HOW TO USE BIOCHAM

A short tutorial for the beginner

by Laurence Calzone & Sylvain Soliman

To facilitate the use of this tutorial, several conventions are used: each font correspond to a type of information.

FONT CODE used in the tutorial:

`courrier:`
`- BIOCHAM model saved in a .bc file`
`- Output of BIOCHAM written in the Command panel`

Bold Times New Roman: **Commands that are typed in the "Command line" (cf. Start BIOCHAM Graphical Interface)**
Italic Times New Roman in brackets: [*keyboard shortcuts*]
IN THE GREY WINDOW (expert mode can be ignored for the first-time users): EXPERT MODE

CREATE A MODEL

We will use the example of the 6-variable Cell Cycle model of Tyson (PNAS, 1991) to illustrate the first steps of writing a model in BIOCHAM.

EXPERT MODE

For other purposes, note that this model can be downloaded from an SBML library of models and loaded in BIOCHAM: **load_sbml(*name_of_file*).**

Step 1: Open a blank document in your favorite text editor

(ex: WordPad …).

Step 2: Define the chemical reactions

First define the structure of the model by listing all the biochemical reactions of interest:

```
_=>Cyclin.
Cyclin=>_.
Cyclin+Cdc2~{p1} => Cdc2-Cyclin~{p1,p2}.
Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}.
Cdc2-Cyclin~{p1,p2} =[Cdc2-Cyclin~{p1}]=> Cdc2-Cyclin~{p1}.
Cdc2-Cyclin~{p1} => Cdc2-Cyclin~{p1,p2}.
Cdc2-Cyclin~{p1} => Cyclin~{p1}+Cdc2.
Cyclin~{p1} =>_.
Cdc2 => Cdc2~{p1}.
Cdc2~{p1} => Cdc2.
```

Step 3: Assign kinetics to the rules

When available and for the numerical models, write the corresponding kinetics, either explicitely:

```
k1                             for _=>Cyclin.
k2*[Cyclin]                    for Cyclin=>_.
k3*[Cyclin]*[Cdc2~{p1}]        for Cyclin+Cdc2~{p1} => Cdc2-Cyclin~{p1,p2}.
k4p*[Cdc2-Cyclin~{p1,p2}]      for Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}.
k4*([Cdc2-Cyclin~{p1}])^2*[Cdc2-Cyclin~{p1,p2}]
            for Cdc2-Cyclin~{p1,p2} =[Cdc2-Cyclin~{p1}]=> Cdc2-Cyclin~{p1}.
k5*[Cdc2-Cyclin~{p1}]          for Cdc2-Cyclin~{p1} => Cdc2-Cyclin~{p1,p2}.
k6*[Cdc2-Cyclin~{p1}]          for Cdc2-Cyclin~{p1} => Cyclin~{p1}+Cdc2.
k7*[Cyclin~{p1}]               for Cyclin~{p1} =>_.
k8*[Cdc2]                      for Cdc2 => Cdc2~{p1}.
k9*[Cdc2~{p1}]                 for Cdc2~{p1} => Cdc2.
```

or use the following shortcuts: MA(k) for mass action law kinetics and MM(Vm,Km) for Michaelian kinetics.

In the law of mass action case, the parameter given as an argument will be multiplied by all the reactants' concentrations to provide the kinetic law.
Ex:
```
MA(k3)                        for Cyclin+Cdc2~{p1} => Cdc2-Cyclin~{p1,p2}.
```
is interpreted as
```
k3*[Cyclin]*[Cdc2~{p1}]       for Cyclin+Cdc2~{p1} => Cdc2-Cyclin~{p1,p2}.
```

In the Michaelian kinetics case, the two arguments represent the Vm and Km of the Michaelian kinetics; the law has the form: Vm*[*S*]/(Km+[*S*]), where *S* is the substrate.
Ex:
```
MM(k10,Ka)                    for Cdc2-Cyclin~{p1} => Cdc2-Cyclin~{p1,p2}.
```
is interpreted as
```
k10*[Cdc2-Cyclin~{p1}]/(Ka+[Cdc2-Cyclin~{p1}])
                              for Cdc2-Cyclin~{p1} => Cdc2-Cyclin~{p1,p2}.
```

EXPERT MODE

Note that a rule can also be defined under specific conditions. For example, the synthesis of protein B activates only when the concentration of protein C is above a threshold value.
```
if [C]>0.15 then 1 else 0   for _=>B.
```

or in the Command line:
add_rule(if [C]>0.15 then 1 else 0   for _=>B).

