COPASI Documentation

Version 4.6 (Build 32)

COPASI Development Team on August 3, 2010

Contents

List of Tables

Chapter 1

Model Creation

1.1 Introduction

The COPASI graphical user interface has been written using the Qt application framework. This allows us to release COPASI on all platforms that Qt supports.

It also has the advantage that COPASI essentially behaves the same on all platforms supported, while still showing platform specific behavior. E.g. on a computer running Mac OS X, the user will have the menu at the top of the screen and the menu entry for the about dialog will appear in the COPASI menu rather than the help menu.

In the following sections, we will explain how to use the graphical user interface of COPASI. Everything should be applicable to all supported platforms. If there is a difference for some platforms we will try to point that out explicitly.

1.2 Commandline Version and Commandline Options

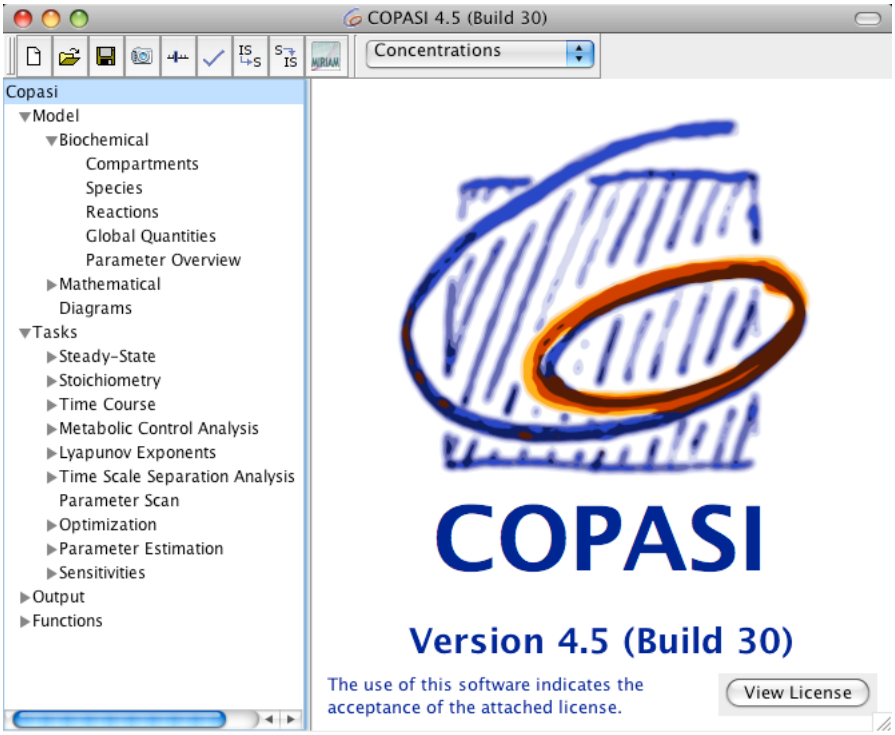
COPASI comes in two versions. One (CopasiUI) with a GUI for interactive work and one (CopasiSE) without a GUI for batch processing of model files. Both versions support the same set of commandline arguments, although some do not make sense for the GUI version and will be ignored.

In addition to the commandline options, you can specify one or more COPASI (or Gepasi) files after the commandline options which will then be processed. Specifying more than one file again only makes sense for the commandline version of COPASI. For each of the files, the activated tasks will be run. The commandline options for importing and exporting SBML as well as the –save option are ignored if more than one file is specified. Since the GUI version of COPASI can only handle one file at a time, it only makes sense to specify one file after the commandline options.

Option	Argument	Description
-s, –save	string value	This option is used to specify the name file where COPASI should store a model. This is useful if you intend to convert some SBML files to COPASI files in a batch job. This also makes sense only for the commandline version and will be ignored by the GUI version.
-i, –importSBML	string value	This options lets you specify an SBML file that COPASI shall import.
-e, –exportSBML	string value	With this option you can specify a name for the SBML file COPASI should export. This is useful if you want to export some COPASI files to SBML in a batch job. This only makes sense for the commandline version and it will be ignored by the GUI version.
–SBMLSchema	L1V2, L2V1, L2V2, L2V3	This switch works in combination with the –exportSBML switch and determines which SBML level and version is going to be used for SBML export. Currently the following schemas are supported SBML Level 1 Version 2 (L1V2), SBML Level 2 Version 1 (L2V1), SBML Level 2 Version 2 (L2V2), and SBML Level 2 Version 3 (L2V3). If no schema is given, the export creates SBML Level 2 Version 3 files.
–exportBerkeleyMadonna	string value	With this option you can specify a name for the Berkeley Madonna file COPASI should export. This is useful if you want to export some COPASI files to Berkeley Madonna file format in a batch job. This only makes sense for the commandline version and it will be ignored by the GUI version.
–exportC	string value	With this option you can specify a name for the C source file COPASI should export. This is useful if you want to export some COPASI files to C source code in a batch job. This only makes sense for the commandline version and it will be ignored by the GUI version.
–exportXPPAUT	string value	With this option you can specify a name for the XPPAUT file COPASI should export. This is useful if you want to export some COPASI files to XPPAUTs ODE file format in a batch job. This only makes sense for the commandline version and it will be ignored by the GUI version.
-c, –copasidir	string value	This specifies the directory where COPASI has been installed. It is needed to find e.g. help files. On Windows and Mac OS X this is set automatically. On Linux it has to be specified if you want to use certain features. The GUI version of COPASI will issue a warning on startup if this has not been set. The commandline version does not need this directory to be specified and therefore ignores this option.
–configdir	string value	This can be used to specify the directory where COPASI stores its configuration files. Normally this is called .copasi and is located in the users home directory. But if you want COPASI to use a different one, you can specify it with this switch.
–configfile	string value	This can be used to specify the filename where COPASI loads and stores its configuration. Normally this is called COPASI and is located in the directory specified with –configdir. But if you want COPASI to use a different filename, you can specify it with this switch.
–home	string value	This can be used to tell COPASI where your home directory is located. Normally you don't have to use this.
-t, –tmp	string value	This option can be used to specify a temporary directory where COPASI will auto-save some data periodically. Normally COPASI uses the systems temporary directory (e.g. /tmp/ under Linux).
–license		With this commandline option, COPASI will print its license and exit.
–nologo		This option suppresses the output of the "Logo" when CopasiSE is run. The "Logo" usually consist of the version of COPASI and some license statement.
–validate		This commandline option can be used to validate the given file. The file can either be a COPASI file or an SBML file.
–verbose		This commandline option tells COPASI to print more output on what it is doing to std::out.

Commandline Options

1.3 COPASI GUI Elements



Elements of the COPASI User Interface.

The COPASI graphical user interface essentially consists of four elements.

On the top of the main window, you have the menu bar (on the Mac, the menu bar is on the top of the screen). Below that, you have a tool bar with some common tasks like opening a file or saving a file. The rest of the window is vertically divided into two parts by a slider. The size of the two elements can be adjusted by moving the bar that separates them. The left element is called the object tree and it shows your current model and the tasks that you can perform on this model. Depending on the element that is selected in the object tree, the view on the right will change in order to enable you to edit the model or run and modify the task you selected in the object tree.

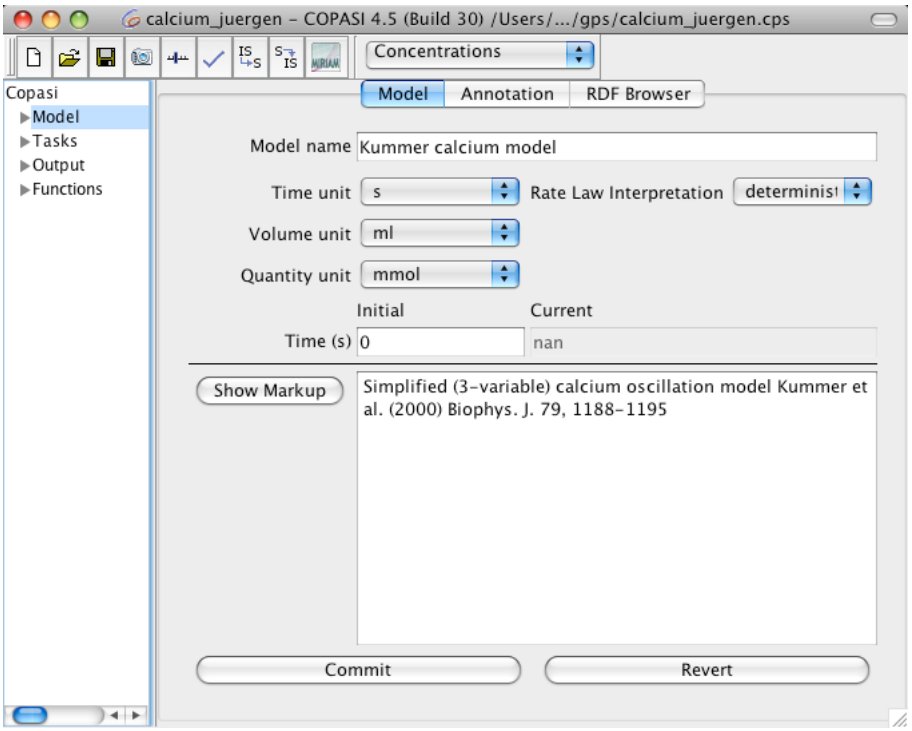
If you start COPASI without any command line argument, COPASI will start with a new model. The root of the object tree will be selected and on the right side of the main window, you will see the COPASI logo.

The object tree has five branches below the root element. The first one contains all objects that belong to the current model. The second and third ones contain all tasks that COPASI can execute, the fourth one contains the different output objects COPASI can handle and the last branch contains all the (kinetic) functions that are defined. These include the build in functions as well as functions defined by the user.

If you now click on the Model branch, the view to the right of the object tree will change and you will see a screen that allows you to make model settings. In the following sections, we will describe the individual dialogs that you can open by selecting different branches in the object tree. During this explanation, you will learn how to create a model in COPASI and run different tasks on this model like calculating a trajectory.

1.4 General Model Settings

If you click on the Model branch of the object tree which was explained in the COPASI GUI Elements section, you activate the dialog that lets you specify certain parameters for your model like its name and the units that are to be used for time, volume and concentration quantities throughout the current model. You can also give a textual description of your model that is more expressive than reactions and equations. You could for example state which part of the metabolism the model describes (e.g. glycolysis) and add some references to articles related to the model. This will help others (and yourself) to understand and identify your models.



Dialog for general Model Settings

Caution You should be aware that changing the default units actually changes the model. If you, for example, change the default volume units from liters to milliliters, all the particle numbers in your model change by a factor of 1000.

COPASI internally represents amounts of species by particle numbers. If a concentration has to be displayed or is needed for some output this is calculated from the particle number, the volume of the compartment the species belongs to and Avogadro's number $6.0221415 \cdot 10^{23} \frac{\text{particles}}{\text{mol}}$. Lets assume that you have set your default volume units to fl and your default substance units to nmol and species A is given with an the amount of $1.0 \cdot 10^{15}$ particles. Further assuming that the volume of the compartment containing A is set to 1.0 fl, we will then have a concentration of $\frac{1.0 \cdot 10^{15} \text{particles}}{6.0221415 \cdot 10^{23} \frac{\text{particles}}{\text{mol}} \cdot 1.0 \text{fl}}$ for the species A. Since the default substance unit is set to nmol instead of mol we have to multiply the result by $1.0 \cdot 10^{9}$. So COPASI would display a concentration of $1.661 \frac{\text{nmol}}{\text{fl}}$.

You are able to easily change the display between concentrations and particle numbers by selecting the drop down list in the menu bar at the top. COPASI, per default, displays the concentrations.

With the drop down list labeled Rate Law Interpretation you can specify how COPASI should interpret the kinetic rate laws you specify for your reactions. Per default, COPASI will interpret all rate laws as deterministic rate laws. Since COPASI allows the user to simulate a model either deterministically or stochastically, COPASI has to make some corrections to deterministic rate laws when using them for a stochastic simulation. This functionality sometimes interferes with rate laws that have been written for stochastic simulation and where the before mentioned corrections have already been made by the modeler. So if you have a model with rate laws that have been written be used in a stochastic simulation, you have to specify this by selecting stochastic from this drop down list. If you do this, COPASI will not apply any corrections to the rate laws specified in the model when doing stochastic simulation.

In the field labeled Time (s) under the text Initial the user can tell COPASI to take the given value as the initial time for tasks like time course. This field is only enabled for editing for non-autonomous models, i.e., models which use the value of time explicitly in an ODE, an assignment, a rate law, or an event. COPASI automatically detects whether a model is non-autonomous. If you have a non-autonomous model you can set the start time of a time course by adjusting this field. For more information on running time course simulation is COPASI see Time Course Simulation.

Caution The update model option on each task will only update the initial time for non-autonomous models otherwise it will be left at 0.

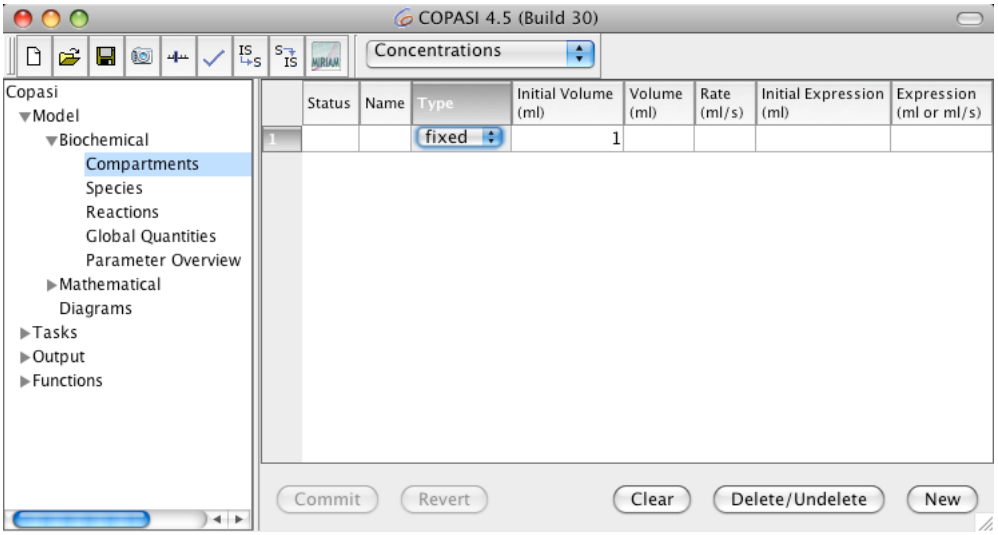
1.5 Compartments

In COPASI, most of the time there are several ways to do something and you just choose the way you prefer. This is especially true for defining the elements of the model.

Actually if you are just defining a model that has a single compartment, you will most likely not even bother to add the compartment explicitly, but we will come back to this in the Species and Reactions sections.

Although you will probably not add compartments to often, it is good to know how it is done, especially since adding other components of the model, e.g. species or reactions, works essentially the same.

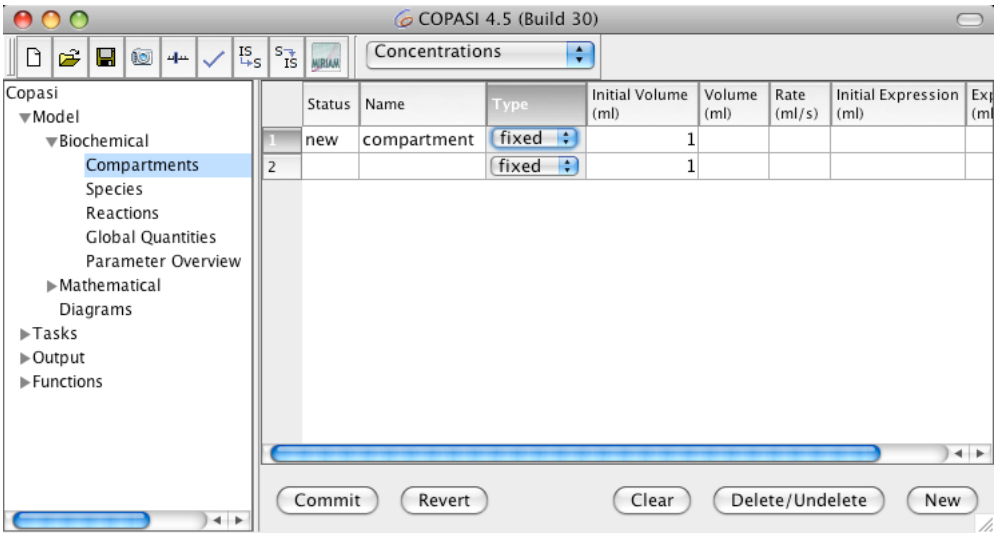
There are three methods to add a new compartment to a model, but for all three, we have to navigate to the Compartments branch of the object tree which is located under the Model->Biochemical branch. So first open the Model branch and there open the Biochemical branch by clicking on the expansion sign in front of the branch name, or by double clicking on the branch name.



Empty Compartment Table

If you start with a new model and you select the Compartments branch, you will get an empty table with eight columns (see above). The columns are named Status, Name, Type Initial Volume, Volume, Rate, Initial Expression and Expression. The meaning of the Status column will be explained below. Name is the actual name of the compartment. The Type specifies how the transient volume of the compartment is determined whereas the Initial Volume and the Volume columns show the volume of the compartment before any calculations and after re-calculation, respectively. These two values, given in the volume units defined in the model settings dialog, should be the same if the type is set to fixed. The Rate column show the rate of change of the compartment (if it has been set by a calculation). The Initial Expression and the Expression columns show the mathematical expressions used to calculate the initial compartments volume and the transient one, respectively. These fields are empty if no expressions has been set.

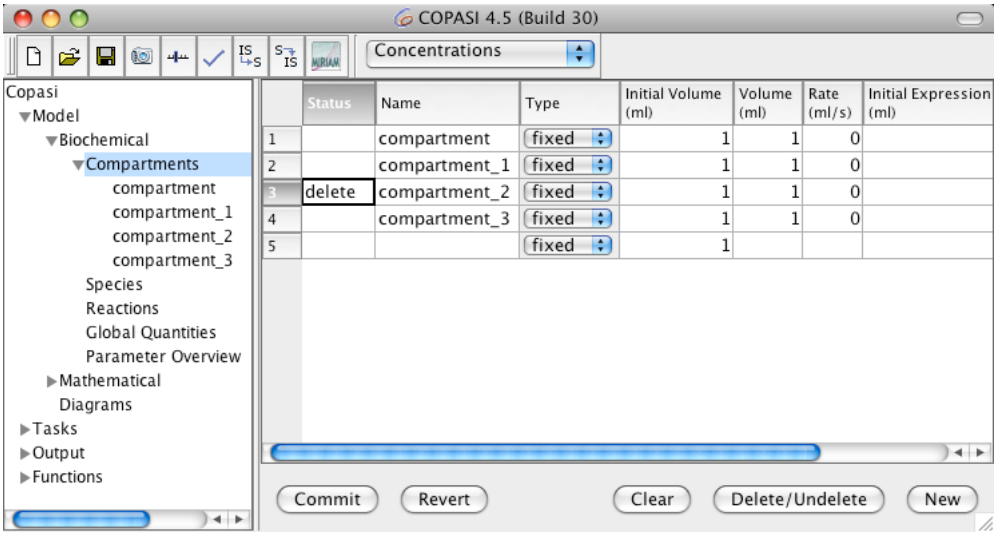
The most obvious way to add a new compartment is to click the New button on the bottom of the window. This will create a new compartment that is added to the table with a status of new (see below).



