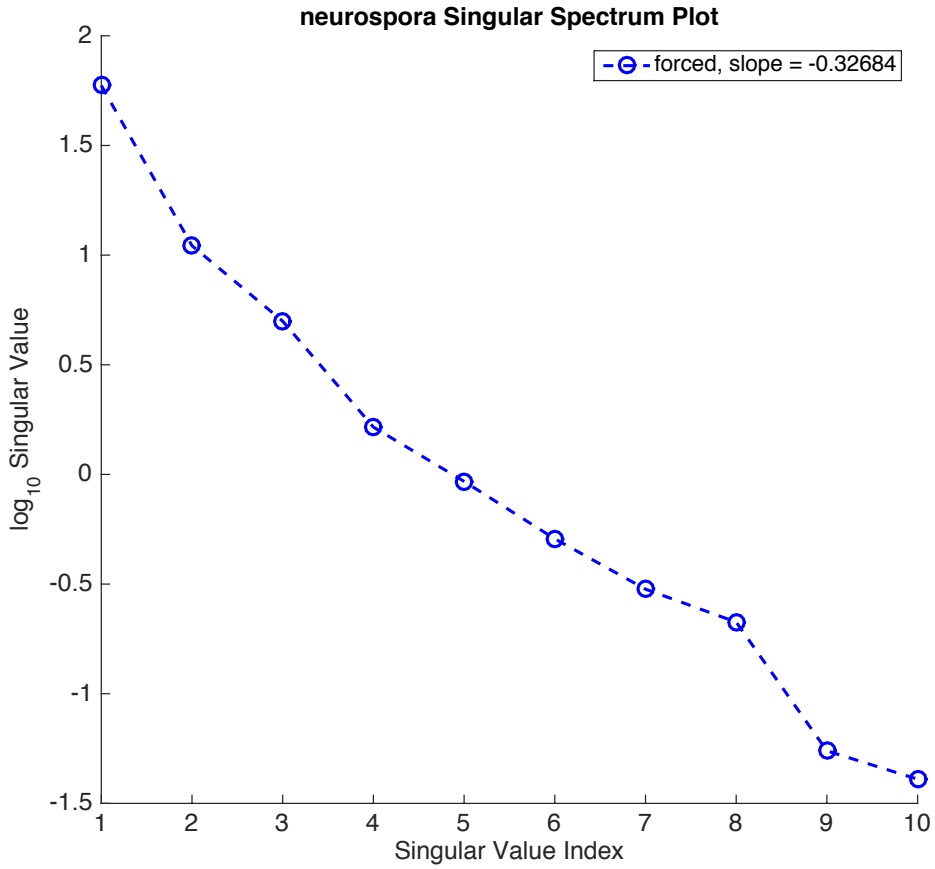
This plot displays the $\log_{10}$ of the singular values, σ, against value index. The user can choose the following.

- How many values to plot. The first two must always be plotted, plus any number of additional non-zero values.
- Whether to normalise the values before taking the log, so that the first has a value of one.

When performing experimental optimization, the singular values for all selected experiments are added to the same axes. If the Plot Combined SDS box is checked, the singular values for the combined SVD are also added.

Here the user has opted to plot the first ten singular values. The legend displays the slope of the

straight line fitted to the data by the least squares method.