Step 4: Define the parameter values

Define the values of the parameters used in the kinetic laws.

```
parameter(k1,0.015).
parameter(k2,0).
parameter(k3,200).
parameter(k4p,0.018).
parameter(k4,180).
parameter(k5,0).
parameter(k6,1).
parameter(k7,0.6).
parameter(k8,100).
parameter(k9,100).
```

Step 5: Define macros

Some macros can be used to define algebraic functions

```
macro(YT,[Cyclin]+[Cyclin~{p1}]+[Cdc2-Cyclin~{p1,p2}]+[Cdc2-Cyclin~{p1}]).
```

EXPERT MODE

You can add events that occur at specific moments. In the example below, we define the following property: the number of nuclei N,

initially set to 1, doubles every time the concentration of the protein A reaches the value 0.5. At the end of the simulation, the parameter is reset to its initial value.
parameter(N,1).
add_event([A]>0.5,N,2*N).

Step 6: Define the initial conditions

```
present(Cdc2,1).
absent({Cdc2, Cyclin, Cdc2-Cyclin~{p1,p2}, Cdc2-Cyclin~{p1}, Cyclin~{p1}}).
```

or you can type directly :

```
present(Cdc2,1).
make_absent_not_present.
```

All the initial conditions do not need to be specified. By default, the molecules are considered undefined for the boolean case and absent for the numerical one.

EXPERT MODE

It is possible to reset the initial conditions from a simulation at a specific time *t*. The data is saved at time *t* and used as initial conditions for the next simulation with the command (here *t*=100):
set_init_from_trace(100).

```
present(Cdc2,0.387461653).
present(Cdc2~{p1},0.386421469).
present(Cyclin,0.000194669398).
present(Cdc2-Cyclin~{p1,p2},0.0329465575).
present(Cdc2-Cyclin~{p1},0.193170321).
present(Cyclin~{p1},0.0993812887).
```

Step 7: Add a temporal logic specification to the model

When available, you can integrate biological information to the model.
All the qualitative or quantitative biological information describing the behavior of the system or individual molecules can be written in temporal logic. See CTL/LTL specification section in the documentation at http://contraintes.inria.fr/BIOCHAM/DOC/biocham.html for more details.
The possible properties are, for example, that a molecule can be reached, or its concentration can oscillate, or that a protein A is needed for the activation of another protein B.

```
add_specs({
      reachable(Cdc2-Cyclin~{p1}),
      reachable(Cyclin~{p1}),
      …
      oscil(Cdc2-Cyclin~{p1,p2}),
      oscil(Cyclin),
      …
```

```
        checkpoint(Cdc2-Cyclin~{p1,p2},Cdc2-Cyclin~{p1}).
}).
```