Compartment Table with new Compartment

A status of new means that the compartment has been created, but it has not been added to the model yet. It will get added to the model if you either click on the Commit button on the bottom of the screen, if you select another element in the object tree on the left, or if you double click on the table. In case of clicking on the Commit button, you will notice that the status of the new compartment is no longer defined as new since it has been added to the model.

While the status of a compartment is shown as new, you can remove the compartment from the table by clicking the Revert button which will cancel all modifications you made to the compartments that have not been committed yet.



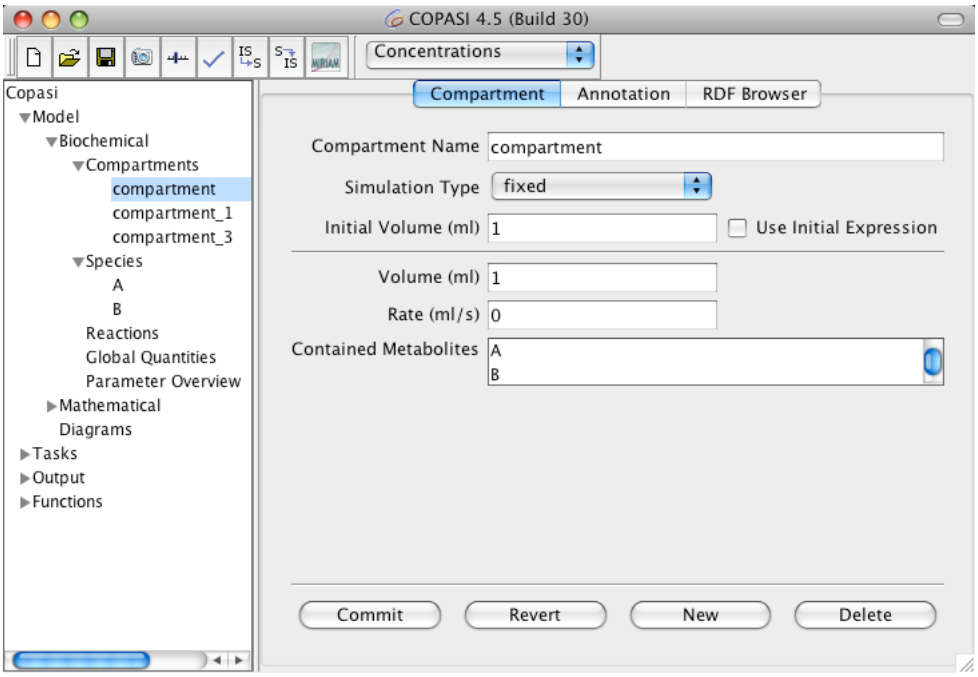
Compartment Table with deleted Compartment

If you have already committed the compartment, you can delete it by selecting the table row (or one cell of the table row) that contains the compartment you want to delete and clicking the Delete button. You will notice that the compartment does not get deleted at once, but rather the status changes to delete (see above). You can still undo the delete by clicking the Revert button or the Delete/Undelete button, or you can finalize the delete action by clicking on the Commit button. Again leaving this dialog by selecting another object in the object tree or double clicking on the table has the same effect as clicking on the Commit button.

The clear button is just a convenience function to delete all compartments. If you click on it, the status of all compartments in the table will be changed to delete and a subsequent Commit will remove the compartments from the model.

The most convenient way to add a compartment is to just click on an empty name cell in the table and type the name of the compartment. Once you leave the cell by either hitting the return or the tab key or by clicking somewhere else, the compartment appears in the table with a status of new. Actually hitting the return key after typing the name brings you directly into the next row and you can continue adding compartments until all compartments are defined. You now only have to commit your changes in one of the ways mentioned above and all the compartments get added to the model.

The third way to add a new compartment is to double click on an empty row in the table. This is essentially the same as clicking the New button and double clicking on the newly added compartment entry.



Compartment Definition Dialog

Double clicking on any compartment entry in the table will bring you to another input dialog that lets you specify the parameters of the compartment (see above). For each compartment, you are able to change the name of the compartment, the type of simulation specifying how the compartment volume, or its rate of change, will be determined, and the initial volume.

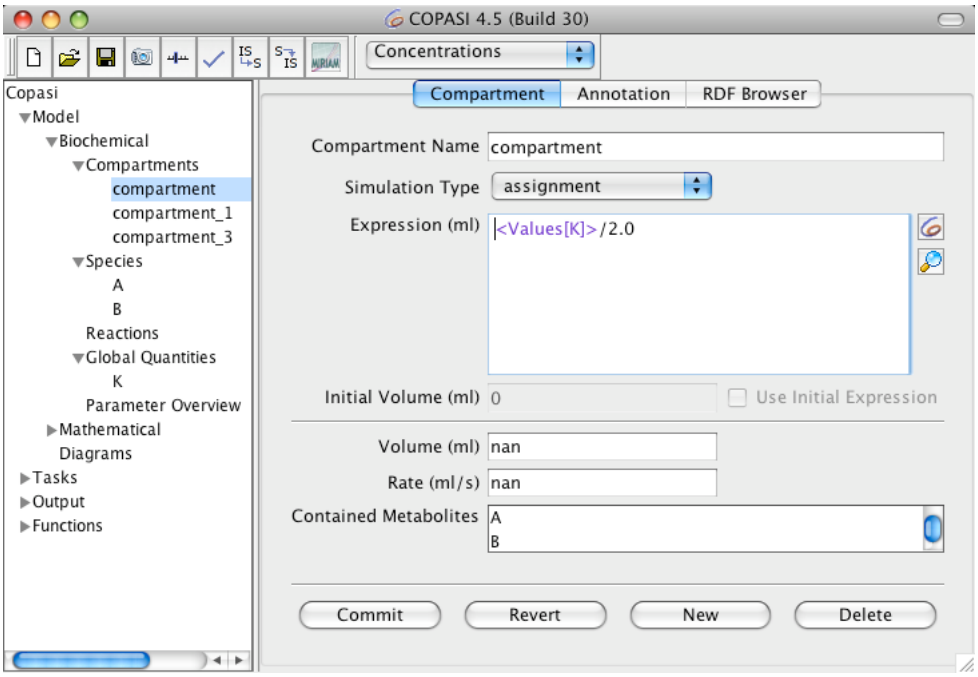
The field labeled Volume specifies the volume the compartment has right now, e.g. during or at the end of a simulation. Since the current version of COPASI does not support variable compartment volumes yet, this will always be the same as the initial volume.

As stated above, compartment volumes don't have to have a constant value but they can be reassigned during e.g. a time course simulation depending the values of one or more model entities. In order to specify whether a compartment has a constant volume or the volume is calculated on the fly according to a mathematical expression, the drop down list called Simulation Type can be used. It contains three entries:

Name	Description
fixed	the volume of the compartment has a constant value (which corresponds to the given initial value)
assignment	the volume of the compartment is determined by evaluating the given mathematical expression
ode	the rate of change of the compartments volume is determined by an ordinary differential equation

Compartments Simulation Types

If you want the compartment to be calculated from a given mathematical expression you select the entry called assignment from the drop down list. This enables a text field where the mathematical expression can be entered.

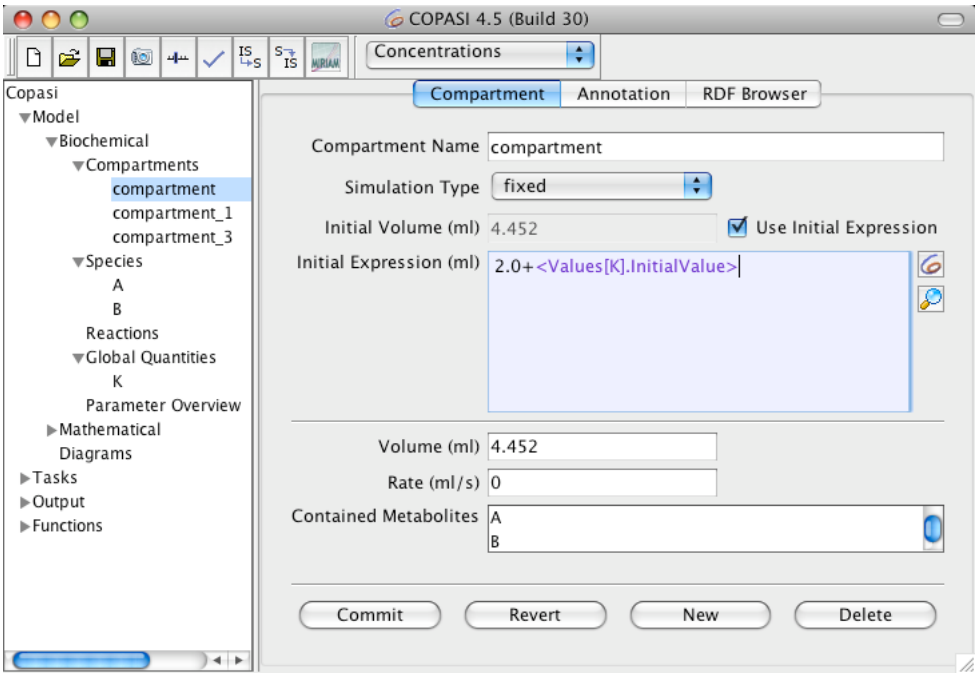


Compartment Widget with Assignment Rule

Caution Please note, it is not possible in COPASI to calculate the volume of a compartment through an assignment based on the concentration of a species contained in the compartment. The reason for this restriction is that COPASI preserves the amount S of a species during calculation, which leads to the following equation for the concentration $[S] = S/V$ and therefore $V = S/[S]$. Obviously, specifying any assignment for the compartment volume like $V = f([S])$ would lead to conflicting values.

Likewise if you want the rate of change of the compartments volume to be determined by an ordinary differential equation (ode) you select the entry called ode from the drop down list.

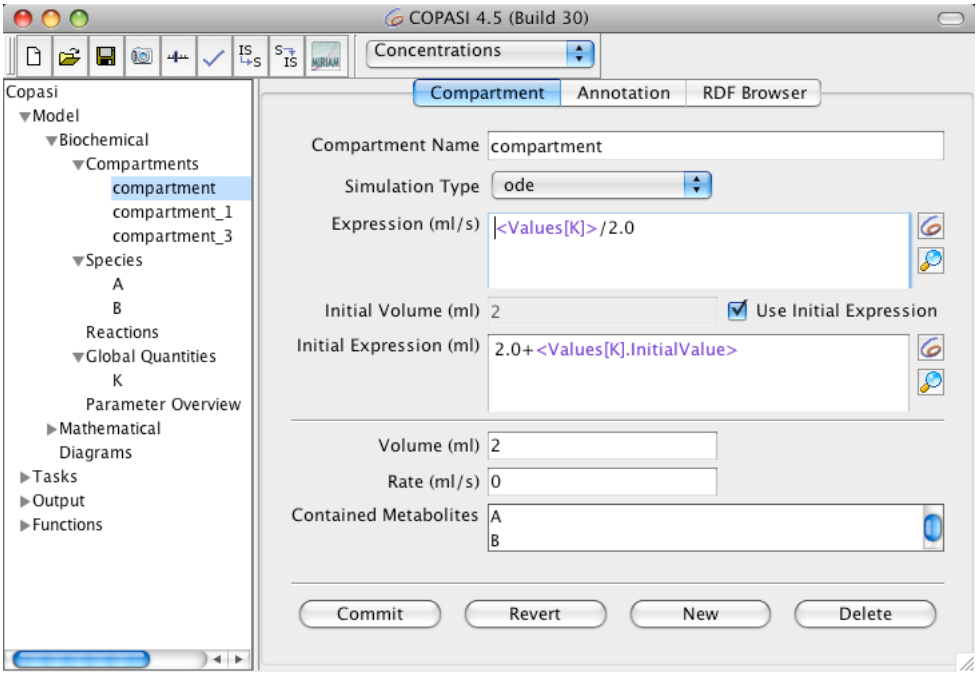
Not only the transient volume of a compartment can be specified as a mathematical expression (see above), but also the initial one (see below). If you want to specify such a mathematical expression for the initial volume of a compartment, check the check box called Use Initial Expression. An initial expression can only be specified if the Simulation Type drop down list is either set to fixed or to ode. If the list is set to assignment, the given assignment automatically acts as an initial assignment and there is no need to specify an additional initial assignment.



Compartment Widget with Initial Assignment

The mathematical expressions that can be specified for rules and initial assignments may contain the same elements as the expressions used to defined function definitions. For a detailed description of the elements see User Defined Functions. When it comes to referencing values of other model entities within mathematical expressions, there is a slight difference between the mathematical expression for a rule and that for an initial assignment. The mathematical expression for a rule may reference transient values of other model entities whereas the mathematical expression for an initial assignment may only reference initial values of other model entities.

As you might already have noticed, this dialog for changing compartment parameters is associated with the individual compartment leaves in the object tree. So if you want to change the parameters, you can also navigate to the leave in the object tree that represents the compartment you want to change instead of double clicking on an entry in the compartment table.

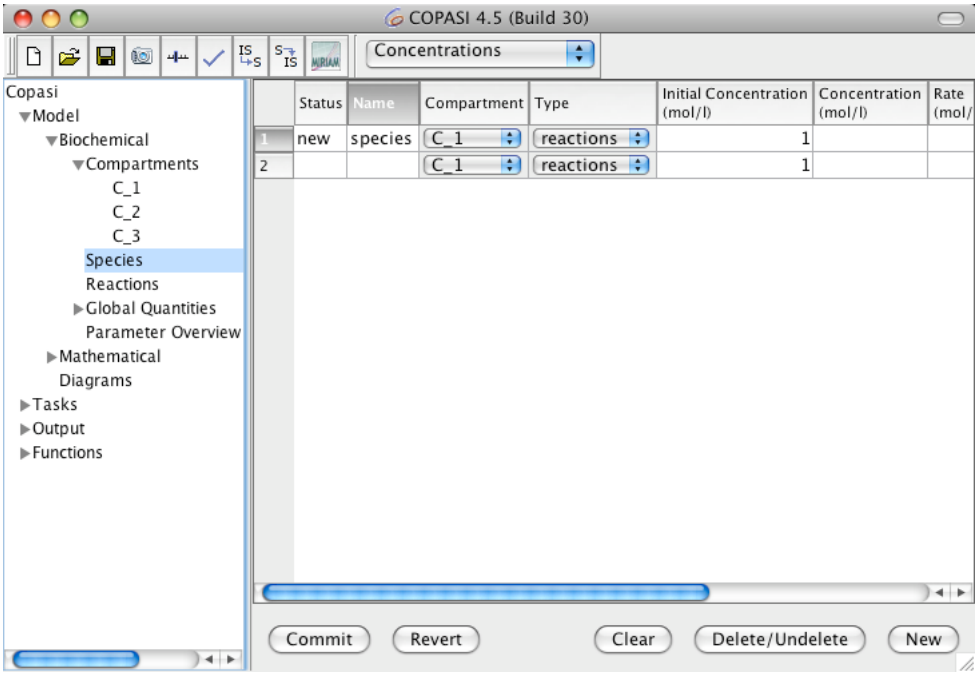


Compartment Widget with ODE Rule and Initial Assignment

If there are already species defined that are part of the compartment being edited, they will be listed in the text widget at the bottom of the dialog called Contained Species. Otherwise, you should add new species.

1.6 Species

Adding new species works exactly the same as adding new compartment, so we strongly suggest reading the section if you haven't already done so. Here we will just cover the differences between adding a compartment and adding a species.

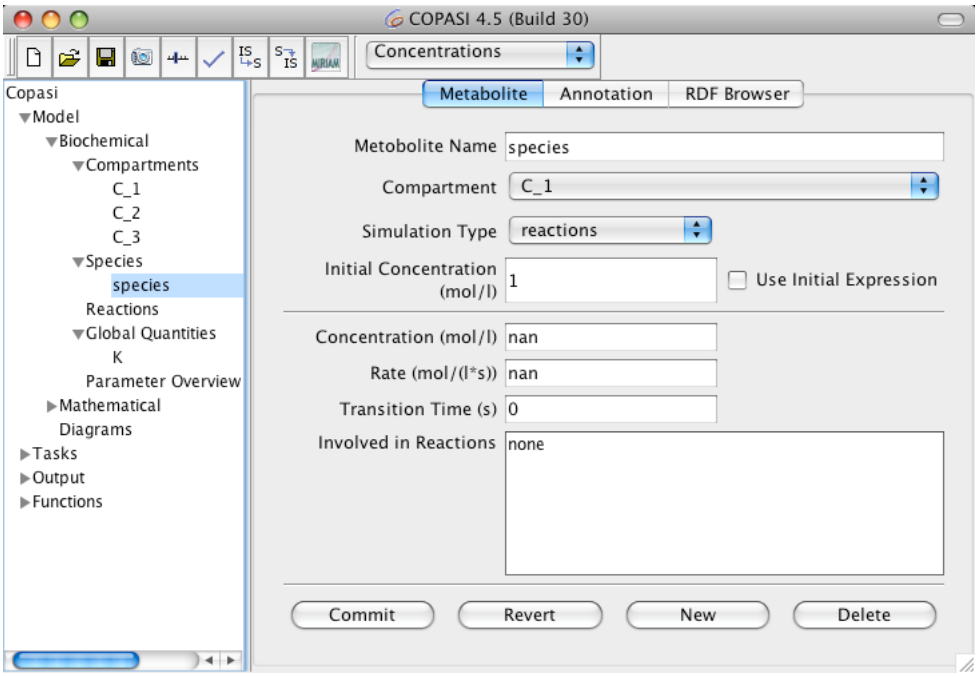


Species Table with new Entry

First of all in order to add a new species, as it is called in COPASI, you have to navigate to the Species branch of the object tree which is located in the Model->Biochemical branch directly below the Compartments. Here again you see a table (see above), but this table consists of nine columns. This is due to the fact that a species has more parameters than a compartment. The Status and Name columns should already be familiar from the compartments table. The other columns specify the compartment where the species is located in, the way the concentration of the species is calculated, initial concentration of the species, the transient concentration, the rate of change of the species as well as two mathematical expressions used to determine the initial concentration and the transient one, respectively. For newly created species the rate will be empty since it needs to be calculated first, e.g. during a time course simulation.