3.9.10. Singular value analysis plot

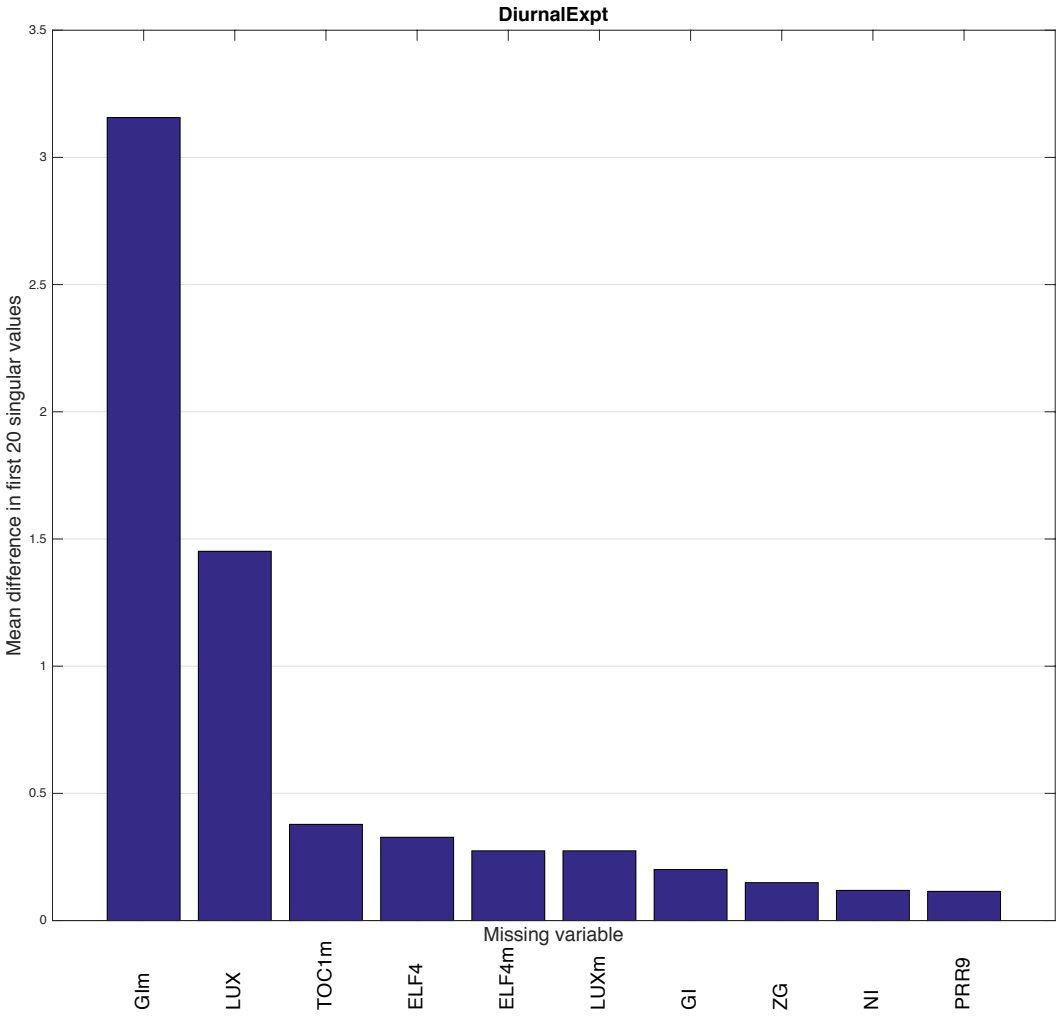
This plot type is designed to determine which of the model variables has the largest effect on the singular values. After the initial SVD is performed on Matrix M, the rows corresponding to the time series of the first selected model variable or composite variable are removed from M and another SVD is performed to generate singular values for the reduced matrix. These rows are then put back and the rows corresponding to the second model variable or composite variable are removed and another SVD performed. This continues until all selected variables have been removed in turn and a set of singular values generated for the matrix excluding each variable. These singular values are then compared to the singular values for the full matrix, ie when all selected variables are included. The mean absolute difference between the full set and each of the other sets in turn is calculated by comparing corresponding singular values. This produces a number for each variable which is a measure of the effect of removing that variable on the singular values. These are then plotted as a bar chart.

You can select how many singular values to consider when performing the comparison. The first to the *n*th will be considered. Choose a value for *n* from the dropdown list. In addition, you can opt not to plot all the variables included in the analysis, but only the most important ones, ie those producing the largest effect on the singular values when removed. Check the box to sort the variables by effect and select how many to plot, starting with the most important, from the drop down list. This list will contain values up to the total number of variables in all selected experiments, ie the total number in the combined matrix, which will be greater than any individual experiment if more than one is selected. If you select a number greater than the number of variables in an experiment, then all will be plotted for that experiment.

When performing experimental optimization with more than one experiment selected, a separate bar chart is produced for each. If the Plot Combined SDS box is checked, an additional plot shows the effect of removing each variable in turn from the combined matrix. All variables from all experiments are shown, unless the user opted to restrict the number.

Note that only one variable is removed for each SVD. No attempt is made to remove corresponding variables from each experimental component of the combined matrix as different experiments may contain different output variables and composite variables.

The example shown below is taken from the *PlantClock* model of the *Arabidopsis* circadian clock (Pokhilko *et al.* 2012), running as a forced oscillator with *no scaling*. Here it can clearly be seen that variable GIm has the most important effect on the singular values.



3.9.11. Parameter sensitivity spectrum (strength) plots

The parameter sensitivity spectrum, or strength values, are calculated by multiplying the jth row of W by the jth singular value, $\sigma_j W_j$. This represents the effect of perturbing each parameter on the jth principal component. Strengths are very useful when you have identified a particular principal component to which the time series in very sensitive. They then allow parameters to be identified which can alter this pc in the desired way.

There are a number of custom controls available. The user must select the following