EXPERT MODE

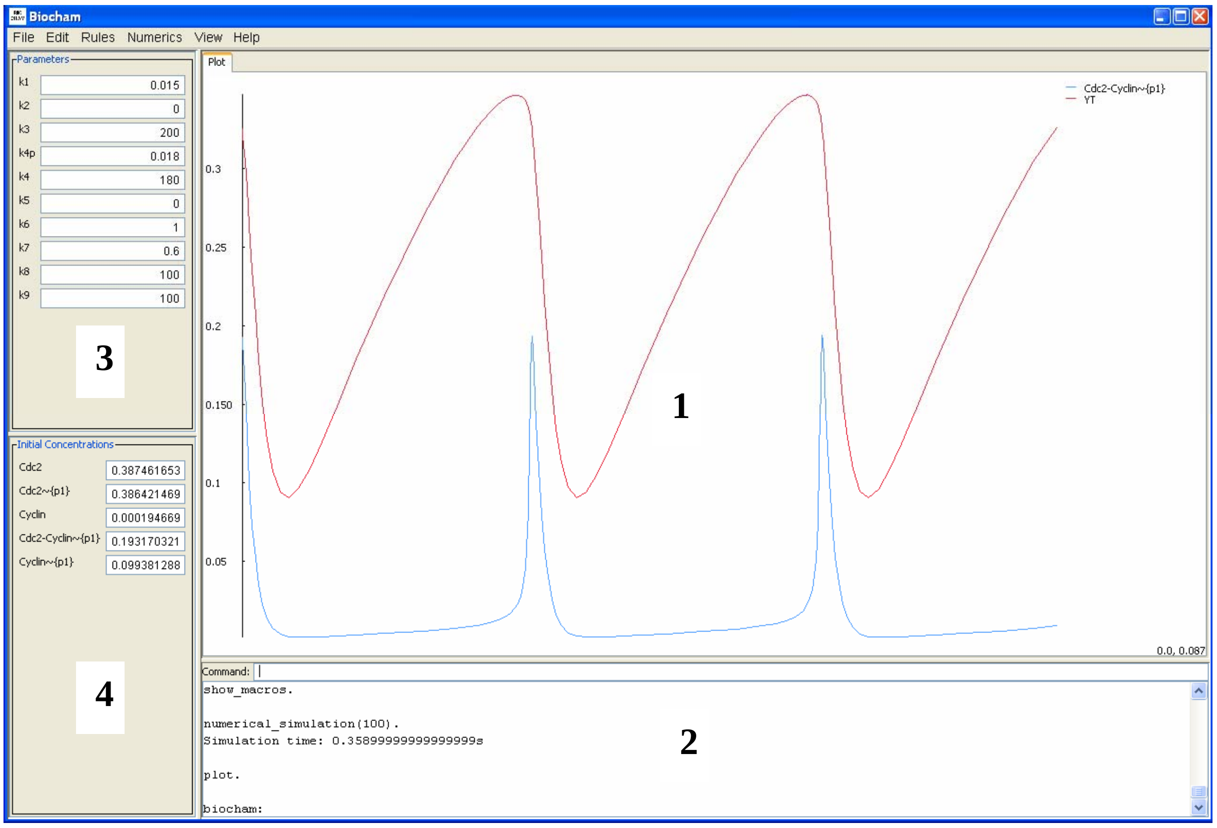
This specification can be saved in the same file as the model (rules, initial conditions and parameters) or in a different file with a .bc extension (specif.bc) and added to the model with the command:
add_biocham(specif).
Alternatively, the specification can be directly typed in the BIOCHAM Command panel:
add_specs({ reachable(Cdc2-Cyclin~{p1}), ...}).

Step 8: Save the file

Save the file as: name_of_file.bc (.bc extension).

START BIOCHAM GRAPHICAL INTERFACE

Click on the BIOCHAM icon



Parameters	
k1	0.015
k2	0
k3	200
k4p	0.018
k4	180
k5	0
k6	1
k7	0.6
k8	100
k9	100

Initial Concentrations	
Cdc2	0.387461653
Cdc2~{p1}	0.386421469
Cyclin	0.000194669
Cdc2-Cyclin~{p1}	0.193170321
Cyclin~{p1}	0.099381288

```
Command: |
show_macros.

numerical_simulation(100).
Simulation time: 0.35899999999999999s

plot.

biocham:
```

The graphical interface is divided in 4 panels:

- The *Plot* panel (**1**): the simulation (solution of the system of differential equations) can be plotted in this panel. Several plots or data sets can be viewed in different tabs (see later for more details).

- The *Command* panel (**2**): Any BIOCHAM command can be typed directly in the Command line. The BIOCHAM shell just below the Command line shows the output of all BIOCHAM commands (launched by the menu, a shortcut, or the Command line).

- The *Parameters* panel (**3**): all the parameters are listed with their corresponding values. The parameter values can be modified in this panel by clicking on the value and by assigning a new value. The change needs to be validated by clicking on <return>. The parameter can be changed manually with the command:
parameter(k1,0.1).

- The *Initial Concentrations* panel (**4**): similarly, initial conditions can be modified as in the parameter case or by typing in the Command line:
present(Cdc2,0.5).