When a species is added and the model does not contain a compartment yet, COPASI will automatically add a new compartment to the model and the species will be added to this compartment. If there already is one or more compartments, the species will be added to the first compartment in the list. This can be changed later.

To add a new species you have the same three ways as for adding compartments and if you are not familiar with those, we recommend to read the section first.



Dialog for changing Species Parameters

Editing the parameters of a species also works exactly the way it does for compartments. Either you double click on a species entry in the table, or you use the object tree to navigate to the species leaf you intend to edit. The parameters of a species lying on below the separator line are determined automatically and can not be changed by the user. The parameters that you can change are the Species Name, the Compartment the species belongs to, and how the concentration is calculated. Unless the Simulation Type is set to *assignment*, its Initial Concentration could also be changed.

If you would rather change the initial particle number instead of the Initial Concentration, you should select *Particle Numbers* in the drop down list in the menu bar at the top. This will change all displays in the program to use particle numbers rather than concentrations and you can now enter the Initial Particle Number (note the label will be changed). The volume used to calculate particle numbers from concentration comes from the compartment associated with the species.

If you change back to initial concentration, COPASI will internally recalculate the initial particle number. In some situations COPASI will prevent the user from changing the initial concentration. This is done when changing the concentration would lead to circular dependencies during the calculation of all initial values. The cause for this is most likely an assignment rule for the containing compartment. In such a case it is still possible to change the initial particle number.

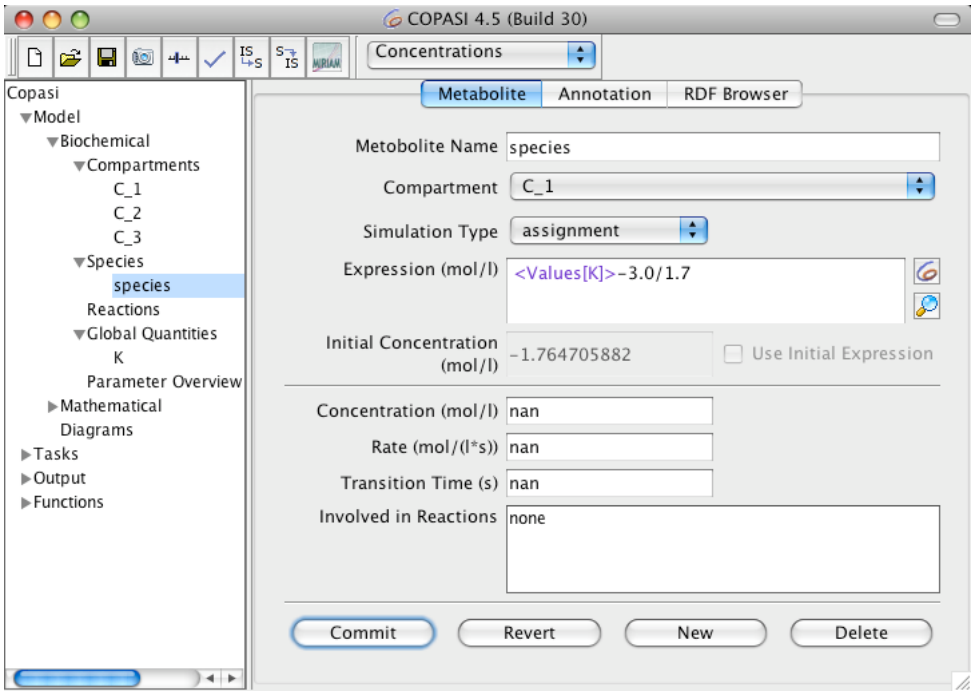
Normally the concentration of a species is determined by its initial concentration and by the reaction kinetics of the reactions it participates in either as a substrate or a product. The reactions are listed at the bottom of the species widget (see above).

Name	Description
reactions	the concentration/amount of the species is determined by the kinetic laws of the reactions that modify the species
fixed	the concentration/amount of the species has a constant value (which corresponds to the given initial value)
assignment	the concentration/amount of the species is determined by evaluating the given mathematical expression
ode	the rate of change of the species volume is determined by an ordinary differential equation

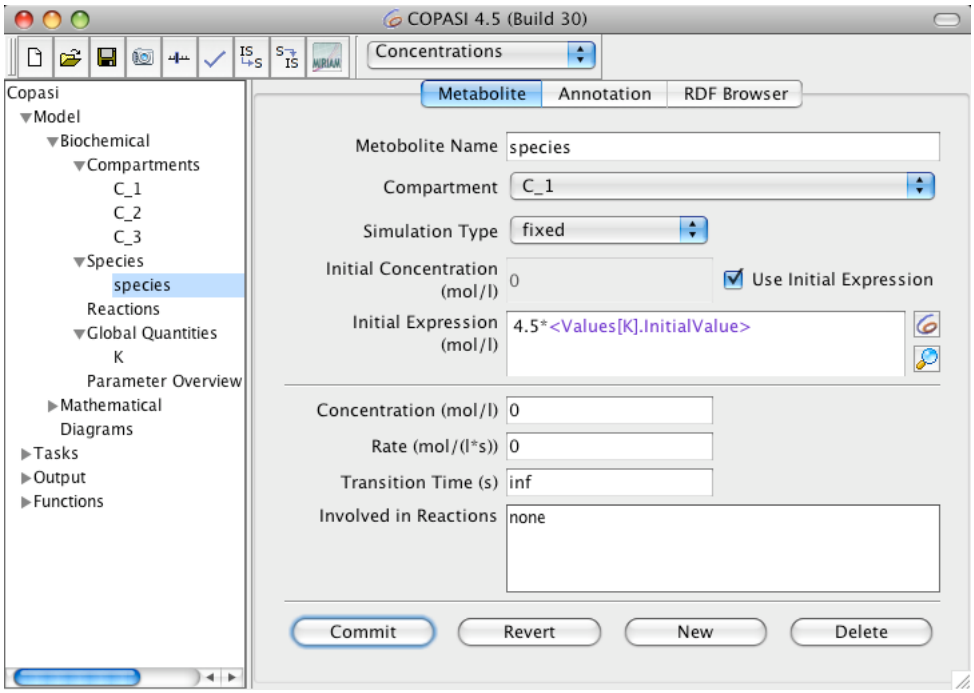
Species Simulation Types

Newer versions of COPASI also allow the concentration of a species to be determined by a mathematical expression or by providing an ordinary differential equation for its rate of change. In order to specify whether a species has a constant volume, it is determined by reactions, or its volume is calculated on the fly according to a mathematical expression, the drop down list called Simulation Type should be used. It contains four entries listed on the above table.

If you want the species to be calculated from a given mathematical expression you select the entry called *assignment* from the drop down list. This enables a text field where the mathematical expression can be entered (see below). Likewise if you want the rate of change of the species concentration to be determined by an ordinary differential equation (ode) you select the entry called *ode* from the drop down list.



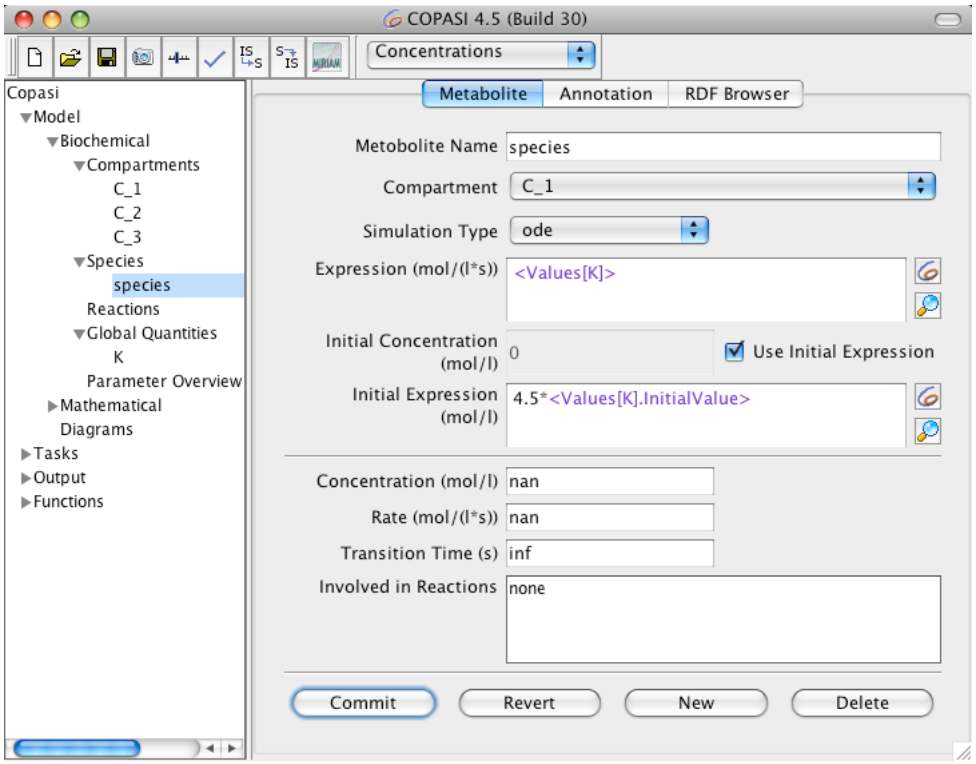
Species Widget with Assignment Rule



Species Widget with constant Species and Initial Assignment

Not only the transient concentration of a species can be specified as a mathematical expression, but also the initial concentration of the compartment. If you want to specify such a mathematical expression for the initial concentration of a species, check the check box called Use Initial Expression.

An initial expression can be specified unless the Simulation Type drop down list is set to *assignment*. Since the given assignment automatically acts as an initial assignment, there is no need to specify an additional initial assignment.



Species Widget with ODE Rule and Initial Assignment

The mathematical expressions that can be specified for rules and initial assignments may contain the same elements as the expressions used to defined function definitions. For a detailed description of the elements see User Defined Functions.

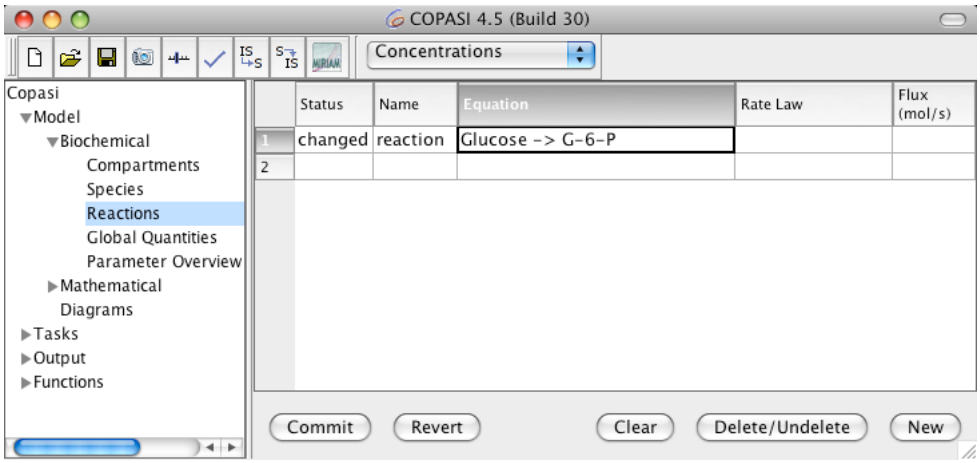
When it comes to referencing values of other model entities within mathematical expressions, there is a slight difference between the mathematical expression for a rule and that for an initial assignment. The former may reference transient values of other model entities whereas the latter may only reference initial values of other model entities.

1.7 Reactions

Again adding reactions essentially works the same way as adding compartments or species. When you navigate to the Reactions branch of the object tree which is located under the Model->Biochemical branch, you will see a table with five columns. The first two are Status and Name of the reaction. The third column called Equation describes the chemical formula and maybe additional modifiers of the reaction. The fourth column states the name of the kinetics for the reaction which depends on the equation. We will come to this in a second. The last column shows the flux through this reaction. The flux can not be set by the user but is calculated automatically when you do a time course simulation.

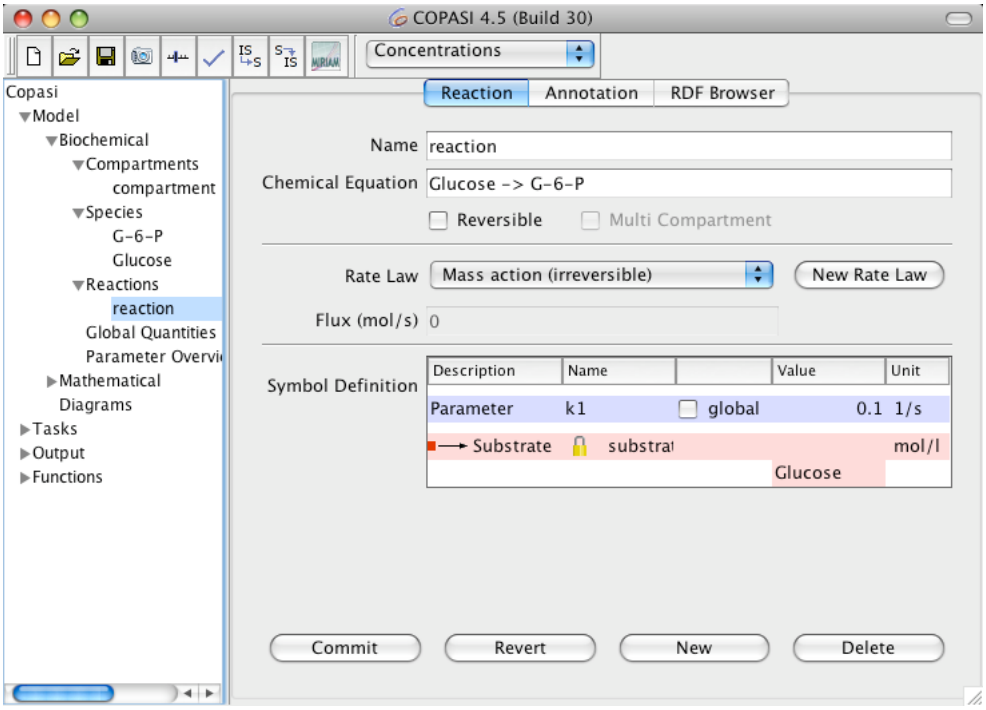
The easiest way to add a reaction is to type the chemical equation into an empty equation cell in the table. After you typed the equation, you hit the return key and automatically land in the next row where you can type the next reaction equation. This way you can enter all the reactions that make up your model. When you are finished with typing the reaction equations, you commit all the reactions. If any of the reactions contain species that are not already present in the model, they are added automatically. If there was no compartment before, a compartment is also added and all new species get added to this compartment. If there is already one or more compartments, all new species get added to the first compartment that is listed in the object tree.

Caution When typing reaction equations you should keep in mind that species names in COPASI can contain characters like "+" or even white-spaces. Since these characters would make it very hard if not impossible to parse the chemical reaction equation, you have to place those species names in double quotes. E.g. "Species 1" + "Species 2" -> "Species 3". No matter if your species names contain special characters or not, the species names have to be separated from the reaction symbols (+, *, =, and ->) by white-spaces.



Reaction Table with new Entry

Each new reaction gets a default kinetic which is irreversible mass action for reactions that contain a substrate. For reaction that only have a product (e.g. influx into a system) a constant flux kinetic is chosen.



Dialog for changing Reaction Parameters

Double clicking on an entry in the table will bring you to another dialog that lets you change certain parameters of the reaction. You can change the name of the reaction, the chemical equation and whether the reaction is reversible or not. Changing the chemical equation and the reversibility of a reaction influences the type of kinetics you can choose for the reaction. Each kinetic function defines how many substrates, products and modifiers it expects. Additionally it defines whether it can be used for reversible or irreversible reactions only or if it can be used on either. So depending on how many substrates, products and modifiers your kinetic equation has and whether it is reversible or not, only a subset of the defined kinetic functions will be available in the Kinetics combo box. If the kinetic function you want to assign to the reaction is not available yet, you can add it by clicking on the New Rate Law button (see also User Defined Functions). Depending on the kinetic function you chose, you get a selection of parameters in the table named Symbol Definition, all functions parameters get a default value of 0.1 which can be changed by clicking into the corresponding cell and typing a new value.

Per default all parameters to a kinetic function are local parameters and they exist only in the scope of the rate law of one reaction. Sometimes it is of advantage to use the same parameter in the rate laws for several reactions. This way if you want to change to value of this parameter, you only have to change it in once instead of having to change it in every reaction it occurs in individually. Parameters that can be used in more than one reaction are called global quantities in COPASI. How you add a global quantity to your model is described in the section called Global Quantities. Let us for the moment assume you already added such a global

quantity to your model and now want to use it in a rate law. Each parameter in the Symbol Definition table has a check box labeled global. When you check this box, COPASI knows that the parameter that belongs to it is a global quantity and lets you select one from a list of global quantities that have been defined in the model. If none have been defined yet, the list contains only an entry termed unknown. In this case, you will have to define one or more global quantities first and then come back to the reaction where you want to use it. If the name of the global quantity is set to the *unknown* term, COPASI will reset the type of parameter to local if you leave the reaction widget.

So far we did not go into the details of how chemical equations are to be specified. Chemical equations have a simple schema. First you state all the substrates separated by "+" characters. Please make sure that you separate the name of the substrate and the "+" character by at least one space character, otherwise COPASI will interpret the "+" sign as belonging to the species name. (Having the "+" character as part of a species name is allowed in COPASI !) after the list of substrates, you specify either an equals ("=") character if the reaction is reversible or the character combination "->" if the reaction is irreversible. This is followed by the list of products which must also be separated by the "+" character. Again make sure you have spaces around the separating "+" characters. Optionally this term can be followed by a semicolon and a list of modifiers which are separated by spaces. Either the list of substrates or the list of products may also be empty, but at least one of them must be present. Lets look at two examples:

1. Species A is irreversibly converted into Species B. The chemical equation you would type is "A -> B".
2. Species A and B are reversibly converted to Species C, the reaction has 2 modifiers C and D. The chemical equation for this in COPASI would be: "A + B = C; C D"Note that one of the modifiers is the product!