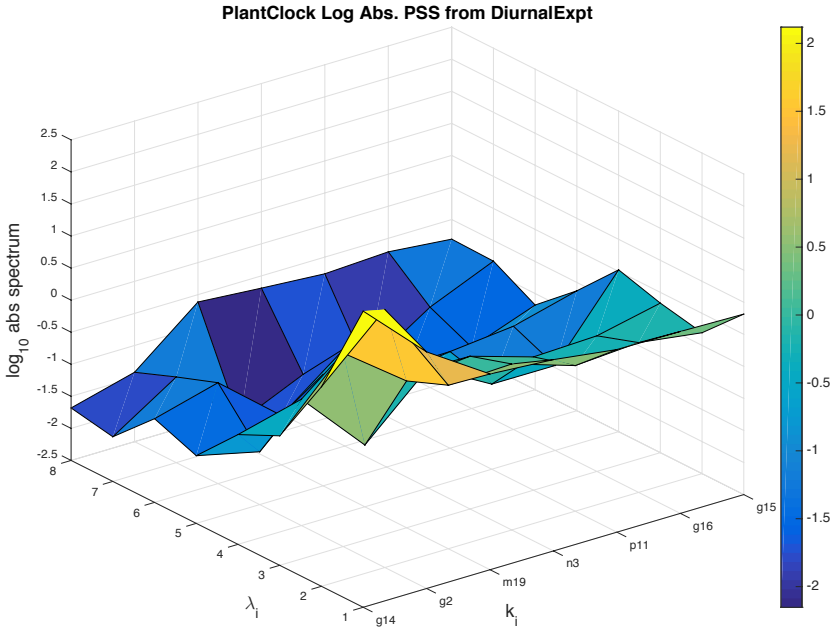
- The type of plot. The options are as follows
    - Surface plot. This plots the strengths matrix on 3D axes, with the x axis representing parameter, the y axis principal component (labelled λ), and the z axis the strength value. Colour is also used to represent strength value.
    - 3D bar chart. This plots the data in the same way, but for clarity the data is split along the parameter axis and two plots produced.
    - 2D bar chart. This produces a series of 2D axes of strength versus parameter in a single figure window, with each representing one of the selected principal components. If the Group plots box is checked, then these are grouped into a single set of axes, with colour representing principal component.
- Which parameters to include. These can be selected from the list, or the most important (those with the largest strength values) can be automatically selected. This involves sorting the columns of the strength matrix according to either the strength for the first pc (first row) in each, or according to the largest strength value in each. The first *n* are then plotted.
- Which principal components to include. These are selected from the list.
- Whether to use the raw strength values, the absolute values, or the $\log_{10}$ absolute values.

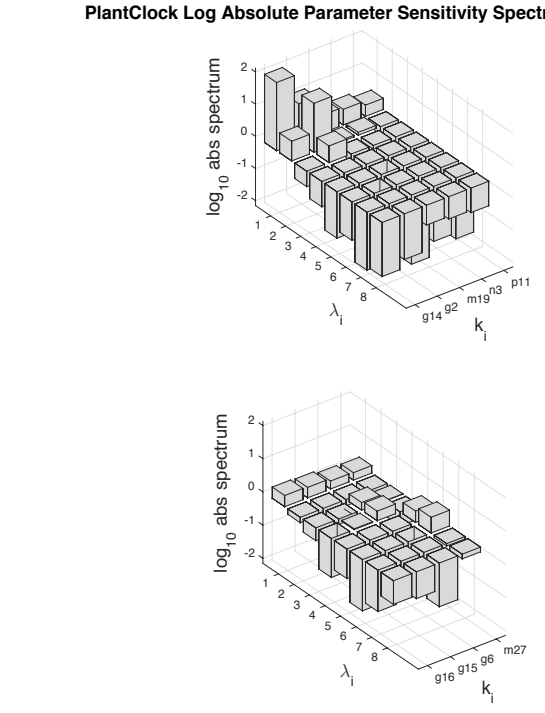
The following only apply when absolute or $\log_{10}$ absolute values are chosen.

- Whether to normalise the data. Checking this box will result in the data being shifted down the z axis so that the global maximum value is zero.
- The 'sea level' value. Checking this will results in all strength values higher than the specified value being plotted as a height above this value. All other values are set to the sea level.

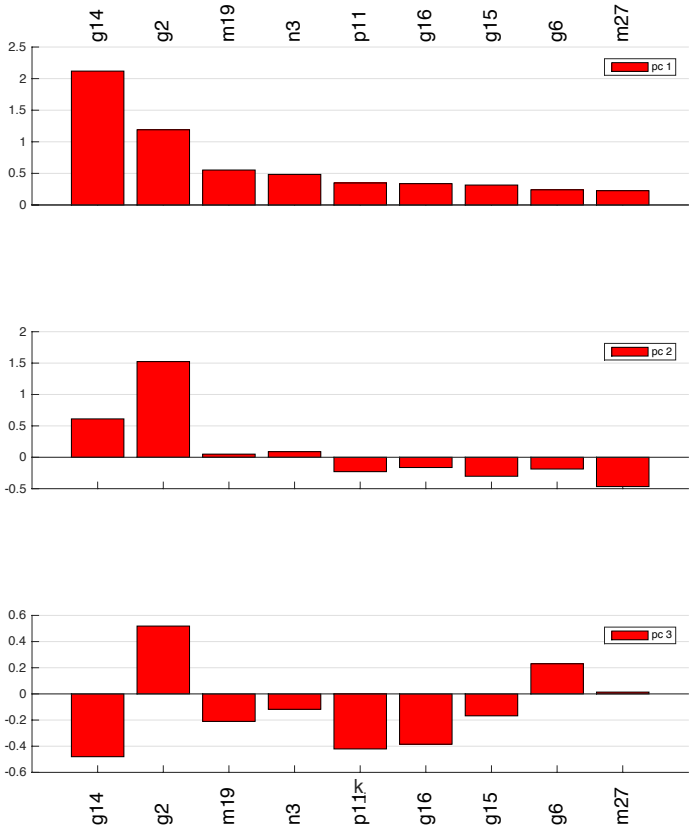
When performing experimental optimisation a separate figure is produced for each experiment. If the Plot Combined SDS box is checked when more than one experiment is selected, then the strength values of the SVD performed on the combined matrix are plotted in an additional figure.