In the Menu, several options and shortcuts are available. For example, a BIOCHAM file can be opened by choosing 'File' then 'Open' from the Menu or directly with the shortcut: [*Ctrl+O*].

RUN A MODEL

Boolean simulation

- Generate the graph of interactions

Another way to look at the model is through a graph of interactions. To generate the interaction graph, type the command:
export_dot(*name_of_file*).

To view the graph as a picture, you need to have Graphviz installed on your machine. Type the command in a different shell (if you are using Windows, use Cygwin, for example):
dot –Tpng name_of_file.dot > name_of_file.png

- Run the simulation

For the boolean simulation, each variable is defined as present or absent. If this step was omitted during the writing of the model, it is now possible to list the variables that are present and force the other variables as absent with the command line:
present(A).
make_absent_not_present.

To launch the simulation, type the command:
boolean_simulation.
which will run the random simulation up to 100 transitions, or
boolean_simulation(*integer*).
up to a specified number of transitions.

Numerical simulation

- Run the simulation

With a predefined set of initial conditions, the system is solved.
In the Menu, choose 'Numerics' then 'Num. Simulation' or type the command line:
numerical_simulation.
By default, the length of the simulation is 20 units of time.

The length of the simulation can be determined by specifying a domain (integer) as an argument in the command:
numerical_simulation(*integer*) .
or in the pop-up window when ‘Num Simul. For’ is chosen from the ‘Numerics’ menu (or with the shortcut [*Ctrl-N*]).

Note that if you use the menu or the shortcut, the command **plot** will be launched immediately after the computation is finished.

- Plot the simulation

The shortcut [*Ctrl+P*] and the command:

- Plot the molecules/macros

You can choose the molecules to plot by specifying:
show_molecules(*A*)
and
hide_molecules(*B*).
The color of each molecule can be changed by clicking on the legend and a palette appears. The color is remembered and saved permanently.

Similarly, you can choose to plot some of the macros:
show_macros({*f1, f2*}).
or all of them:
show_macros.

- Use the set of nonlinear ordinary differential equations

The Menu, 'Rule' then 'Show kinetics' and the command:
show_kinetics.
generate automatically, in the lower panel, the corresponding set of ODEs .
These equations can be saved as an ode file to be read by another solver of ODEs (e.g. xppaut) by typing in the Command line:
export_ode(*name_of_file*).
or can be saved as a latex file:
export_ode_latex(*name_of_file*).
Or in the menu, select 'File' then 'Export ODE' or 'Export LaTeX'.

Similarly the parameter values and the initial conditions can be saved as a different BIOCHAM files:
export_parameter(*name_of_file*).
export_init(*name_of_file*).
That way, changes can be made and saved in a different file than the one of the initial model.

- Set the numerical methods

By default, the numerical method used to solve the system of differential equations is Rosenbrock. The user can change

The method
=> Runge-Kutta:
numerical_method(rk).

=> Rosenbrock:
numerical_method(stiff).

The step

=> For Runge-Kutta:
By default, the initial step size is set to 0.01 but it can be specified manually:
step_size(*float*).
Similarly, by default, the step-doubling error is 0.005 but the following commands can modify it:
step_doubling(*float*).
no_step_doubling.

=> For Rosenbrock:
Only the step-doubling error can be modified in this method.

BOOLEAN ANALYSIS

Note that NuSMV is required for the boolean analysis.

Verify the Formal Biological Properties

Verify that the model satisfies the specification (biological properties)
The biological properties can document the model with a list of CTL formulae called the specification. See step 7 in the part "Create a Model".

The command:
check.
verifies if the specification is true or false.