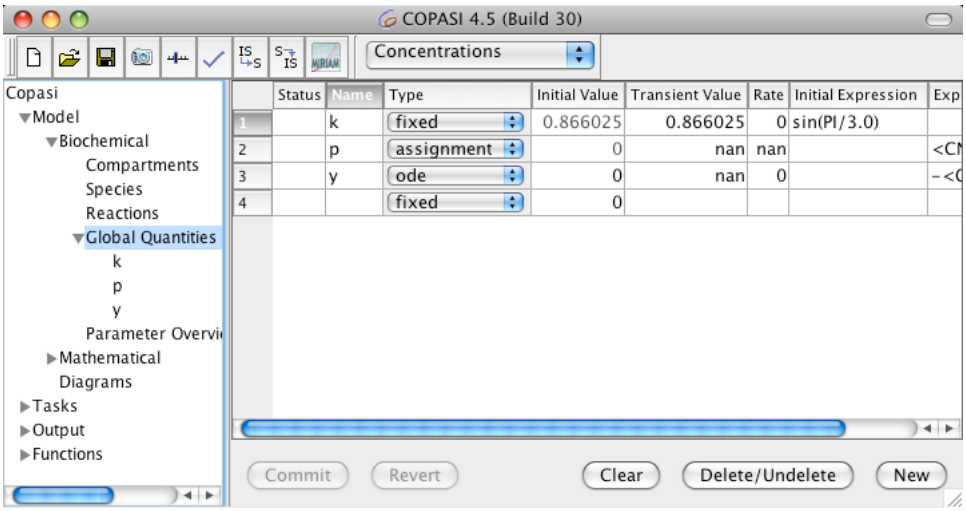
Caution If the reaction takes place in one compartment, the reaction kinetic specifies a rate of concentration change, whereas if the reaction takes place in several compartments, the kinetic specifies the amount of substance change over time.
E.g. in the reaction A -> B, if A and B are in the same compartment, the kinetic function for the reaction returns its result in mol/(l*s). If A and B reside in different compartments, the result is returned in mol/s. (This assumes that your default units are set to mol, l and s.)

1.8 Global Quantities

In the model tree right below the Reaction branch is the Global Quantities branch. If you select this branch, you see a table with all the global quantities that have been defined in your model. When you start a new model this table, just like all the others so far, is empty (see below). The table has eight columns and the ones named Status and Name should by now be already familiar. The third column specifies the type of the global quantity. There are three alternatives for that: *fixed* means the global quantity has a constant value (that can be used as a global parameter), *assignment*means the value of the quantity is calculated from a mathematical expression, and *ode*. If the type is *ode* the quantity is treated as a variable of the model and a differential equation for this variable can be provided.

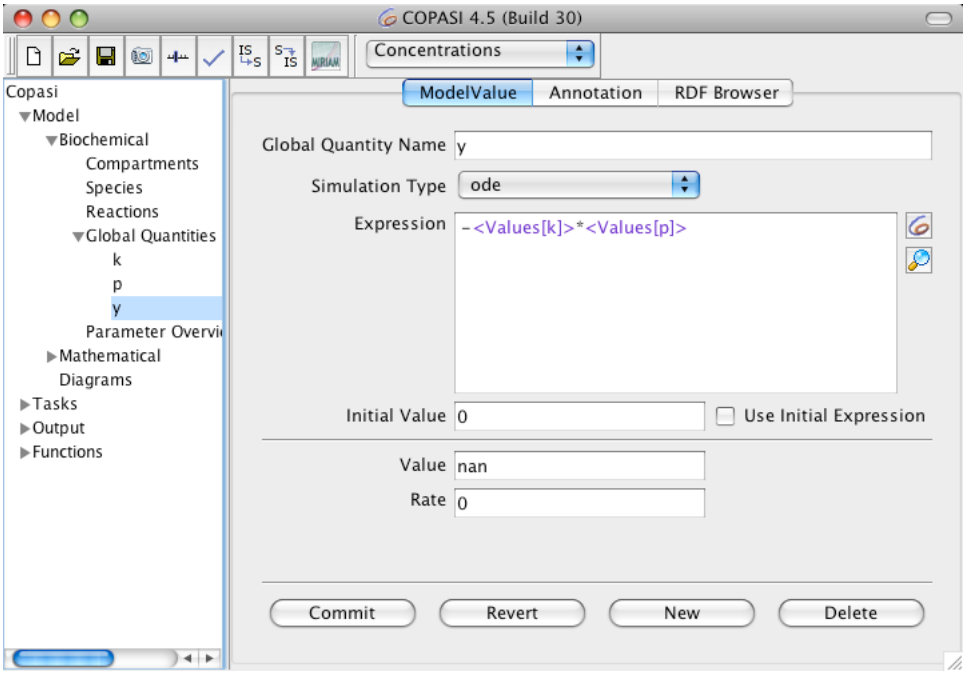
The forth and fifth column contain the initial and transient value of the global quantity as well as the rate of change of the value is displayed on the sixth column. The last two columns contains the mathematical expressions which may be required as long as the type is not *fixed*.

As with all other model elements, you can choose how you want to add a global parameter. The most convenient way is to just enter a name in an empty cell of the column termed Name.



Global Quantities Table with 3 Entries

If you click on the name of a global quantity in the tree on the left or double click on a row of the table only the information correlated with the chosen global quantity will be displayed.



Global Quantity User Interface with ODE Rrule

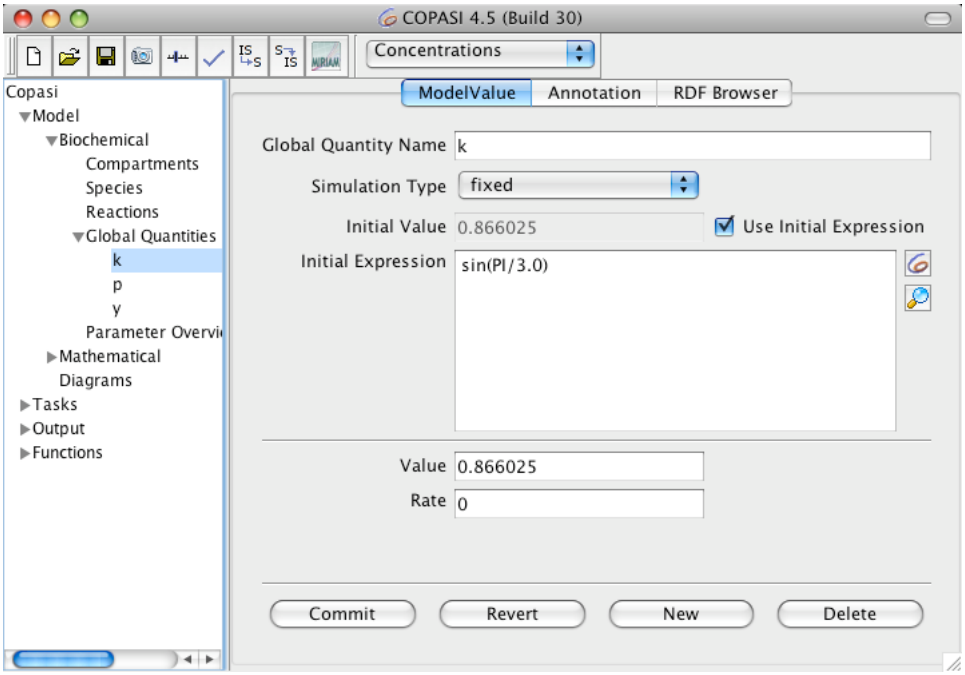
Just as compartments and species, global quantities do not have to be a constant but can be reassigned during e.g. a time course simulation depending the values of one or more model entities. In order to specify whether a parameter has a constant value or the value is calculated on the fly according to a mathematical expression, the drop down list called Simulation Type should be used. The drop down list contains three following entries:

Name	Description
fixed	the value of the parameter is constant (which corresponds to the given initial value)
assignment	the value of the parameter is determined by evaluating the given mathematical expression
ode	the rate of change of the parameters value is determined by an ordinary differential equation

Global Quantities Simulation Types

If you want the parameters value to be calculated from a given mathematical expression you select the entry called *assignment* from the drop down list. This enables a text field where the mathematical expression can be entered. Likewise if you want the rate of change of the parameters value to be determined by an ordinary differential equation (ode) you select the entry called *ode* from the drop down list.

Not only the transient value of a parameter can be specified as a mathematical expression, but also the initial value of the parameter. If you want do specify such a mathematical expression for the initial volume of a compartment, check the check box called Use Initial Expression. An initial expression can only be specified if the Simulation Type drop down list is either set to *fixed* or to *ode* . If the list is set to *assignment*, the given assignment automatically acts as an initial assignment and there is no need to specify an additional initial assignment.



Global Quantity Widget with an Initial Assignment

The mathematical expressions that can be specified for rules and initial assignments may contain the same elements as the expressions used to defined function definitions. For a detailed description of the elements see User Defined Functions. When it comes to referencing values of other model entities within mathematical expressions, there is a slight difference between the mathematical expression for a rule and that for an initial assignment. The former may reference transient values of other model entities whereas the latter may only reference initial values of other model entities.

Even so COPASI does not support building models from ordinary differential equations, one can specify a model as a set of differential equations using a set of global parameters which have the type ode.

1.9 Events

An event, which can be viewed as discrete conditional state transition of the model, consist of two required parts: a trigger, which causes the event, and at least one assignment, which modifies the model. If an event has multiple assignments, they will be carried out simultaneously, i.e., independent from the order in which they are defined. In addition, an event may include an optional time delay, which determines the model time COPASI will wait after detecting an event and the execution of the assignments.

1.9.1 Trigger

An event trigger is a Boolean expression. The exact moment at which the expression value changes from FALSE to TRUE is the time point when the event is fired. The Boolean expression may not use any of the random functions (uniform() and normal()) which COPASI provides.

1.9.2 Assignment

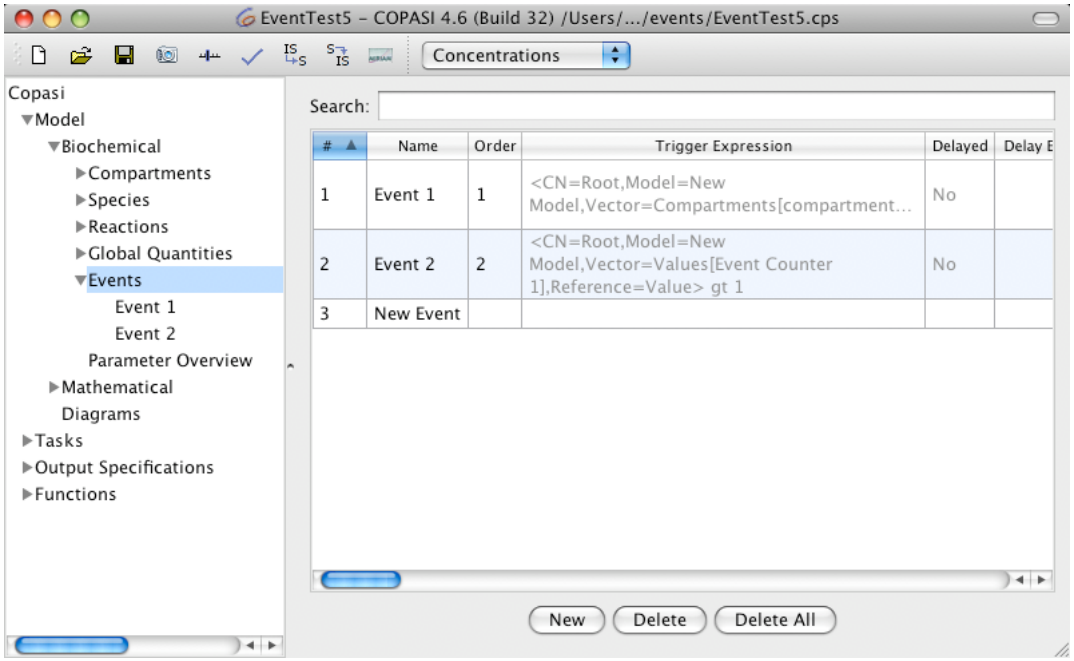
An event assignment has two components an expression and a target. All assignments associated with a single event are executed as one unit. This means further updates and checking for new events is done only after all assignments for a single event are carried out. To carry out an assignment means that the expression is evaluated and its value assigned to the target.

1.9.3 Delay

An event delay, which is optional is the time interval between the firing of an event and the execution of the event assignments. A delay can be inserted in two places:

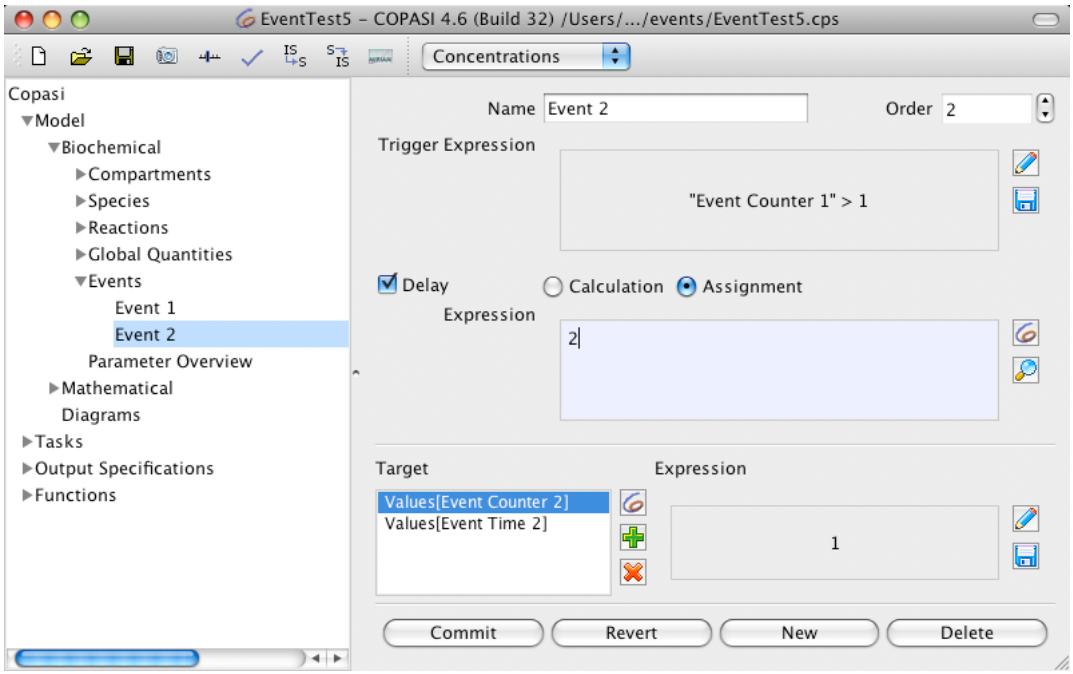
1. between firing and calculation of the target expression (Delay Calculation)
2. between calculation of the target expression and the assignment to the target object (Delay Assignment)

In the model tree right below the Global Quantities branch is the Events branch. If you select this branch, you see a table with all the events that have been defined in your model. When you start a new model this table is empty. The search field allows you to specify a filter for the displayed events. The table will only show events for which the filter expression is matched in any of the columns. This will allow you to search for event names or event targets.



Event Table with 2 Entries

If you click on the name of an event in the tree on the left or double click on a row of the table the detailed information correlated with the chosen event will be displayed.



Detailed Event with 2 Assignments and Delay

In this single event screen you are able to specify the Boolean trigger expression, the optional delay, and multiple event targets. When adding an event target COPASI will allow you to only select values which are not determined by assignments as this would lead to conflicts.

For each of the mathematical expressions, which are the trigger, delay, and assignment expressions, that can be specified for an event you may use the same components, which are used to create function definitions. For a detailed description of these elements see User Defined Functions. You may additionally reference values of other model entities within mathematical expressions, the appropriate values can be selected by pressing the button with the COPASI icon.

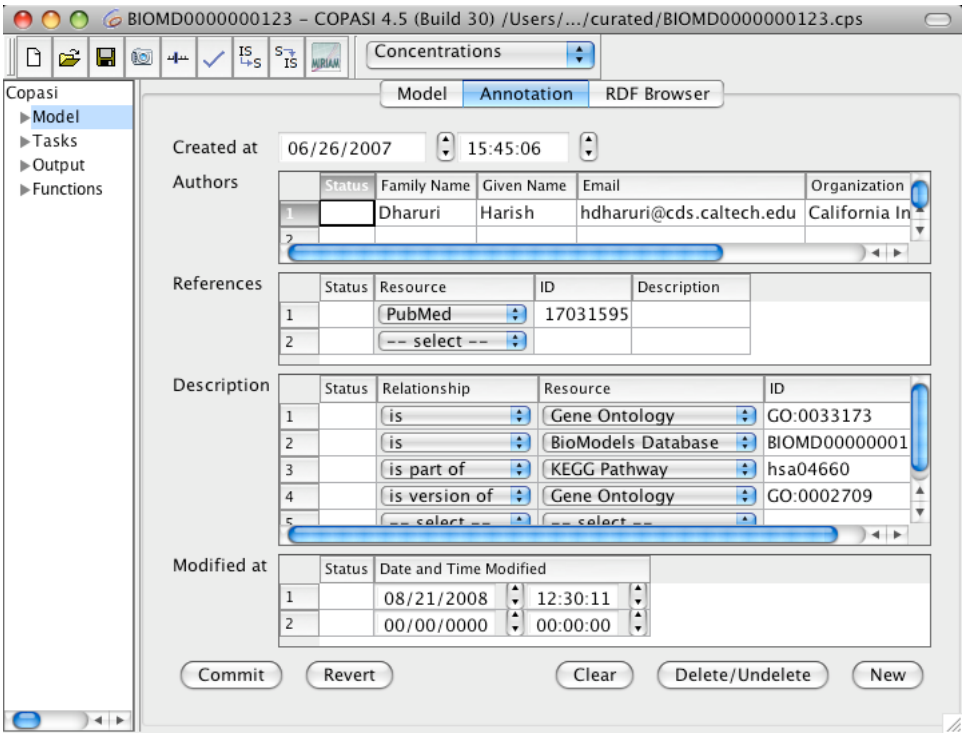
Caution COPASI currently does not support simultaneous assignments caused by different events. We plan to implement this in future releases.

1.10 Annotating Models and Model Elements

Sometimes it would be beneficial to include additional information in a model that describes the model creation process or that gives biological information about the individual model elements. Especially when you get a model from someone or load some published model from the authors Website or some model repository, this information sometimes comes in handy.

In order to standardize the process of model annotation, the MIRIAM (Minimal Information Requested in the Annotation of biochemical Models) guidelines have been developed at the EBI in Hingston. These guidelines specify what information is needed to create properly annotated models that are suitable for publication.

Current versions of COPASI allow the user to annotate the model and all model elements according to the MIRIAM guidelines. For this, the model widget, as well as the widgets for the compartment, the species, the reaction and the global quantities have two additional tabs at the top of the widgets. One tab is called Annotation the other tab is called RDF Browser.



Annotation Widget with Annotation from Model 123 from biomodels.net Database

Before you use the annotation feature for the first time, you have to update the MIRIAM annotation that COPASI uses to create the list of resources that can be associated with an element. You update the list by pressing the the update MIRIAM button in the toolbar. Pressing this button triggers the download of the MIRIAM resources from the internet, so this will only work if you are connected to the internet (once this is done, you should restart COPASI).

COPASI (as well as SBML) use RDF (Ressource Description Framework) to store the annotation in the model file. To view the resulting RDF tree, you can select the RDF Browser tab. This is purely for displaying and not for editing. You will probably not look at this information very often.

If you select the Annotation tab, the same annotation is presented in a more user friendly way. There are different pieces of information that can be given for a certain model element, e.g. the author you added the element or created the model or for species some biological information to the identity of the species.