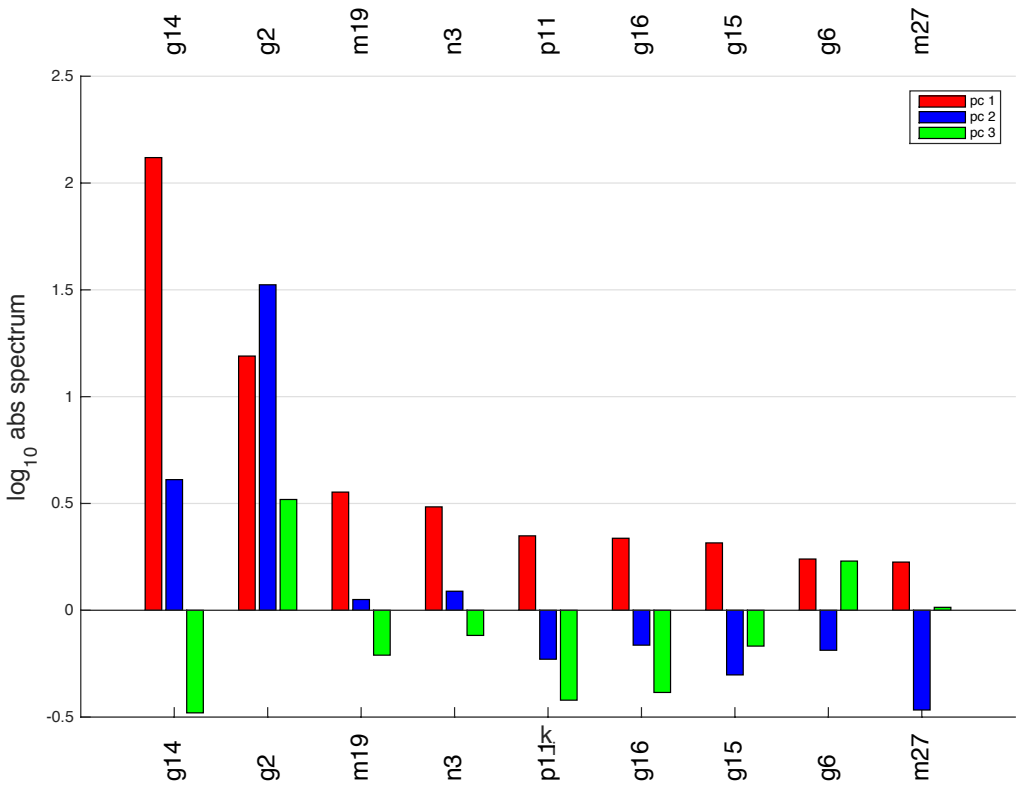
Here $\log_{10}$ strength values, with parameters sorted according to the value of the first principal component are plotted in the two 3D formats.





Here the same data is shown as 2D bar charts. The first three principal components are plotted as a series of 2-D bar charts, one for each PC, and then a grouped bar chart with a data series for each pc.


g14
g2
m19
n3
p11
g16
g15
g6
m27
pc 1
pc 2
pc 3
p11k


log10 abs spectrum
g14
g2
m19
n3
p11
g16
g15
g6
m27
pc 1
pc 2
pc 3
p11k

3.9.12. Amplitude-period scatter plot

This plot allows the user to view the relative effects of the principal components (λ) and parameters (k) on the phase (θ) and amplitude (Amp) of the model variables. The derivative of the phase of a selected variable with respect to a principal component or parameter is plotted against the derivative of the variable amplitude with respect to the same.