Apply Model-checking

Query the structure of an existing model
When a network is translated in BIOCHAM, it is possible to make some queries about paths of the network (gene or protein networks or signalling pathways).
These queries concern:

- Reachability properties: is the molecule *A* able to activate or to be reached?
`reachable(A)` or in temporal logic: `Ei(EF(`*A*`)).`

- Oscillation properties: does the molecule *A* activate and inactivate consecutively?
`oscil(`*A*`)` or in temporal logic: `Ai(…)`

- Checkpoint properties: can the molecule *A* activate without *B* active?
`checkpoint(`*B*`,`*A*`)` or in temporal logic: `Ai(…)`

For example:

```
   biocham: check_checkpoint(Cdc2-Cyclin~{p1,p2},Cdc2-Cyclin~{p1}).
   Ai(!(E(!(Cdc2-Cyclin~{p1,p2}) U Cdc2-Cyclin~{p1}))) is true

   biocham: check_checkpoint(Cyclin~{p1},Cdc2-Cyclin~{p1}).
   Ai(!(E(!(Cyclin~{p1}) U Cdc2-Cyclin~{p1}))) is false
```

If one of the CTL formulae is not verified, BIOCHAM proposes a counter-example.

```
   biocham: why.

   Cdc2 is present
   1 _=>Cyclin.
   Cyclin is present

   9 Cdc2=>Cdc2~{p1}.
   Cdc2 is absent Cdc2~{p1} is present

   3 Cyclin+Cdc2~{p1}=>Cdc2-Cyclin~{p1,p2}.
   Cdc2~{p1} is absent Cyclin is absent
```

Correct / Validate a model

Correct an incomplete model
If the CTL formulae are not verified, BIOCHAM searches for rules to complete the model with the commands:
learn_one_addition(*reaction_patterns*).
to learn one rule to add or
learn_one_deletion.
to learn one rule to suppress.

As an example, we delete one rule from the Tyson model and verify that BIOCHAM is able to find the missing rule:

```
biocham: delete_rules(Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}).
biocham: learn_one_addition(elementary_interaction_rules).
Time: 9.00 s
Rules tested: 118

3 Good rules to be added:
Cdc2-Cyclin~{p1,p2} =[Cyclin]=> Cdc2-Cyclin~{p1}
Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}
Cdc2-Cyclin~{p1,p2} =[Cdc2-Cyclin~{p1,p2}]=> Cdc2-Cyclin~{p1}
```

The missing rule (in red) was found among 3 rules proposed by BIOCHAM to complete the model.

The command:
revise_model.
proposes additions and deletions if more than one rule needs to be learned.

A pattern can be provided as an argument to narrow the search.
Some are defined as shortcuts and can be used directly, such as reactions of *phosphorylation* or *dephosphorylation*; *complexation* or *decomplexation*; *synthesis* or *degradation*; all of them in an argument *elementary_interaction_rules* or with catalysts *more_elementary_interaction_rules*.

NUMERICAL ANALYSIS

Check the rules

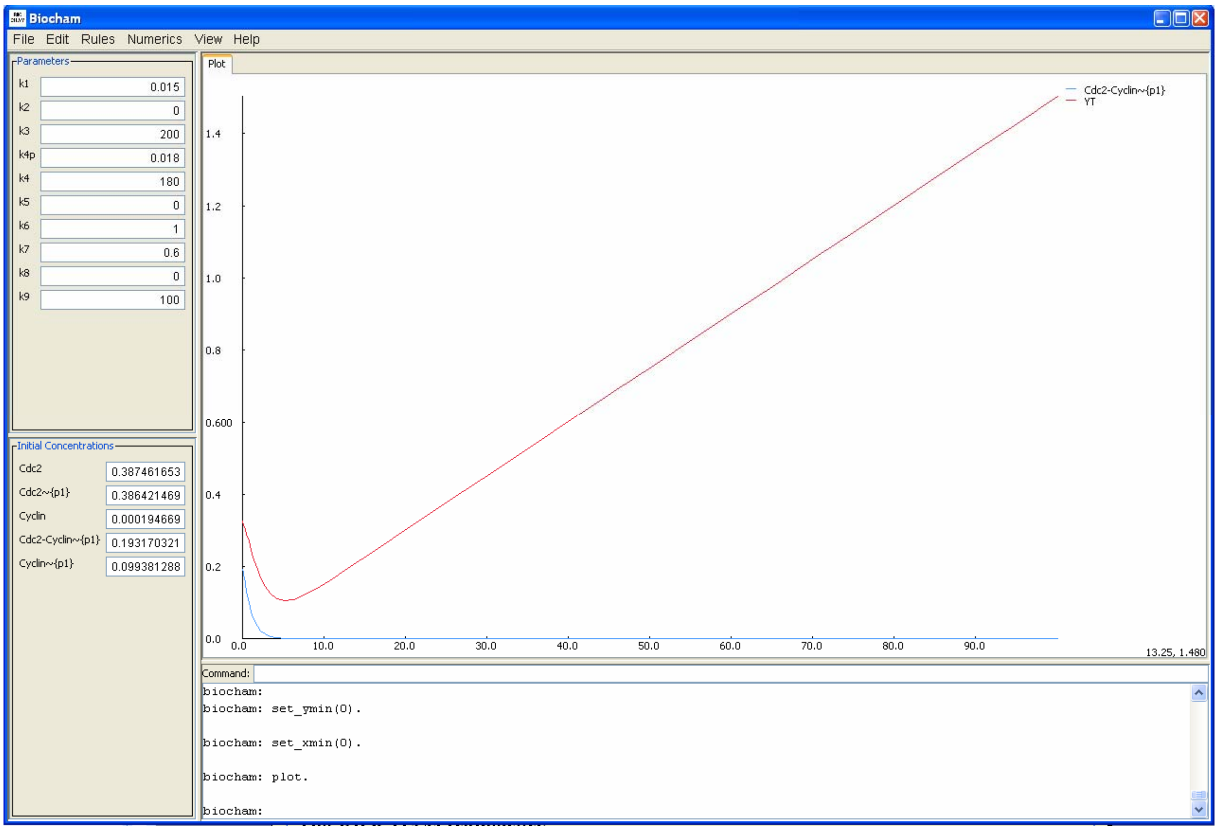
Similar to the boolean analysis, it is possible to query the numerical simulation.
For example,
trace_check(F([A]>0.15).
checks that the concentration of the molecule A reaches a value above 0.15.