Each piece of information is stored as a triplet that is composed of the relationship, the resource for that relationship and the corresponding id. The relationship can be chosen from a limited set of relationships. The same goes for the resource. E.g. if you want to specify that species A in the model corresponds to ATP, the relationship would be set to *is* and the resource could for example be the ChEBI (Chemical Entities of Biological Interest) dictionary. The id would then be the ChEBI identifier for ATP, which is CHEBI:A15422.

The COPASI GUI allows the user to store more information than the MIRIAM guidelines for SBML allow, so if you export your model to SBML after annotating it, some information might not be available for other SBML tools.

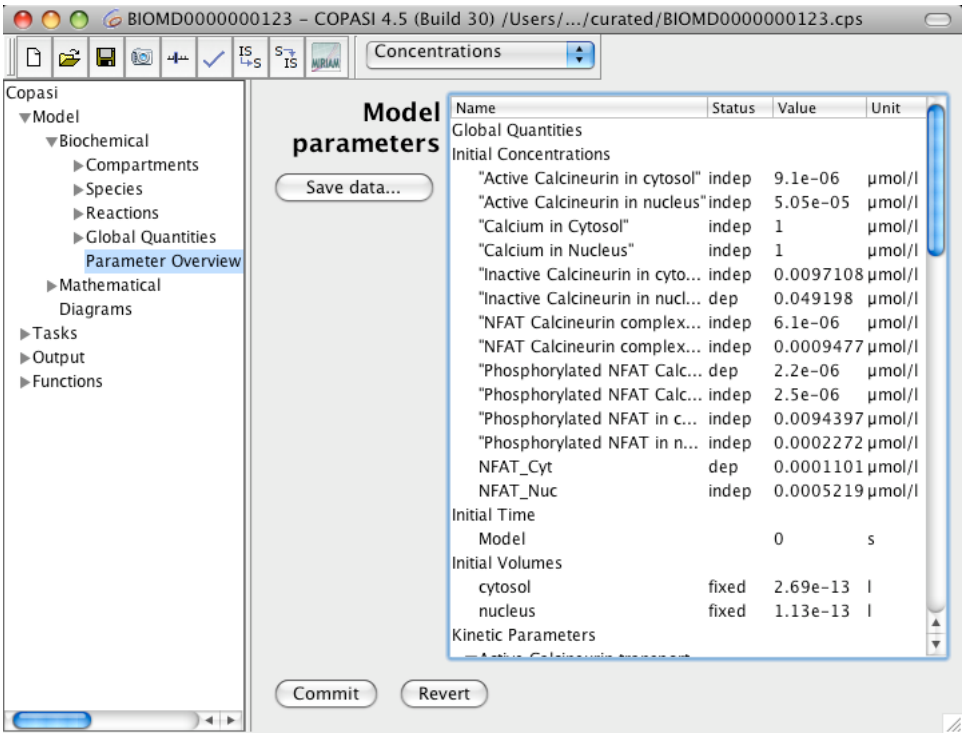
There is also a page on the COPASI Website that has some more technical details about the way annotations are stored in COPASI.

1.11 Parameter View

The parameter view widget can be displayed by selecting the leaf called Parameter Overview on the Model->Biochemical branch (see below). This widget allows you to view and edit all parameters of the model in one place. This saves you from moving around the model tree if you e.g. first have to edit the initial concentrations for some species and afterward parameters of one or more reactions. The view shows you the initial concentrations for the species at the top followed by the initial time and the volumes of all the compartments and at the bottom the kinetic parameters of all reactions.

In order to change a value, you double click on it which lets you input a new value. On hitting the return key or clicking somewhere else, the new value is not written to the object directly, but a '*' character appears in front of the name of the changed

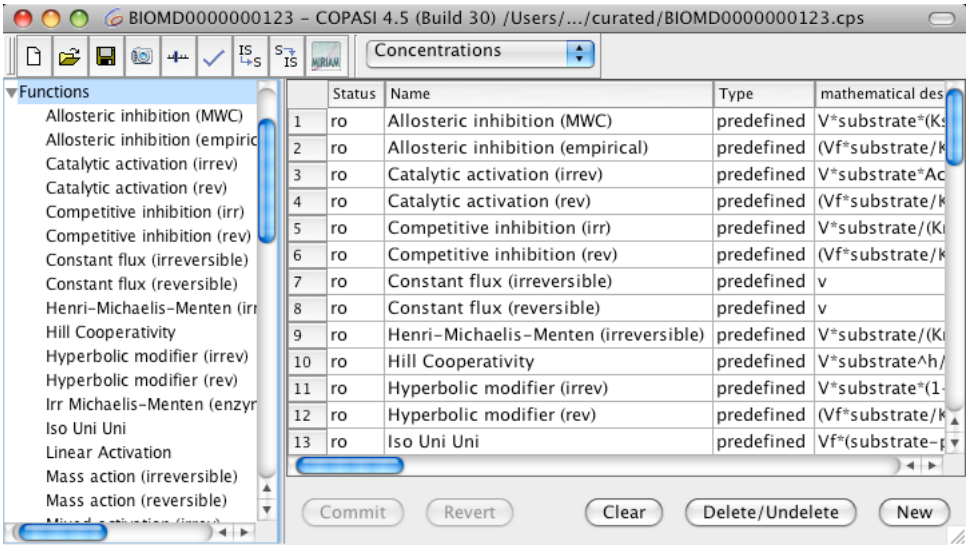
parameter. If you now leave this widget or press the Commit button at the bottom of the dialog, the new value is written to the corresponding object in the model.



Parameter View

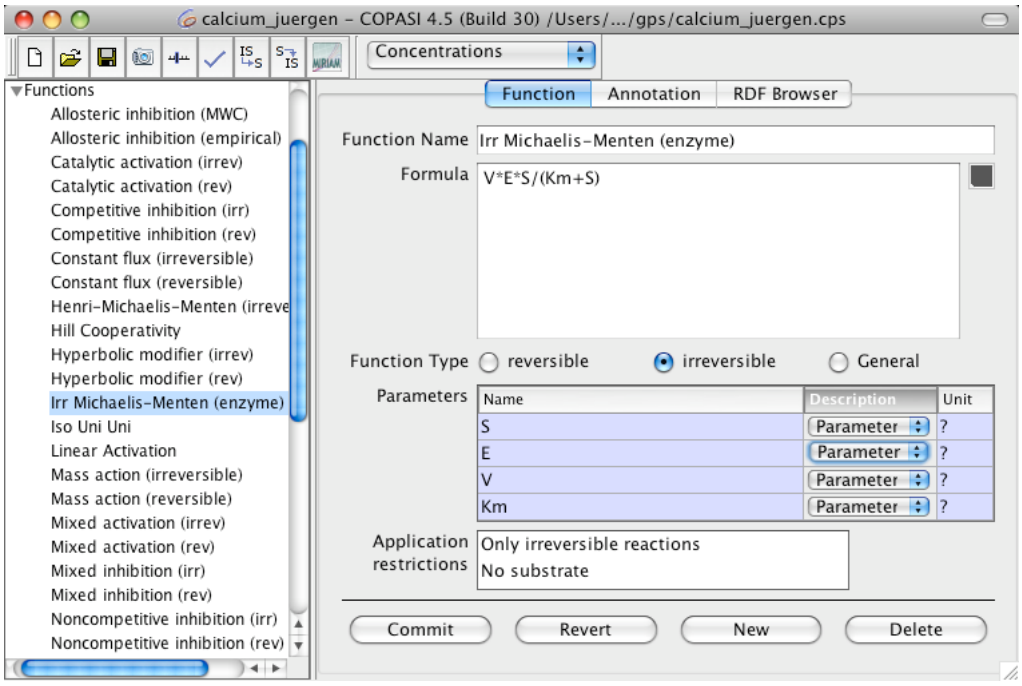
1.12 User Defined Functions

COPASI already defines a large set of commonly used kinetic functions to choose from. The list of defined functions is located at the last branch in the object tree.



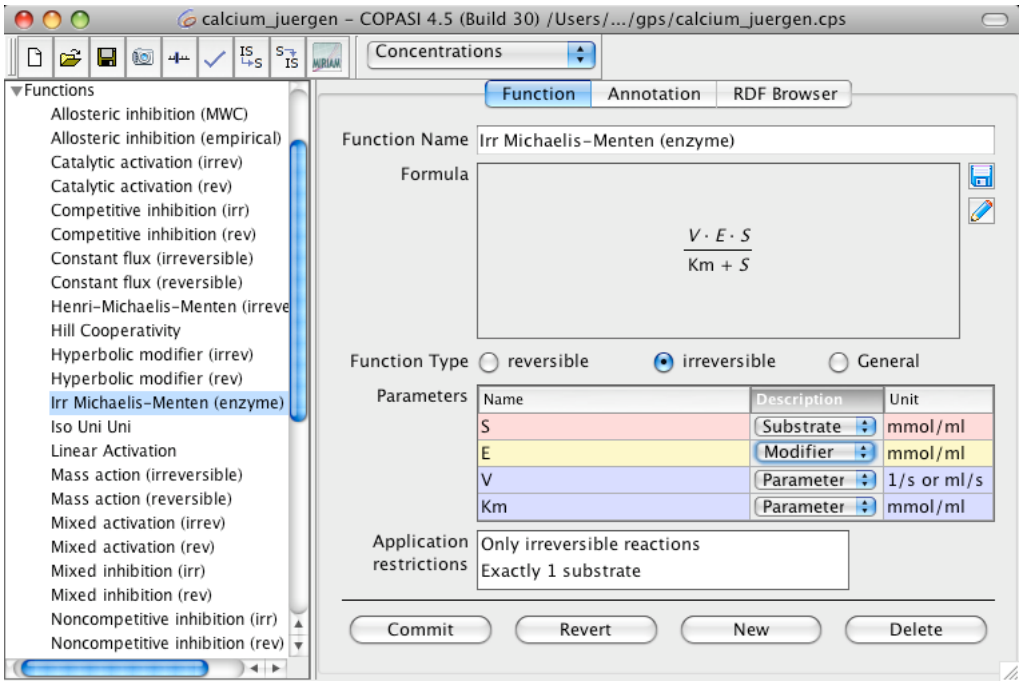
Function Table with predefined Functions

Nevertheless sometimes you need to define your own kinetic function to solve a specific problem. COPASI enables you to add a new function either by double clicking on an empty table row or by clicking on the New button on the bottom of the screen. In the function definition dialog, you give your function a name in the Function Name field. This name has to be unique within the list of defined functions. Next, you have to specify a formula that defines the reaction rate of your kinetic function in the Formula field. The function string only covers the right side of the rate function.



Function Definition Dialog

So for Michaelis-Menten which is defined as v=V*(S/(Km+S)) you would enter v=V*(S/(Km+S)) into the Formula field. While you are typing the formula, COPASI already tries to parse the equation and extract the parameters. All parameters that COPASI finds are listed in the Parameters table.



Function Definition Dialog with graphical Display of the Function

In COPASI parameters can have arbitrary names, there are only a few rules that one has to be aware of. If a parameter starts with a letter or underscore character and contains only letters, digits and underscore characters all is well, if however the parameter name contains other characters, the whole parameter name has to be enclosed in double quotes. If the parameter name contains double quotes or backslashes those have to be escaped by a backslash.

As all variables found are defined, per default, as *Parameters* , you should specify their correct types by selecting one from the drop down list Description.

However, the specific type of variables will affect to the type of reactions the function can be used for. E.g, if you define the function to contain two substrates and a modifier, you can later only use it for reactions that really do have two substrates.

Caution The restrictions on the number of modifiers is not strict since substrates and reactants could act as modifiers. So the above mentioned rate law could be used on reactions that do not explicitly specify a modifier.

You can also see this in the Application restrictions table below the Parameters table. Let's say you define the function A*B and define *A* and *B* to be substrates, you will see that the Application restrictions say that there must be exactly two substrates in the reaction for that kinetics to be applicable. After defining this function, you will be able to use it for all chemical reaction that have exactly two substrates. Last but not least, you have to define whether this function can be applied to reversible, irreversible or both reaction types by selecting the reversible, irreversible or General radio button respectively. You can also call other functions from function definitions. There are four things you have to watch out for when you call a function within another function.

1. Recursive function calls are not permitted. That is a function may not call itself, neither directly nor by calling another function that might call the first function again further along the line.
2. You have to specify the correct number of arguments to the function called.
3. You have to specify the correct argument types to the function call. I.e. if you call "Henry-Michaelis-Menten (irreversible)" from within another function, you have to make sure that the first call argument has a usage of *Substrate* and the other two have the usage *Parameter*.
4. Which brings me to the last point. The built in function names in COPASI often use characters like "-" or even spaces, so if you want to call one of those functions, you have to quote this function name. So calling "Henry-Michaelis-Menten (irreversible)" from another function would look like this:

```
"Henry-Michaelis-Menten (irreversible)"(S, Km, V)
```

After you commit the function, you can use it for the definition of reactions.

The operators and functions that COPASI knows and therefore can be used to create user defined functions are the following:

1.12.1 Standard Operators

Operator/Function	Description
+	plus operator
-	minus operator
/	division operator
*	multiplication operator
%	modulus operator
^	power operator

Standard Operators

1.12.2 Miscellaneaous Functions

Operator/Function	Description
abs / ABS	absolute value
floor / FLOOR	floor value
ceil / CEIL	next highest integer
factorial / FACTORIAL	factorial function
log / LOG	natural logarithm
log10 / LOG10	logarithm for base 10
exp / EXP	exponent function

Miscellaneaous Functions

1.12.3 Trigonometric Functions

Operator/Function	Description
sin / SIN	sine function
cos / COS	cosine function
tan / TAN	tangent function
sec / SEC	secand function
csc / CSC	cosecand function
cot / COT	cotangent function
sinh / SINH	hyperbolic sine function
cosh / COSH	hyperbolic cosine function
tanh / TANH	hyperbolic tangent function
sech / SECH	hyperbolic secand function
csch / CSCH	hyperbolic cosecand function
coth / COTH	hyperbolic cotangent function
asin / ASIN	arcsine function
acos / ACOS	arccosine function
atan / ATAN	arctangent function
arcsec / ARCSEC	arcsecand function
arccsc / ARCCSC	arccosecand function
arccot / ARCCOT	arccotangent function
arcsinh / ARCSINH	hyperbolic arcsine function
arccosh / ARCCOSH	hyperbolic arccosine function
arctanh / ARCTANH	hyperbolic arctangent function
arcsech / ARCSECH	hyperbolic arcsecand function
arccsch / ARCCSCH	hyperbolic arccosecand function
arccoth / ARCCOTH	hyperbolic arccotangent function

Trigonometric Functions

1.12.4 Random Distribuitions

Operator/Function	Description
uniform/UNIFORM	This functions takes 2 arguments min and max. It returns a normally distributed value in the open interval (min, max).
normal/NORMAL	This function takes 2 arguments mean and standard deviation. It returns a uniform distributed value with the given mean ad standard deviation.

Random Distributions

1.12.5 Logical Operators

The logical operators and comparisons are evaluated in the order they are listed in the table.

Operator/Function	Description
le / LE	smaller or equal (<=)
lt / LT	smaller (<)
ge / GE	greater or equal (>=)
gt / GT	greater (>)
ne / NE	not equal (!=)
eq / EQ	equal (=)
and / AND	logical and (&)
or / OR	logical or (|)
xor / XOR	logical xor
not / NOT	logical negation

Logical Operators

1.12.6 Conditional Statement

In addition to defining "normal" functions, COPASI allows the definition of piecewise defined functions. Piecewise defined functions are constructed with the IF statement.

Operator/Function	Description
if()/IF()	if statement for the construction of piecewise defined functions etc.

Conditional Statements

The functions name can be written with either all lowercase letters or all letters uppercase. Mixing of upper and lowercase letters is not allowed and will lead to errors. This function takes 3 arguments separated by a comma:

1. Boolean expression
2. Expression evaluated if the first argument evaluates to *true*.
3. Expression evaluated if the first argument evaluates to *false*.

So in order to make this a little more clear, we will look at how one would implement the Heaviside step function in COPASI:

```
if(x lt 0.0, 0.0, if(x gt 0.0, 1.0, 0.5))
```

Caution Although COPASI allows the usage of discontinuous functions (ceil, floor, factorial, etc) all integration is done by LSODA which officially can not handle discontinuous functions. Nevertheless in most cases this will lead to correct results, however you should be aware of the fact that the usage of discontinuous functions in COPASI can lead to errors. Later versions of COPASI will use different integration methods that will be able to deal with discontinuous functions.

In addition to the function above, COPASI knows some predefined constant names:

1.12.7 Parenthesis

Operator/Function	Description
()	parenthesis for grouping of elements

Parenthesis

1.12.8 Built-in Constants

Operator/Function	Description
pi / PI	Quotient of a circles circumference and its diameter (3.14159...)
exponentiale / EXPONENTIALE	Euler's number (2.7183...)
true / TRUE	Boolean true value for conditional expressions
false / FALSE	Boolean false value for conditional expressions
infinity / INFINITY	Positive infinity

Built-in Constants

Again, built-in constant names can be written with either all lowercase letters or all letters uppercase. Mixing of upper and lowercase letters is not allowed and will lead to errors.

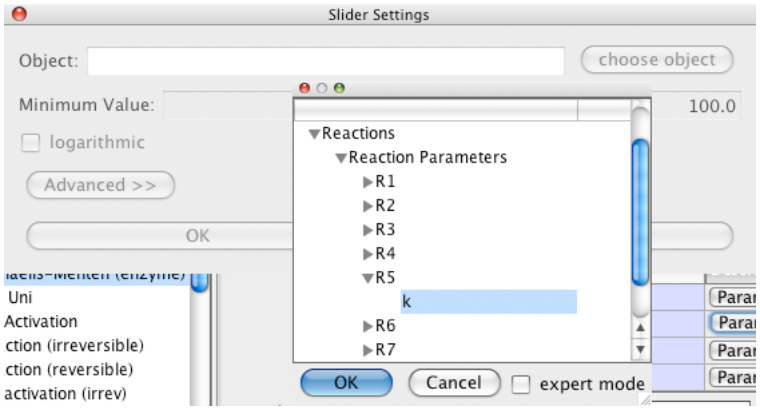
1.13 Sliders

Sliders are user interface elements that let the user change model values without having to go to the corresponding dialog first. This way the user can change certain model values and immediately see the result this has on e.g. a time course simulation. The use of sliders is not limited to the Time Course task, but they can also be used for the Steady-State task and the Metabolic Control Analysis.

Per default the slider dialog is hidden, if you want it to be displayed, you have to activate it by toggling the corresponding menu entry in the Tools menu or through the slider button in the tool-bar. Depending on what element is selected in the object tree, the dialog will be disabled and a text will be displayed within the dialog that tells you so. If you select a task (or an element below the task) that supports sliders, the sliders dialog will be enabled.