The derivatives are calculated as follows, with m representing variable index, i representing pc or parameter index, and t representing time point.

$$\frac{dAmp_m}{d\lambda_i} = \frac{\sum_{t=0}^{t=N} U_{mit} y_{mt}}{\sum_{t=0}^{t=N} y_{mt} y_{mt}} \qquad \frac{dAmp_m}{dk_i} = \frac{\sum_{t=0}^{t=N} \frac{\partial y_{mt}}{\partial k_i} y_{mt}}{\sum_{t=0}^{t=N} y_{mt} y_{mt}}$$

for all time series

$$\frac{d\theta_m}{d\lambda_i} = \frac{\sum_{t=0}^{t=N} U_{mit} \frac{dy_{mt}}{dt_t}}{\sum_{t=0}^{t=N} \frac{dy_{mt}}{dt_t} \frac{dy_{mt}}{dt_t}} \qquad \frac{d\theta_m}{dk_i} = \frac{\sum_{t=0}^{t=N} \frac{\partial y_{mt}}{\partial k_i} \frac{dy_{mt}}{dt_t}}{\sum_{t=0}^{t=N} \frac{dy_{mt}}{dt_t} \frac{dy_{mt}}{dt_t}}$$

for periodic values, ie forced oscillators and unforced with periodic principal components selected

$$\frac{d\theta_m}{d\lambda_i} = \frac{\sum_{t=0}^{t=N} U_{mit} \frac{dy_{mt}}{dt_t} T}{\sum_{t=0}^{t=N} \frac{dy_{mt}}{dt_t} T \frac{dy_{mt}}{dt_t} T} \qquad \frac{d\theta_m}{dk_i} = \frac{\sum_{t=0}^{t=N} \frac{\partial y_{mt}}{\partial k_i} \frac{dy_{mt}}{dt_t} T}{\sum_{t=0}^{t=N} \frac{dy_{mt}}{dt_t} T \frac{dy_{mt}}{dt_t} T}$$

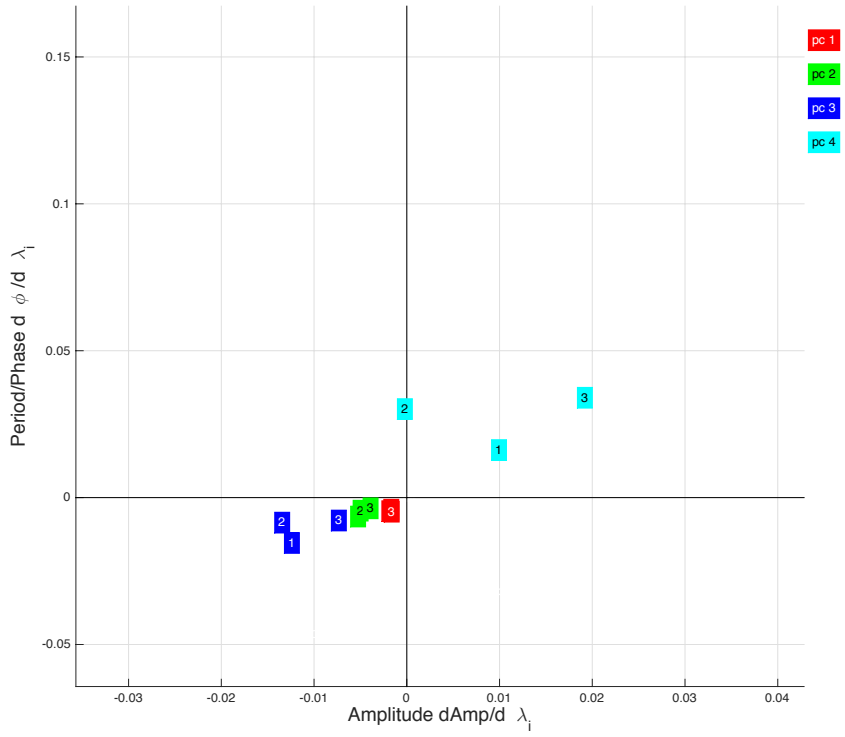
for non-periodic values, ie signal solutions and unforced oscillators with non-periodic principal components selected

These equations show the derivatives of amplitude or phase of variable m with respect to pc or parameter i at time t, where t is zero to N. Amplitude derivatives are equal to the dot product of the corresponding pc or derivative with respect to parameter, and the variable value. This is divided by the dot product of the variable value squared. Phase derivatives are equal to the dot product of the corresponding pc or derivative with respect to parameter, and the variable derivative with respect to time. This is divided by the dot product of the variable derivative with respect to time squared. For non-periodic phase derivatives, variable derivatives are multiplied by time, T.

The custom controls for this type of plot are as follows:

- Select whether to show derivatives with respect to principal component or parameter
- Select which variables to include. Variables available are those defined by the selected experiment(s). When performing experimental optimisation, all variables from all selected experiments appear in the variable list, preceded by the name of the experiment.
- Select which pc or parameters to include.

The values to be plotted are displayed as numbers, representing the variable. The numbers correspond to the index values of the selected variables. They are colour coded to represent principal component or parameter. For example, in the plot below (for the PlantClock, under ‘photo’ forcing, with no scaling) the user has opted to plot derivatives with respect to the first four principal components. These are colour-coded red, green, blue, and cyan respectively. The first three variables are plotted. The x-axis represents amplitude derivative and the y-axis phase derivative. The plot illustrates that the principal components affect the amplitude of the selected variables much more than their phase.