Find parameter values

BIOCHAM can assist the modeler in finding the appropriate values for some parameters. The search for the parameter value is oriented by the LTL specification.
learn_parameters([k1,k2],[(0,10),(0,50)],20,oscil(A,2) & period(A,24),300).
We search for values of k1 and k2 in the respective domains [0,10] and [0,50] with 20 different tries for each value such that the molecule A must oscillate twice with a period of at least 24 in 300 time units.

Example:
From the example used in this tutorial, one parameter is changed:

k8=0 instead of 100 as in the initial model.



Parameters

k1	0.015
k2	0
k3	200
k4p	0.018
k4	180
k5	0
k6	1
k7	0.6
k8	0
k9	100

Initial Concentrations

Cdc2	0.387461653
Cdc2~{p1}	0.386421469
Cyclin	0.000194669
Cdc2-Cyclin~{p1}	0.193170321
Cyclin~{p1}	0.099381288

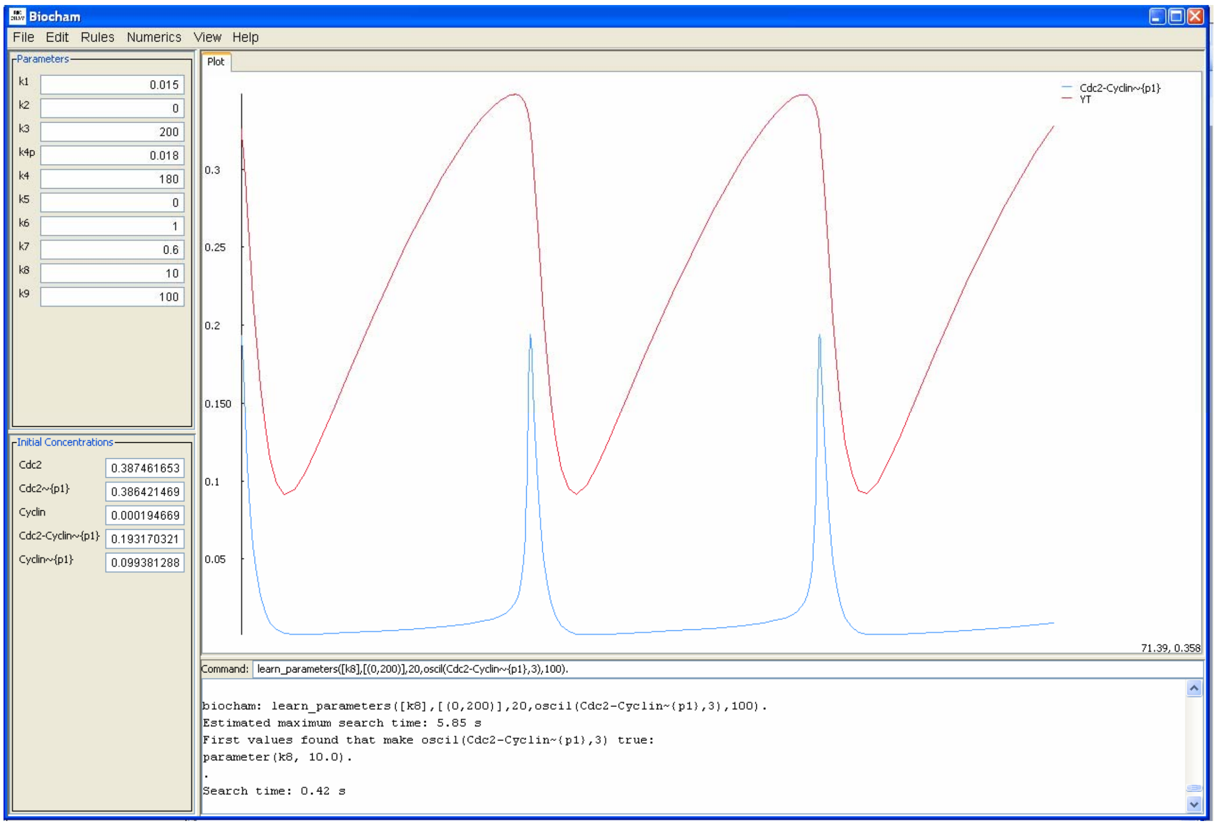
Command:

```
biocham:
biocham: set_ymin(0).

biocham: set_xmin(0).

biocham: plot.

biocham:
```

Choose a parameter to vary in a given interval such that the concentration of the active complex oscillates 3 times in 100 min:
learn_parameters([k8],[(0,200)],20,oscil(Cdc2-Cyclin~{p1},3),100).

The solution proposed by BIOCHAM appears in the Command panel as follows:

```
biocham: learn_parameters([k8],[(0,200)],20,oscil(Cdc2-Cyclin~{p1},3),100).
Estimated maximum search time: 5.85 s
First values found that make oscil(Cdc2-Cyclin~{p1},3) true:
parameter(k8, 10.0).
```



Parameters

k1	0.015
k2	0
k3	200
k4p	0.018
k4	180
k5	0
k6	1
k7	0.6
k8	10
k9	100

Initial Concentrations

Cdc2	0.387461653
Cdc2~{p1}	0.386421469
Cyclin	0.000194669
Cdc2-Cyclin~{p1}	0.193170321
Cyclin~{p1}	0.099381288

Command: learn_parameters([k8],[(0,200)],20,oscil(Cdc2-Cyclin~{p1},3),100).

```
biocham: learn_parameters([k8],[(0,200)],20,oscil(Cdc2-Cyclin~{p1},3),100).
Estimated maximum search time: 5.85 s
First values found that make oscil(Cdc2-Cyclin~{p1},3) true:
parameter(k8, 10.0).
.
Search time: 0.42 s
```

The minimum value for k8 to get 3 oscillations in 100 min is 10.

BEFORE QUITTING

The model can be saved using different formats.

You can choose from the menu ‘File’, the type of file you want to export or type directly:

To save the modifications of the model:

- A BIOCHAM file:
export_biocham(*name_of_file*).

To save the reactions:

- An SBML file:
export_SBML(*name_of_file*).

To save the set of differential equations:

- An ODE file (that can be read by XPPAUT):
export_ODE(*name_of_file*).

- A LaTeX file:
export_ode_latex(*name_of_file*).

To save the structure of the model:

- A DOT file:
export_dot(*name_of_file*).

Finally, to quit BIOCHAM, you can type in the command panel:
quit.
or you can use the shortcut [*Ctrl+Q*],
or you can click on the cross at the top right of the BIOCHAM window.