Empty Slider Dialog

Once the slider dialog has been activated, you can add new sliders with the new slider button at the top of the slider dialog. To edit the parameters of an already defined slider, just click on the edit button, which is located on the right side of the slider towards the bottom. If you no longer need a slider you have defined, you can close it by clicking on the remove button which is the top button on the right side of the slider. Alternatively you can add or modify the existing sliders via a context menu. The context menu can be activated with the right mouse button (CTRL + Mouse button for single button mice on Macs!). If you right click on an existing slider, the menu will offer the possibilities to remove this slider or to edit it. If you right click into the dialog window where there is no slider, you will be offered the possibility to define a new slider.

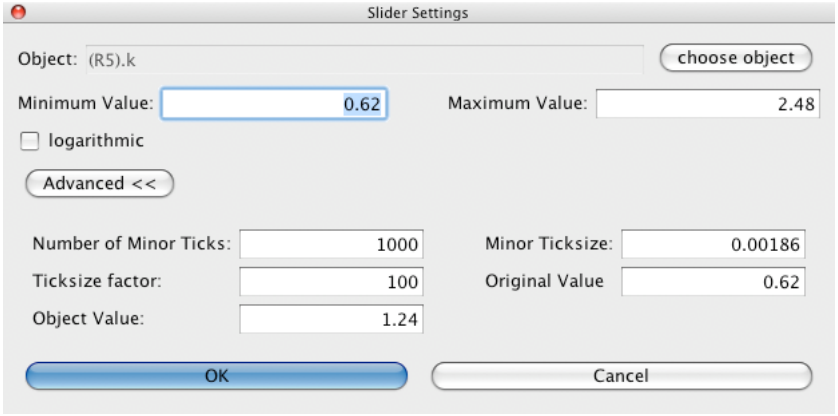


Basic Slider Definition Dialog

Once you are in the dialog for the definition of a new slider (see above), you have to choose an object to manipulate via the slider. This object can be selected from the selection dialog that pops up when you click on the choose object button. The selection tree is the same as the ones you get when selecting objects for a report definition. The selection dialog also offers an expert mode in case the object you want to manipulate with the slider is not present in the simplified tree. Usually the objects that you want to modify in the time course simulation are one or several of the reaction parameters. This way you can interactively see how the behavior of your model changes when you change a specific parameter. In combination with plots this is a very powerful way to examine the behavior of your model.

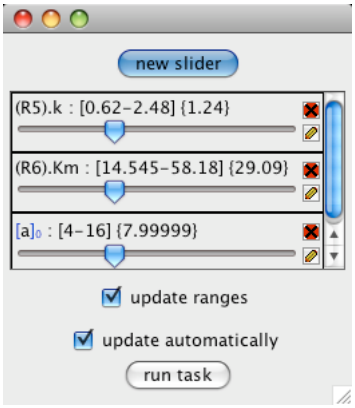
After selecting an object, you can set the parameters for the slider. You specify the range the slider has to cover in the Minimum Value and Maximum Value fields. Additionally you can specify if the slider shall have a logarithmic or a linear scale. Per default sliders have a linear scale. Normally a slider will be available for all tasks that support sliders. An exception to this are slider for objects that are directly associated with a specific task, e.g. the number of steps in a time series, if you choose such an object for a slider, the slider will only be available for this task.

At the bottom of the dialog there is a button labeled Advanced. If you click on this button more options that influence the sliders behavior will show up (see above). Clicking the button a second time will hide these options again. The advanced options contain mostly fields that let you modify how many steps the slider has (see explanation below). The Object Value field determines the current value of the object that is associated with the slider. In the Number of Minor Ticks field you specify how many minor ticks your slider will have. And in the Tick size factor field you specify how many minor ticks make a major tick. Major ticks can be used to coarsely go through the range of the slider whereas minor ticks allow you to step through the range in a more fine grained fashion. Instead of specifying the number of minor ticks, you can specify the size of a minor tick. If you change either of those two values, the other one will be adjusted accordingly. The formula is: $\text{minor tick size} = \frac{\text{Maximum Value} - \text{Minimum Value}}{\text{number of minor ticks}}$.



Extended Slider Definition Dialog

Once you have made all the settings, you confirm the slider definition with the OK button. A new slider will appear in the slider dialog.



Slider Dialog with 3 Sliders

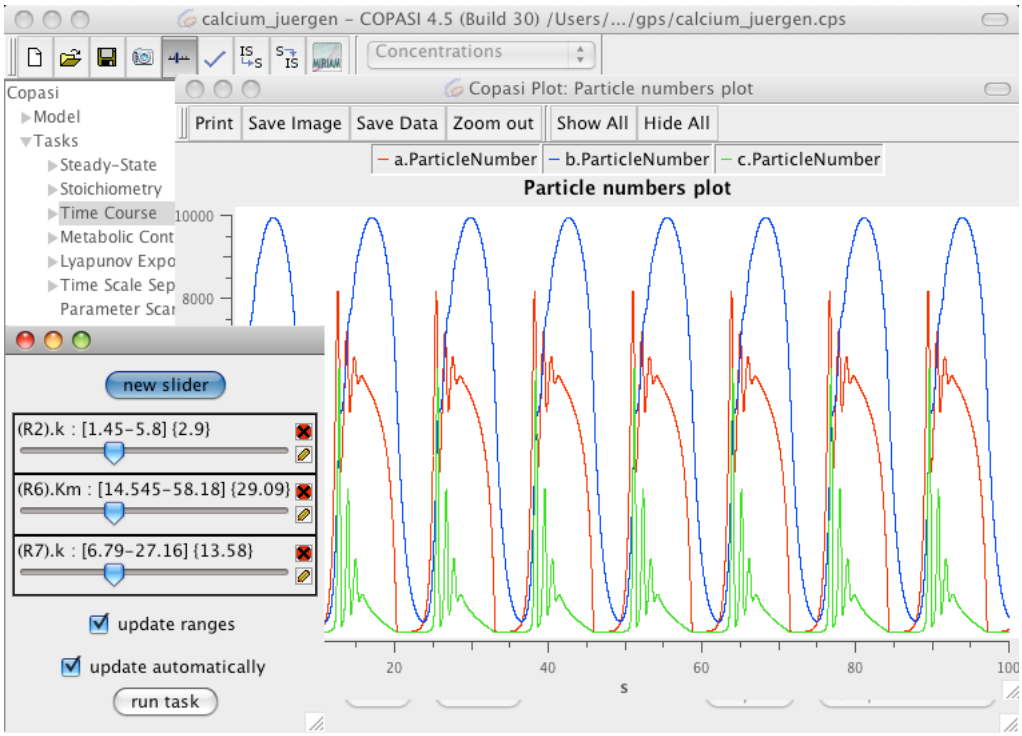
Each slider shows the name of the object, the current value and the range in its label (see above). If the current value of the slider is outside of the defined range, a corresponding warning will be shown. The slider dialog has a check box called update automatically if this box is checked, COPASI will run the corresponding task each time you release a slider handle after moving it. If you don't want COPASI to automatically run the task each time you change a sliders value, you can uncheck this box and run the task manually by clicking on the run task button.

If the value of an object for which a slider has been defined changes when a task is run, the slider will automatically show the new value. If the check box called update ranges is checked, the sliders will automatically adjust their range if the value falls outside of the defined range. If the new value is larger than the old maximum value, the new maximum will be set to twice the new value. Likewise if the new value falls below the old minimum value, the new minimum will be set to half the current value.

Modifying a slider is essentially the same as defining a slider. The only difference being that you can not change the object the slider is connected to. In order to do that, you have to delete the slider and define a new slider for the new object.

Most people will probably use the sliders in combination with the mouse, dragging the sliders pointer to the desired new value. Since this way of using a slider is rather coarse, especially if the slider contains a large number of steps, you can also use the keyboard to change a sliders value. In order to do this, the slider you want to manipulate needs to have the keyboard focus. To give the keyboard focus, you have to hit the TAB key several times until the slider is surrounded by a small frame. Once the slider has this frame, you can increment and decrement the sliders value via the cursor and the page up/down keys on the keyboard.

The cursor keys are used to change the value in small steps, the page up/down keys can be used to change the value in larger steps (see the explanation for major and minor keys above).



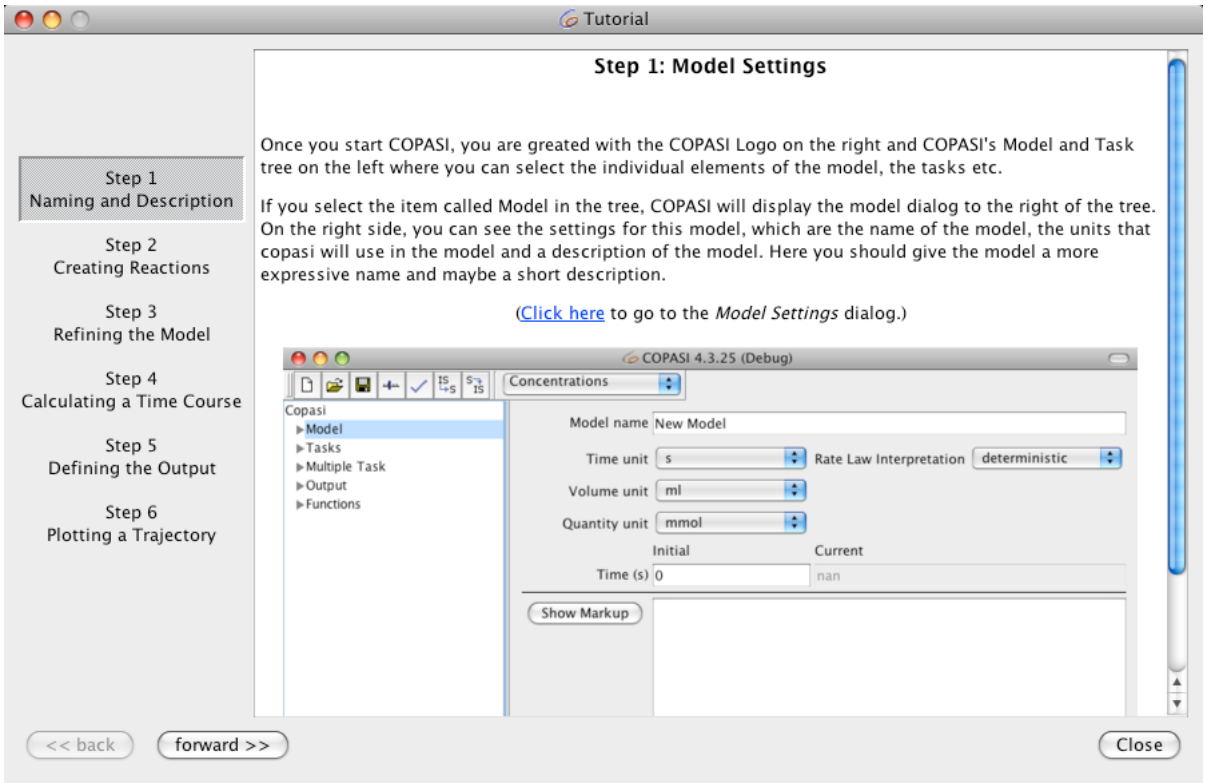
COPASI with Plot Window and Sliders

1.14 Tutorial Wizard

COPASI also includes a short tutorial on how you create a model and run a time course simulation with plotting. You can start the wizard from the help menu. The corresponding menu entry is called Simple Wizard.

On the left side of the wizard there are six buttons that correspond to the six steps. When you first open the wizard, step one is selected and the widget on the right shows some documentation that you should read and possibly repeat the steps explained. Once you are finished with step 1, you can either click directly on the button that corresponds to step 2, or you can click the forward button which will always bring you to the next step. If you would like to reread something from an earlier step, you can move to the specific step with the buttons or use the forward and back buttons to navigate.

The wizard leads you in six steps from creating a model to doing a time course simulation of the model and plotting the results of the time course.



Tutorial Wizard

Chapter 2

Output

After you have defined all reactions that make up your model, you could go ahead and do some calculations with your model. For some of the task this is fine since they show you the results in a separate results dialog and it might not be necessary to store those results into a file. For other tasks, like e.g. a time course, it is very likely that you want to store the resulting trajectory in a file or plot the result for visual inspection. So if you want COPASI to store the results or do a plot of them, you have to define either a report and associate it with a file or define a plot. (In the section we will also offer an alternative, albeit less flexible, way for storing the results of a time course simulation.)

COPASI already has a list of predefined reports suitable for some of the calculation task. These are described in the next paragraph. If you want to create your own plots or reports you can either use the output assistant (the easy way) or you can define the output manually (the flexible way), as described in the section.

2.1 Predefined Reports

For some of the calculation tasks predefined reports exist in COPASI. They are automatically created when COPASI is started or when a file is loaded that does not already contain them. These reports have the same name as the task they are suitable for. E.g. the predefined report for Steady-State calculations is *Steady-State*. The predefined reports print a description of the settings you provided for the calculation, usually (if it applies for the specific calculation) a table with intermediate results, and in the end a detailed report of the result. The report definitions can be changed or deleted using the mechanism described in the section. If you changed one of the predefined reports and want to go back to the original, just delete the report definition and save the file. When you load it again the current missing default report will be generated automatically.

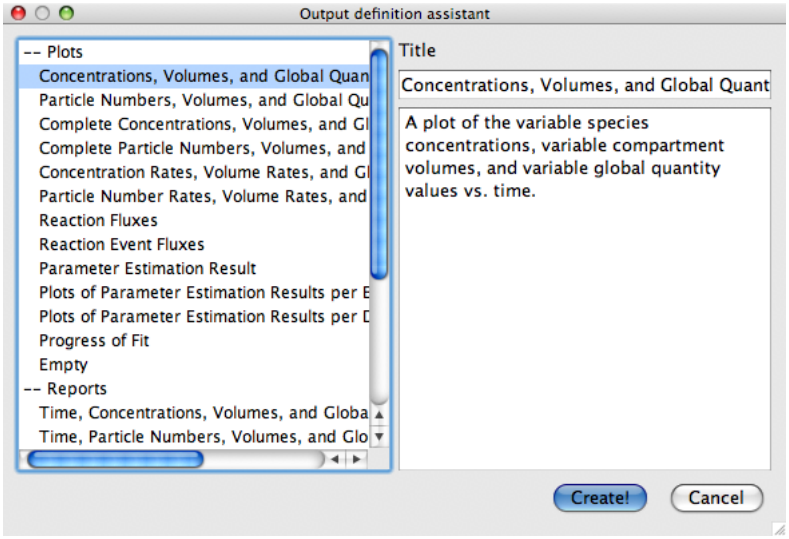
If you want to use the predefined reports you have to make sure that the report is selected for a task and that a file name is associated with the report. For this you can use the Report button as described in the sections about the specific calculation tasks.

2.2 Output Assistant

The output assistant presents the easiest way to generate your own output definitions which you can later adapt to your wishes using the techniques described in Manual Definition. Almost all task dialogs in COPASI have a button at the lower right that is labeled Output Assistant. If you click on this button, a new dialog will open with a list of predefined output definitions on the left. If you select one of the output definitions from the list, you will get a short description of what the output does on the right side of the dialog. Above the description is the title of the output definition. This title can be changed in order to be able to identify the different output definitions in case you are planning on creating more than one output definition of a certain type. Using this dialog both plots and reports can be created.

Creating an instance of the selected output objects is as easy as clicking on Create! at the bottom of the dialog. Once you clicked this button, a new report or plot, depending on what you selected in the dialog, will appear in the corresponding branch of the Output section of the model tree. The Output branch is the second to last branch in the tree on the left. The name of the output definitions will be the title of the object which you selected from the list. If another output definition in this section already has the same name, a postfix will be appended to the name. The so created output can now be edited or deleted. If the newly created

output is a report it will automatically be selected as the active report for the current task. You then still have to select a filename for the output using the Report button. This is described in the sections about the specific calculation tasks below. How output definitions are created, edit and deleted manually is the topic of the next sections.

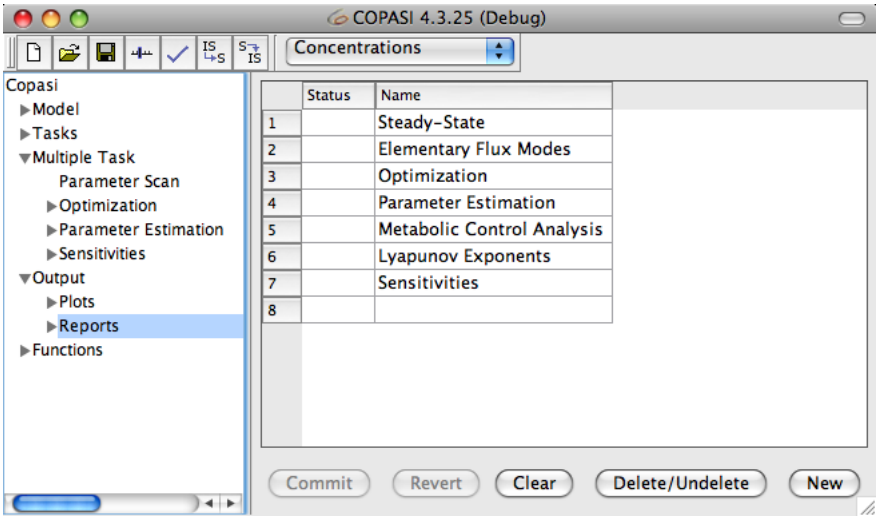


Output Assistant Window

2.3 Manual Definition

2.3.1 Reports

This section describes how to create or edit a report definition. Keep in mind that you still have to select this report for the specific calculation task you want to perform. You can do this (and also choose a filename to write to) using the Report button as described below in the sections about the different tasks.