When performing experimental optimisation a separate figure is produced for each experiment. The combined plot is only available if derivative with respect to principal component is selected. If this is the case, and the Plot Combined SDS box is checked and more than one experiment is selected, the pcs of the combined matrix for each experiment are plotted on a second figure for that experiment.

3.9.13. Composite plot

The composite plot combines sensitivity and parameter sensitivity spectrum plots for a selected variable and principal component combination. It allows the user to view the sensitivity of the variable to the pc, and the sensitivity of the pc to the model parameters. The plot includes a heat map of variable sensitivity, a line graph of the same data, a heatmap of the sensitivity derivative, the variable time series, and a bar chart showing the parameter sensitivity (strength) values for the selected pc. Sensitivity and its derivative are calculated as described for the sensitivity plot (section 3.9.7).

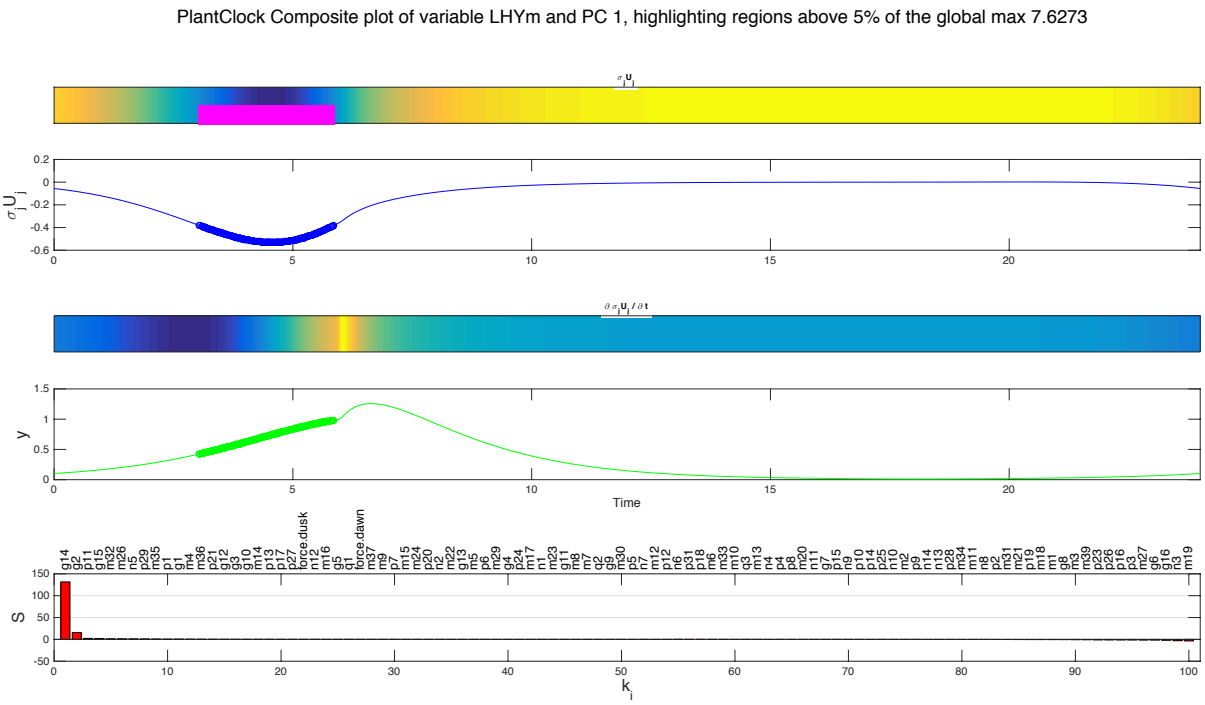
There are a number of custom controls.

- The definition of sensitivity, as described for sensitivity plot
- The principal component, the sensitivity to which will be plotted.
- The model variable to plot. Variables available are those defined by the selected experiment(s). When performing experimental optimisation, all variables from all selected experiments appear in the variable list, preceded by the name of the experiment.
- Whether to highlight the most important regions (highest sensitivity) of the sensitivity heat map and line plots. Checking this box will add a magenta bar to the lower half of the heat map to indicate areas where the underlying values exceed the specified proportion of the absolute global maximum of the sensitivity function across all pcs and variables for that

experiment. Moving the mouse pointer over one of these bars will produce a tooltip string displaying the threshold value. Highlighting will be added to the corresponding region of the line plots.

- For the parameter sensitivity spectrum (strength) values, whether to plot the raw values, absolute values, or the $\log_{10}$ absolute values; normalise them; or apply a 'sea level'. These options are as described for parameter sensitivity spectrum plots in section 3.9.11.
- Whether to sort these parameters by size or not.

Here the user has plotted sensitivity of variable *LHYm* to the first principal component. Regions where the underlying values exceed 5% of the global maximum sensitivity value are highlighted on both plots. The sensitivity derivative is then shown as a heat map. The time series of *LHYm* is then plotted below this, with the same regions highlighted. Finally, the absolute parameter sensitivity values, sorted by size, are plotted for the first principal component.



Here we can see that *LHYm* is most sensitive to the first principal component close to the trough in its time series, and this pc is best manipulated by perturbing parameter *g14*.

When performing experimental optimisation two figures are produced when the Plot Combined SDS box is checked. These are the selected variable/pc combination from the experiment that the variable was taken from analysed alone, and the same data taken from the output of the combined analysis.

3.9.14. Exporting data

Once you have performed the SVD analysis, as well as plotting the results you can also export it to the Matlab workspace for further analysis. To do this, go to the File->Export data menu item and enter a name for the variable to create. This name will be automatically modified if a variable with the same name already exists in the workspace so this will not be overwritten. The new variable will be a structure with the following fields

The following fields describe the overall analysis.

Field	**Description**
mymodel	The name of the model
mymodeltype	Either 'oscillator' or 'signal'.
which_pars	The indices of the parameters that were selected
parn	A cell array of selected parameter names. These are the symbolic names used in the model equations. For each model force, parameters called *forecename* dawn and *forcename* dusk will be appended to this list.
parnames	A cell array of longer, more descriptive parameter names.
scaling	The scaling selected; '-', 'p', 'yp' or 'zp'.

The following fields describe the selected experiments which are input for the analysis. They are generally cell arrays, with one element for each experiment.

Field	**Description**
exptnames	The names of the selected experiments.
periodic	A vector of non-zero and zero values to indicate whether each experiment uses the periodic or non-periodic derivatives respectively
lc	The time series from which the derivatives were calculated. Each element is a matrix with row representing time points and column the variables and composite variables defined by the experiment.
vector_field	The derivatives of the above with respect to time.
t	The time points corresponding to the above fields.
force	The time series of the external force(s). Each element is a matrix with a column for each force.
forcename	Each element is a cell array of the name(s) of the force(s) corresponding to the columns of the force matrix.
par	The parameter values used to create the time series in lc. Each element is a vector of all the parameter values, not just the ones selected for SVD analysis in 'which_pars'. This field is equivalent to the par field output by PeTTSy, described in section 3.8
per/tend	The length of the time series. For oscillators this field is called per, as it represents the period, for signal models it is called tend. This is equivalent to the per/tend field output by PeTTSy, described in section 3.8. It does not depend on the time points selected for SVD analysis.
dim	A vector of the number of variables or composite variables in each experiment.

vnames	Each element is a cell array of corresponding variable names.
main_deriv	The solution derivatives with respect to parameter, dgs, as defined by the experiment. Each element is a matrix with column corresponding to a selected parameter, and variable time series stacked in rows. The values are divided by the square root of the number of selected time points for that experiment. This is the matrix upon which SVD analysis is performed. **Note**, although the matrix is always scaled by the number of time points before SVD analysis, if scaling is set to no scaling by parameter or amplitude ('-'), then the main_deriv matrix is output before this time point scaling is applied. This matrix is plotted as described in section 3.9.5.

The following fields are the output of the SVD analysis. They are generally cell arrays, with one element for each experiment

U_all	The principal components, each element is a matrix with one column for each selected parameter and a number of rows equal to the product of the number of selected time points and selected variables for that experiment.
V_all	The square matrix V output by SVD, as described in section 3.9.2. Each has size equal to the number of selected parameters.
spec_all	The singular values, each element is a vector of length equal to the number of selected parameters
slope_spec_all	The slope of a linear trend line fitted to $\log_{10}$ singular values. If more than ten parameters are selected for an experiment, only the first ten singular values are considered here.
strengths	The parameter sensitivity spectrum, as described in section 3.9.11. Each element is a square matrix with size equal to the number of selected parameters.
sigma_missing_var	This is the data plotted in section 3.9.10. Each element is a cell array with one element for each variable or composite variable in that experiment, with the *j*th element being the singular values generated by SVD analysis with the *j*th variable removed.

The following fields are the result of the combined SVD analysis, and will be empty unless more than one experiment was selected.

bigU	The principal components of the combined matrix. There will be one column for each selected parameter and a number of rows equal to the product of the number of selected time points and selected variables summed for all selected experiments.
bigV	The square matrix V output by SVD of the combined matrix, with size equal to the number of selected parameters.
bigspec	The singular values of the combined matrix.
bigslope_spec	The slope of these values, as described above.
bigstrengths	The parameter sensitivity spectrum of the combined matrix, a square

	matrix with size equal to the number of selected parameters.
bigsigma_missing_var	This is the data plotted in section 3.9.10. A cell array with one element for each variable or composite variable in all selected experiments. Each element is a vector of singular values with the corresponding variable removed. Variables appear in the order defined by the exptnames and vnames fields.

4. References for Described Models

A. Pokhilko, A.P. Fernàndez, K.D. Edwards, M.M. Southern, K.J. Halliday, A.J. Millar (2012). The clock gene circuit in Arabidopsis includes a repressilator with additional feedback loops. Molecular Systems Biology **8**: 547 doi: 10.1038/msb.2012.6.