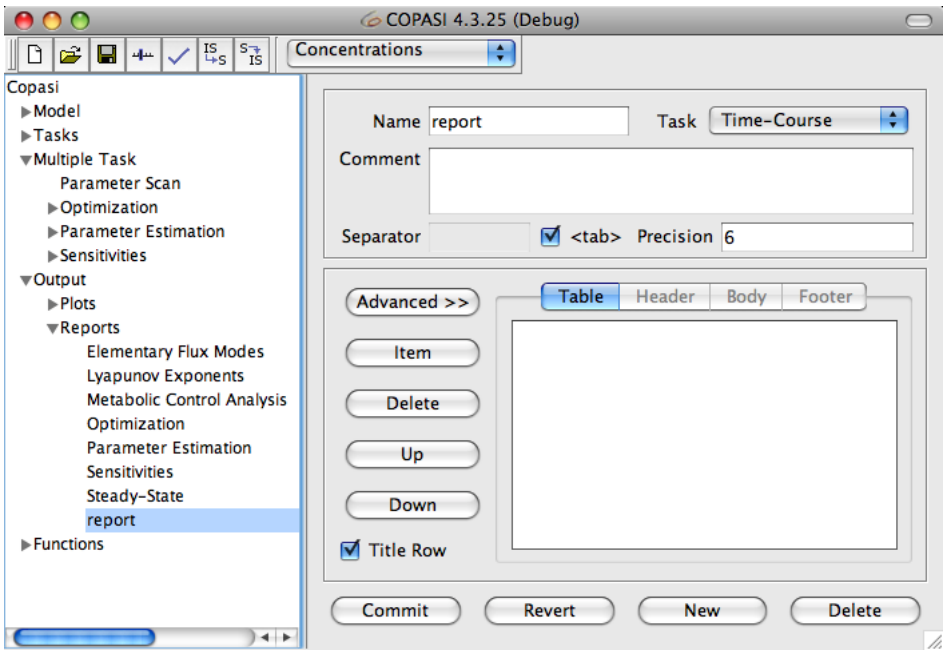


Report Table with default Reports

The dialog for defining report definitions is located under the Output->Reports branch in the object tree. Double clicking on an empty row in the table creates a new report object and opens the dialog for modifying the report definition. In this dialog, you can specify a name for the report in the Name field. From the Task drop down list, you can choose for which kind of task this report should be written. So if you want to store the result of a time course, you should choose *Time-Course* here. The report usually stores the results of its task as a table; the standard separator character for elements in this table is the tab character (\t). If you want to have another character as the separator field, you have to uncheck the <tab> check box and specify the separator character or string you want in the Separator field.

Beside the <tab> check box there is another input field labeled Precision. With this field you can specify how many significant digits are used for numerical output. The default value is 6.

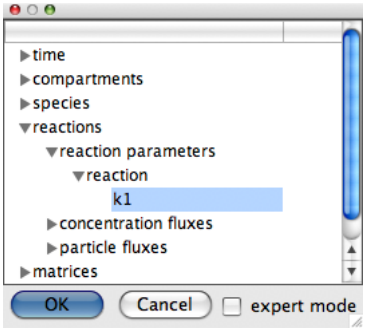
If you want to, you can also add a comment for the report, e.g. you could state what this report is supposed to represent.



Simple Object Selection Dialog

Next you have to define the objects that you want to appear in the report. There are two modes to define a report definition. Per default, the report is laid out as a table. For example, the report for a time course simulation will write one line per time step and each line will typically contain the time followed by one or more species concentrations. In the advanced mode activated by clicking on the button labeled Advanced, the report is split up into three sections, a header, a body and a footer. You can define the output for each of the sections separately by clicking on the corresponding tab. In order to get back to the standard table layout, you have to click on the Advanced again. COPASI will warn you that you might loose some information by converting from the advanced report definition format to the table format.

Let us discuss how to define a report definition with table layout first since you will probably use this most of the time. To add a new object to the report definition, you have to click on the button labeled Item. This will open the object browser dialog.



Object Browser Dialog Expert Mode

The selection dialog shows a tree that contains what we think are the objects that would be most commonly used in generating report, plots, sliders etc. You select objects by clicking on the corresponding leave in the selection dialogs tree view. For plots and report, the simple selection dialog will allow you to select several objects at once. To select a continuous range of objects, you select the first object of the range, then you press and hold the SHIFT button and select the last object in the range. You can also make non-continuous selection by holding down the CTRL key while clicking on the object to select/deselect. Also selecting a whole branch in the selection dialog will select all the leaves under that branch.

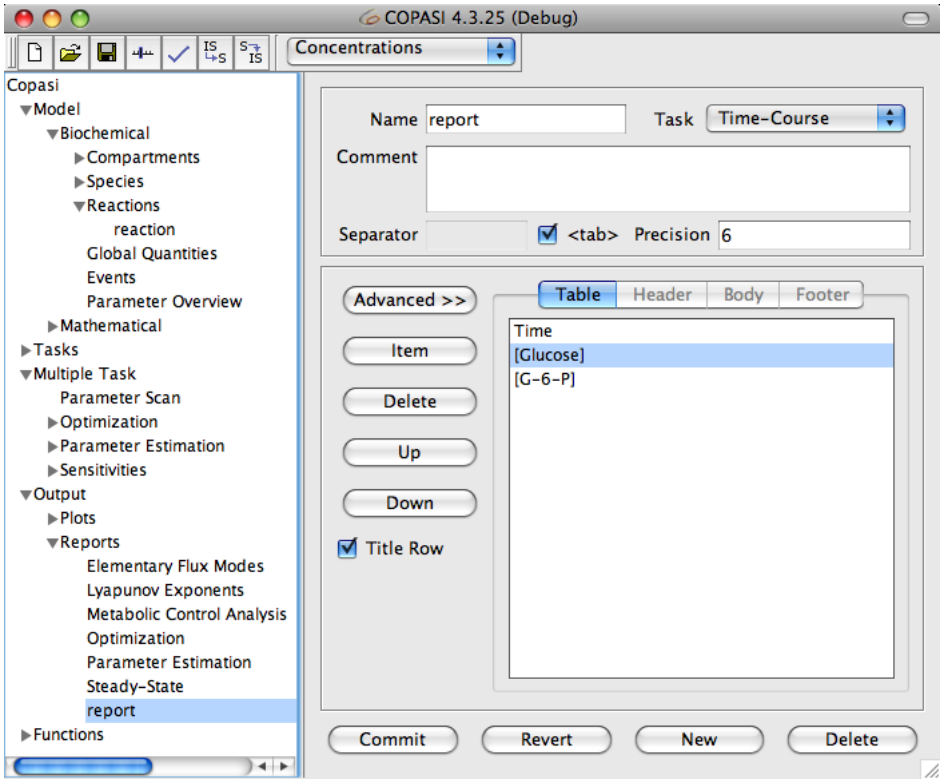
If this simple selection dialog does not contain the object that you want to include in your report, you can activate an extended selection tree by activating the expert mode check box. The tree you now see contains all objects COPASI knows about. In this tree, the objects belonging to your model are located in the branch that corresponds to the name of your model. The position of

that branch varies since the branches are sorted alphabetically. The selection that you have already made in the simple selection tree should be preserved, and vice versa. Any selection you make in the full tree is preserved when you switch back to the simple tree. Each branch of the full tree has a check box up front which can assume three states. The unchecked state means that no objects in this subtree are selected. A check mark on a black background means that the whole subtree is selected, i.e. all objects in this subtree are selected. A check mark on a gray background means that part of the subtree is selected.

Due to the model structure, most objects appear more than once in the tree.

So do not be surprised if you select some objects you may be see that not only the selected objects will suddenly change their selection state. E.g. if you select the whole Compartments subtree of your model, all the species which are part of the compartments get selected as well, which means that on selecting the Compartments branch, the whole Species branch changes its state to be selected.

Let us assume you want to define the report for a trajectory task in expert mode. In this case, you will probably want the time and some or all of the transient concentrations of the species in your report. The time for the time course is the last item in the Model branch, you select it by clicking on the check box in front of the name. If you want to add the concentrations of all species, you open the Species sub-branch in the Model branch and open the Select by attribute branch. There you can select the Concentration attribute. Selecting the Concentration attribute will select the concentrations for all species. If you only want to have some of the species in your report, you open the sub-branches of the wanted species in the Species branch and select the Concentration attribute only for those species. If your model contains many species and you want to have all except one species in your report, it is often easier to first select all concentrations via the Select by attribute branch and then deselect the one you don't want, rather then selecting the individual concentrations you want. Once you are finished with selecting the objects for your report, you confirm the selection by clicking the OK button in the selection dialog. The objects you selected will now appear in the list box of the report definition dialog.



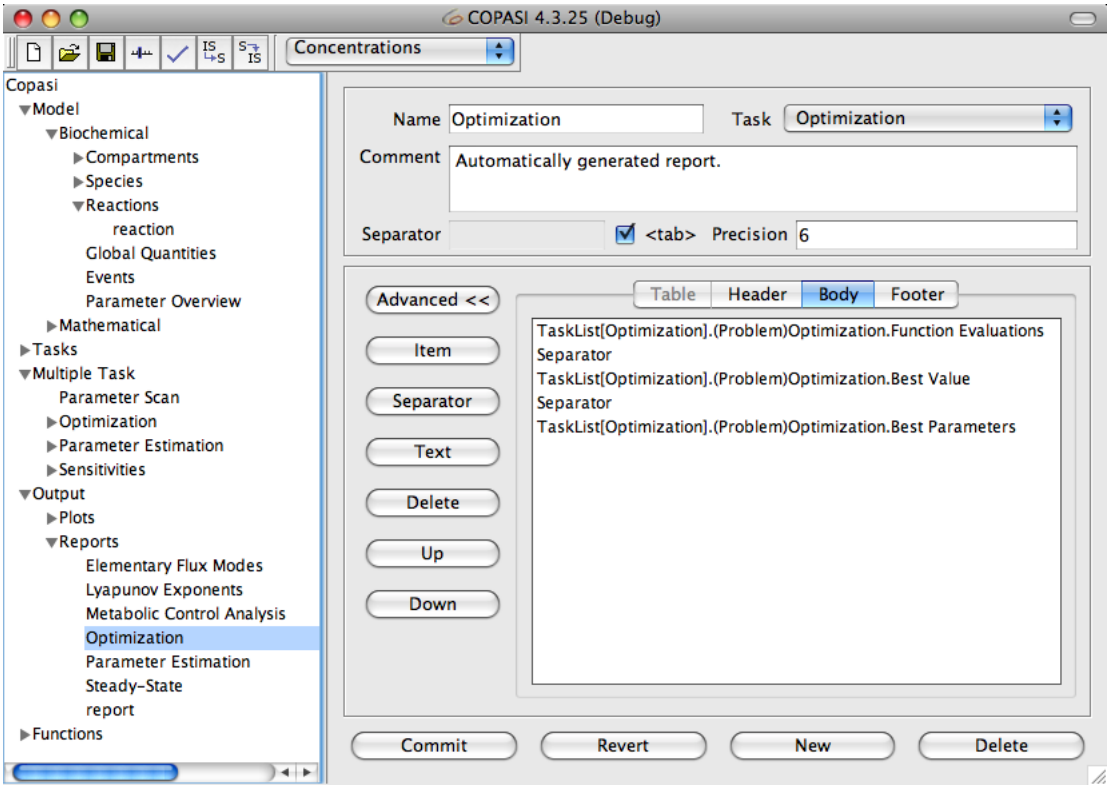
Report Definition Dialog

They will appear in the report in the same order as they appear in this list. To reorder the entries in the list, you can select individual entries and move them up and down in the list with the corresponding buttons on the left of the list. For example it might be a good idea to move the time object to the top of the list so that it will appear as the first table column in the file since this is the way most programs would expect it. Also in order to delete unnecessary items, you just select them and click on the Delete button. The only thing that is now left is to connect this report to a file. This has to be done in the dialog for the specific task and we will cover this when we explain how to run the individual tasks.

The report we just defined will be written in the form of a table. So if this is a report for a time course simulation, the final output would have one line per time step of the simulation and in each line, each object that is in the list would be written once separated

by whatever you defined as your separator character (Normally this would be the <tab> character).

Per default, the check box labeled Title Row is activated, which means that COPASI will write a header line before the table with the names of the objects that make up the individual columns. If you don't want such a header in the output you have to deactivate this check box.



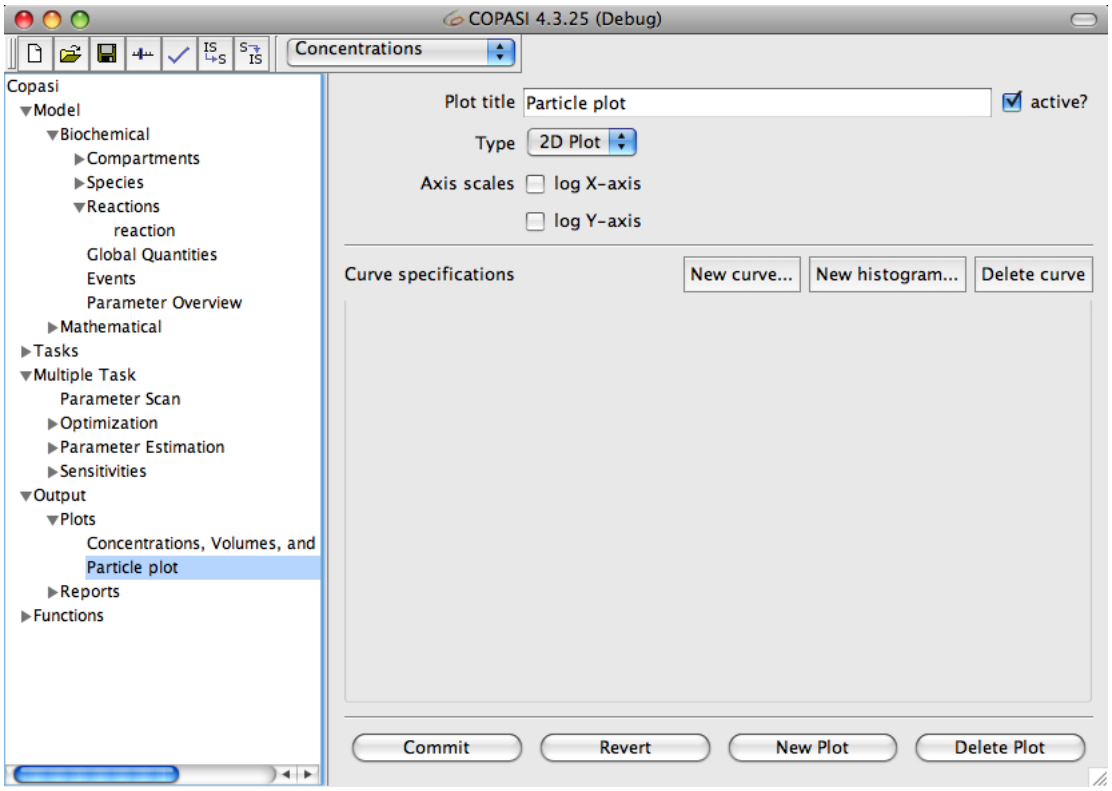
Advanced Report Definition Dialog

As stated above, the advanced report definition allows you to define the output for the three parts of the report separately. The *header* part of the report is written once before the corresponding task, the output of the *body* elements occurs once per step of the corresponding task, e.g. for the time course once every time step, and the *footer* is written once after the task has finished. In this sense, the standard table report definition is nothing but an advanced report definition with a title header and a body that consists of the time and some species concentrations separated by separator items. The footer is empty. If you write your own advanced report definition, you are responsible to add separator tags where appropriate. An advantage to the advanced report definition is that you can add arbitrary text to any of the three sections. Everything else works as described for the standard report definition, you can add and delete items by selecting them from the object browser dialog. You can also move these items up and down by selecting them and clicking on the Up or Down button. Separator items can be added by clicking on the Separator button. The symbol or text that makes up the separator item is again defined by the check box and the adjacent input field towards the top of the report definition dialog.

2.3.2 Plots

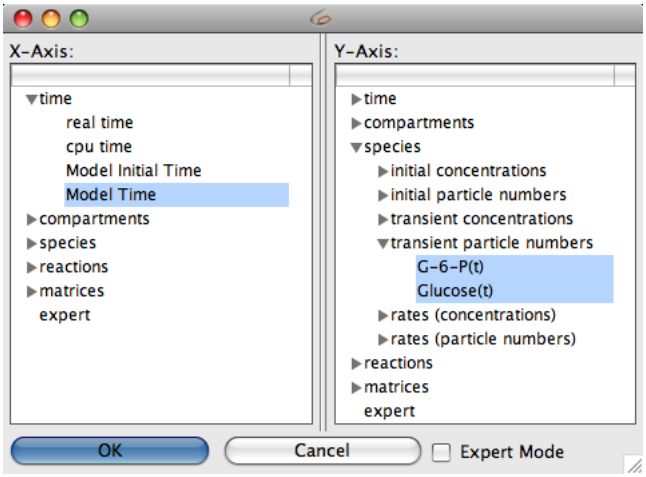
Plotting is another form of output that COPASI can do. Most of the time, you probably want to plot some or all of the species concentrations during a time course simulation and, as described above, this is easiest done by choosing that predefined plot template from the Output Assistant. But since the output assistant can not cover all possible plots sometimes you will have to define your own plots.

Currently, COPASI only supports two dimensional plotting. To define a plot, you open the plot definition dialog by selecting the Output->Plots branch in the object tree. A plot in COPASI is made up of a number of curve or histogram objects. In order to add a new curve object you click on the New curve... button.

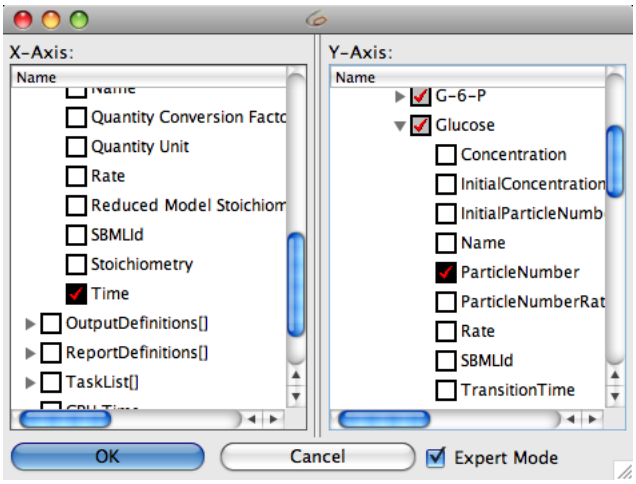


Empty Plot Widget

A selection dialog similar to the one described for the creation of reports will appear. The major difference is that you now have two tree views side by side instead of only one. The left tree is a single selection dialog that lets you specify the object to be drawn along the x-axis. For a plot of concentration against time this would be the simulation time. The right tree is, again, a multi-selection tree that lets you specify what would be drawn along the y-axis. In case of plotting concentrations against time, this would be one or more concentrations of species. Just as described for the report definitions, you can switch between a simple tree and the full tree, but for most plots the simple tree will be enough.



Selection Dialog with some Items selected



Expert Selection Dialog for Curve Objects

Once you have finished your selection, you click on the OK button which will take you back to the curve definition dialog. For each object selected in the right tree, you will now see one tab which represent the corresponding curve object for the plot. To remove one of the curves, select the tab that corresponds to the curve you want to remove and then click on the Delete curve button. The next time you do a time course simulation, each plot that is marked as active will be plotted automatically. How you define a plot as being active will be explained below.

Each curve object has a title, and the information what will be drawn on the x- and y-axis. Additionally you can specify whether the curve should be drawn as a line, as points or as symbols. This has to be specified for each curve separately. You can also specify if the plot should be drawn with a logarithmic scale for one or both of the axis.

In the plot definition dialog, in addition to adding and removing curve objects, you can give a name for the plot definition and specify whether the plot should be active or not with the active check box. (Only plots that are active are drawn when a time course simulation is run!)

Another way to specify whether a plot is active or not is in the plot table where all the plots are listed. Each row in the table contains a column named active that contains a check box with which you can toggle the state of a plot. If you changed the state of one or more plots, you have to commit these changes either by clicking the Commit button or any other action that is equivalent to pressing the Commit button (see compartments section).