J.-C. Leloup, D. Gonze & A. Goldbeter (1999). Limit cycle models for circadian rhythms based on transcriptional regulation in *Drosophila* and *Neurospora*. *Journal of Biological Rhythms* 1**4(6):**433-448 doi: 10.1177/074873099129000948

5. Appendix

5.1 *Adjusting Matlab's screen resolution to display the GUI correctly*

It is anticipated that our GUIs will be used on a wide variety of display sizes. To ensure that they are always displayed at an appropriate size it might be necessary to tell Matlab to alter what it thinks is your screen resolution. For example, the main PeTTSy gui is intended to be displayed 24cm wide, excluding the Tip window if present. If it displays too small, for example 16 cm, run the following command.

```
ppi = get(0, 'screenpixelsperinch');
```

This is what Matlab thinks your screen resolution is, in pixels per inch, as this is what your graphics card has told it it is. Alter this value according to this formula, substituting 16 for whatever the real width is on your screen.

```
ppi = ppi * 24/16;
```

Now tell Matlab the new resolution.

```
set(0, 'screenpixelsperinch', ppi);
```

As an example, the author found that on a 13 inch laptop with a true screen resolution of 227 ppi, Matlab initially had a resolution set at 70 ppi. Changing this to 90 displayed the guis at the correct, larger size.

See www.mathworks.com/matlabcentral/newsreader/view_thread/6836 for an explanation.

5.2 *Choosing an ODE Solver*

When selecting an ODE solver the main issue to consider is whether or not the system is a stiff or a non-stiff problem. When selecting a MATLAB solver, `ode45` will be used for non-stiff problems. This is a general purpose solver of medium accuracy that is often the best first choice. For stiff problems, `ode15s` is used. This is a good choice when `ode45` is slow as the problem is stiff. If Cvode is selected it can be optimized appropriately. For a non-stiff problem, Linear Multistep Method is set to the 'Adams' method, and Nonlinear Solver is set to 'Functional'. For stiff problems these parameters are set to 'Backward Differentiation Formulas' and 'Newton' respectively.

See the MATLAB and Cvode documentation for more information.
https://computation.llnl.gov/casc/sundials/documentation/sundialsTB.pdf

5.3 *Software Licenses*

PeTTSy is distributed freely and without warranty under the terms of the GNU General Public License. See http://www.gnu.org/licenses/gpl-3.0.en.html.

PeTTSy makes use of the SBML Toolbox for MATLAB (http://sbml.org/Software). The PeTTSy distribution itself contains some modified code from this software. Please note the following copyright and licensing statement for this software.

```
Copyright (C) 2009-2012 jointly by the following organizations:
    1. California Institute of Technology, Pasadena, CA, USA
    2. EMBL European Bioinformatics Institute (EBML-EBI), Hinxton, UK

Copyright (C) 2006-2008 jointly by the following organizations:
    1. California Institute of Technology, Pasadena, CA, USA
    2. University of Hertfordshire, Hatfield, UK

Copyright (C) 2003-2005 jointly by the following organizations:
    1. California Institute of Technology, Pasadena, CA, USA
    2. Japan Science and Technology Agency, Japan
    3. University of Hertfordshire, Hatfield, UK

SBMLToolbox is free software; you can redistribute it and/or modify it
under the terms of the GNU Lesser General Public License as published by
the Free Software Foundation.  A copy of the license agreement is provided
in the file named "LICENSE.txt" included with this software distribution.
```

Our software also makes use of the file findjobj.m, downloadable from http://undocumentedmatlab.com/. Pease note the following statement that applies to this file.

```
Copyright (c) 2009, Yair Altman
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

The SBML export functions use SnuggleTex to generate the required MathML. This is available freely from the School of Physics and Astronomy, University of Edinburgh (http://www2.ph.ed.ac.uk/snuggletex/). Please note the following copyright and licensing statement for this software.

SnuggleTex makes use of the Saxon XSLT processor, http://saxon.sourceforge.net/
This is released under the Mozilla public license, see https://www.mozilla.org/en-US/MPL/

PeTTSy incorporates CVODE, part of the Sundials suite of solvers (Sundials 2.6.2), see https://computation.llnl.gov/casc/sundials/main.html. This is implemented via the SundialsTB Matlab interface. Please see the following publication and license

A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers, ACM Transactions on Mathematical Software, 31(3), pp. 363-396, 2005. Also available as LLNL technical report UCRL-JP-200037.

provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer (as noted below) in the documentation and/or other materials provided with the distribution.

3. Neither the name of the UC/LLNL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OF THE UNIVERSITY OF CALIFORNIA, THE U.S. DEPARTMENT OF ENERGY OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Additional BSD Notice
---
1. This notice is required to be provided under our contract with the U.S. Department of Energy (DOE). This work was produced at the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-ENG-48 with the DOE.

2. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights.

3. Also, reference herein to any specific commercial products, process, or services by trade name, trademark, manufacturer or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.