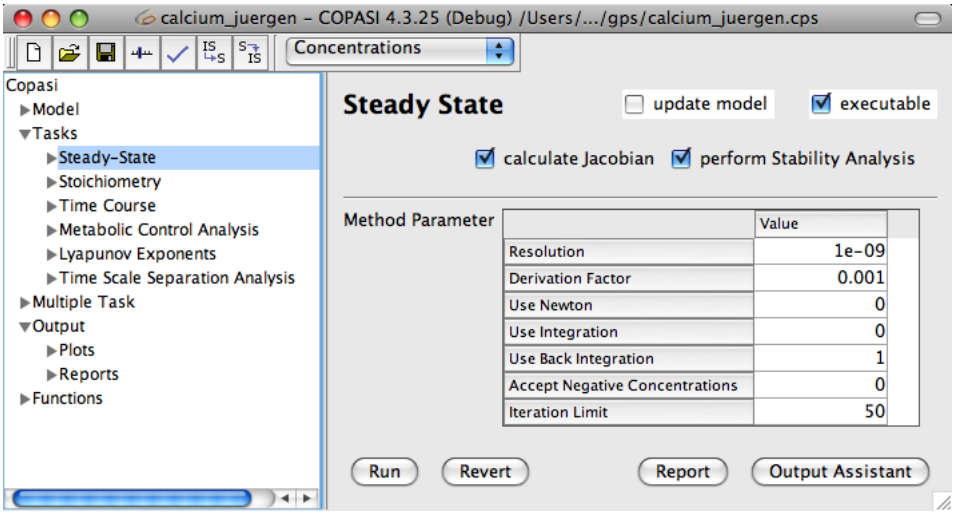
In addition to curves, you can also tell COPASI to draw a histogram of the data generated during a time course simulation (see time course simulation) or a parameter scan (see parameter scan). A histogram draws a bar graph that shows how often the parameter took a certain value. To define a histogram instead of a curve, you click on the New histogram... button in the plot definition dialog. For the new histogram you can specify a title, the variable for which the histogram should be drawn, and the increment of the value. The increment parameter tells COPASI how wide the individual bars of the histogram are going to be. Let's say the value of the parameter was in the range of 3 to 8 and you set the increment to 0.1, COPASI will draw a histogram with 50 bars, each bar representing a value range of 0.1 units. Curves and histograms can be combined in a single plot.

Chapter 3

Tasks

3.1 Steady-State Analysis

In order to run a Steady-State analysis, you have to navigate to the Task->Steady-State branch in the object tree.



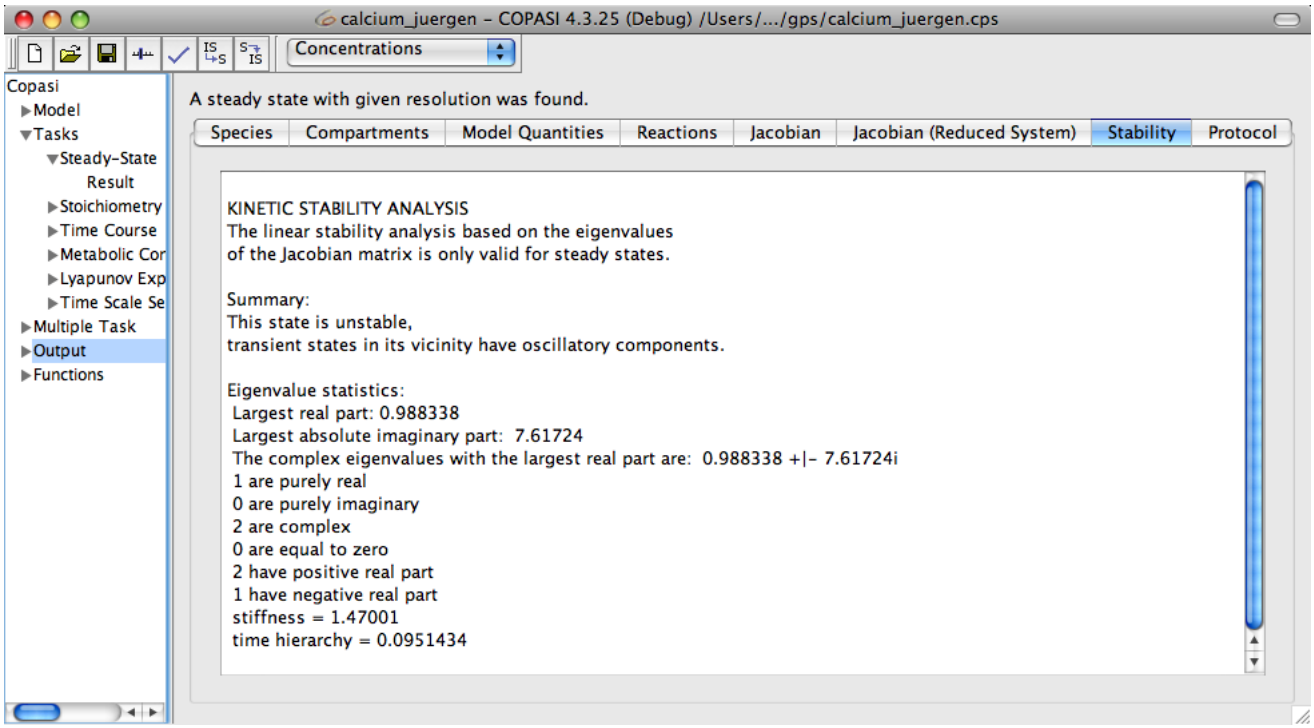
Steady-State Task Dialog

In the dialog that appears there, you can make several settings that influence the way the Steady-State analysis is calculated. First of all, you can decide whether COPASI should calculate the Jacobian matrix and/or do a stability analysis as well by checking the corresponding check box. The executable check box is used to instruct the commandline version CopasiSE to do this task if running on the corresponding file. In the Method Parameter table you can also make several settings that influence the method for calculating the Steady-State. For a detailed description of those parameters see the corresponding methods part of this documentation. To finally run the Steady-State calculation, click on the Run button at the bottom of the screen. After the calculation, COPASI will jump to the Result widget.

The Result widget for the Steady-State calculation contains several tabs for the individual results.

The first tab contains information about the species at the Steady-State condition as well as the compartments at the second tab. The third tab shows the model quantities whereas the reactions and their fluxes are displayed at the fourth one. The display between the concentration fluxes and the particle one is depending on what you have selected on the drop down list at the tab menu. As mentioned at the very beginning this documentation, you are able to easily change the display at any time.

The fifth and the sixth tab only contain results if you told COPASI to calculate the Jacobian matrix. The fifth tab then shows the Jacobian matrix for the full system and the sixth one contains that for the reduced system. For both Jacobian matrices, the eigenvalues are also shown. The seventh tab contains the results for the stability analysis if a stability analysis was requested. The last tab shows protocol that lists the calculation steps COPASI took while trying to find a Steady-State.



Results of the Stability Analysis

In order to have an output from the Steady-State Task, you have to create an output definition as described in the output section or you use the default report named *Steady-State*. The default report prints a message telling whether a Steady-State was found. It also reports the concentration, concentration rate, particle number, particle number rate, and transition time of all species as well as the flux for all reactions. If they have been requested, the Jacobian matrix and the eigenvalues are also reported. The easiest way to get a customized output is probably to use the output assistant activated via the Output Assistant button. This is described in the output assistant section. All that is left to do in order to write the output to a specific file is to connect output definition with a file. This can be achieved by clicking on the Report button. This opens a dialog that lets you connect the report of a specific task to a file on your hard disk. First we choose a report that is suitable for the Steady-State task from the drop down list at the top of the dialog. Next, we specify a file that will be used to store the report by clicking on the browse button and selecting the destination in the file dialog that opens. Per default, COPASI creates a new file or overwrites an existing file with the same name. Alternatively, you can tell COPASI to append the report to the end of an existing file by selecting the corresponding check box labeled Append at the bottom of the dialog. Once you are finished, you click on the Confirm button. If you now run the task, COPASI will write the output to the file you specified.

3.2 Stoichiometric State Analysis

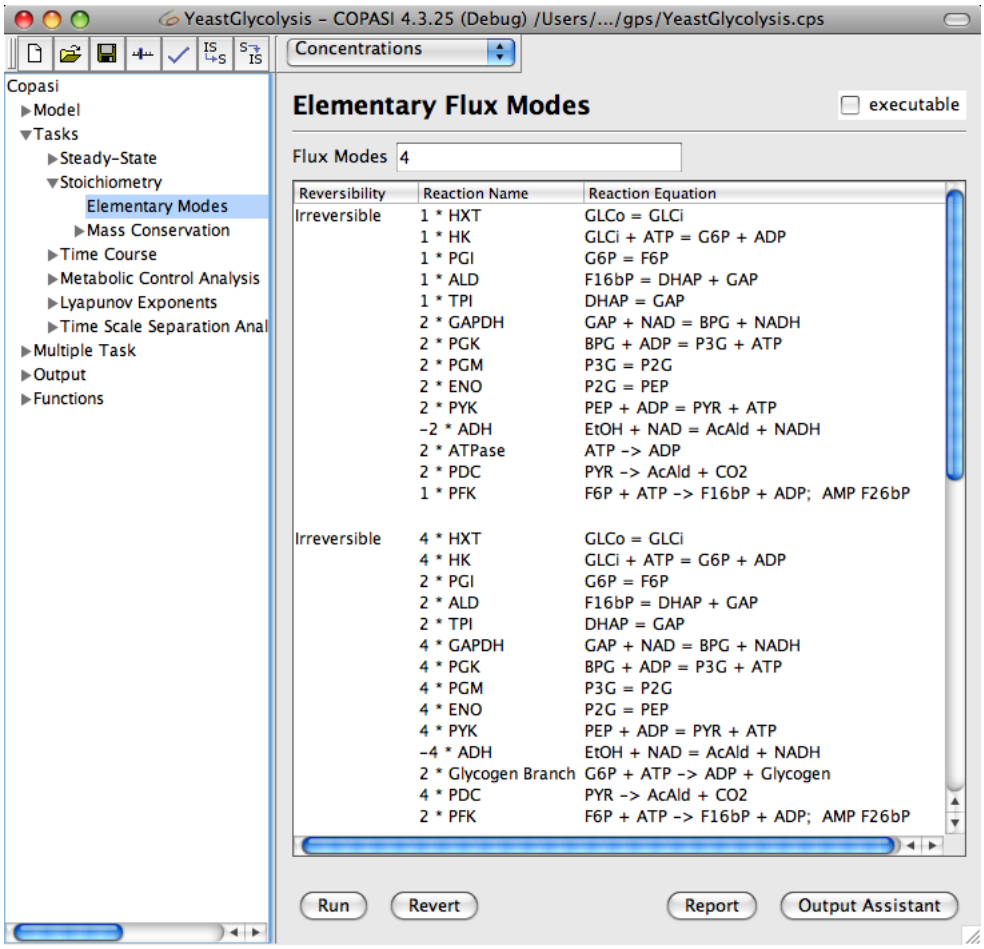
3.2.1 Elementary Flux Modes

Letting COPASI calculate the elementary flux modes for the system is very easy. Select the Tasks->Stoichiometry->Elementary Modes in the object tree and click on the Run button in the dialog that appears.

The elementary modes found will be displayed directly in this dialog. Each elementary mode lists the reactions it consists of together with their chemical equations.

If you want to have output from the Elementary Mode calculation, you have to create an output definition as described in the manual definition section. The easiest way is probably to use the output assistant which activated via the Output Assistant button. This is described in the output assistant section. All that is left to do in order to write the output to a specific file is to connect an output definition with a file. This can be achieved by clicking on the Report button. This opens a dialog that lets you connect the report of a specific task to a file on your hard disk. First we choose a report that is suitable for the Elementary Flux Modes analysis task from the drop down list at the top of the dialog. Next, we specify a file that will be used to store the report by clicking on the browse button and selection the destination in the file dialog that opens. Per default, COPASI creates a new file, if the file does not exist, or overwrites an existing file with the same name. Alternatively, you can tell COPASI to append the report

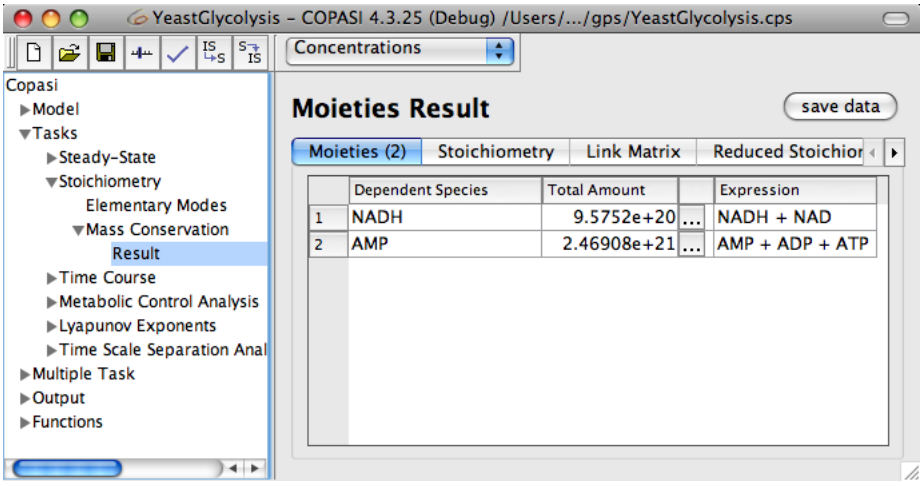
to the end of an existing file by selecting the corresponding check box labeled Append at the bottom of the dialog. Once you are finished, you click on the Confirm button. If you now run the task, COPASI will write the output to the file you specified.



Elementary Flux Modes Analysis Dialog with Results

3.2.2 Mass Conservations

Calculating mass conservations in COPASI is also very easy. Navigate to the Tasks->Stoichiometry->Mass Conservation branch in the object tree and click on the Run button as with every other task.



Mass Conservation Task Dialog with Results

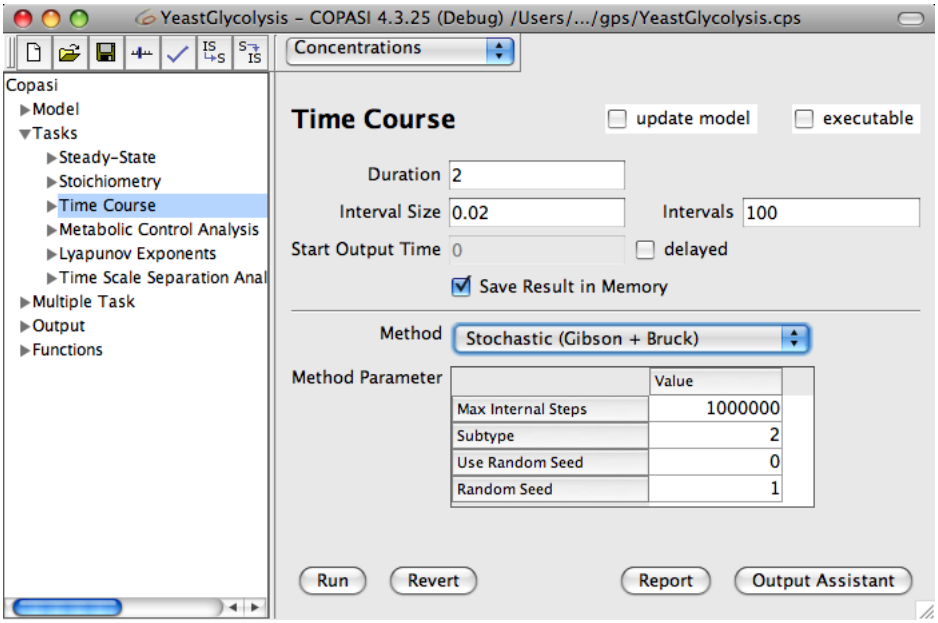
Moieties are the result of stoichiometric analysis. The result of this analysis is not unique it depends on the algorithm chosen.

COPASI uses the Householder reduction method described in [Vallabhajosyula06]. To make matters worse it even depends on the implementation of the Householder reduction for which COPASI relies on platform specific libraries for performance reasons, i.e., the result may differ on Windows and MacOS X. These differences are not crucial as the moieties build a basis of the linear dependent subspace of the solution and the choice of such a base system is obviously not unique.

However, often people like to know the total preserved amount of a moiety. Since the moiety calculation is not unique COPASI provides a convenience method to create a global quantity of type assignment to calculate this value. This method is accessible through the tool button. The result of this assignment is now independent from the algorithm chosen to calculate the moieties and thus may even be exported to SBML.

3.3 Time Course Simulation

In order to do a time course simulation, you have to navigate to the corresponding task branch in the object tree which is located at Tasks->Time Course.



Trajector Task Dialog

In the *Time Course* widget you can change several parameters for the time course, e.g. the duration of the simulation and the number of intervals that are being calculated in the time range. Alternatively to setting the number of intervals, you can also set the size of the interval. If you set either one, the other will be updated accordingly. Because if you change the duration, the number of intervals will stay the same, which means that the interval size will be adjusted. The check box labeled Save Result in Memory tells COPASI to keep the result of the time series calculation in memory in order to display it in a result dialog. Since this can be a large amount of data depending on the size of your model and/or on the number of intervals you want COPASI to calculate, you should disable this if you think the result might not fit into memory. The consequence of disabling this check box is that you need to define a report in order to store the results of the time course simulation.

Another thing that you can adjust in this dialog is the time at which COPASI starts to record the output. Normally COPASI will store all output and display it in the plot and reports, if there are any, or in the results dialog, if that feature was not disabled. With the delayed check box and input field of Start Output Time, you can specify that COPASI shall drop all results prior to a certain time point. For example, you could decide that you want to run a time course simulation for 100 seconds but only interested in the last 50 seconds of the simulation. In this case you would activate the delayed check box and specify a delay of, lets say, 50. This delay will be used for all kinds of output.

The time course simulation does not necessarily start with $t_0 = 0$. The start time of the time course simulation depends on the initial model time which can be specified in the general model settings dialog.

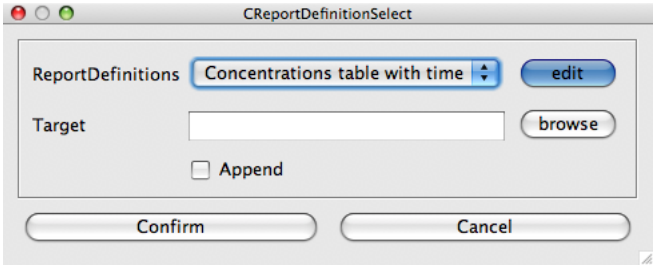
COPASI supports six different methods for calculating time course simulations. COPASI can use the *LSODA* solver, or its variant called *LSODAR*, to calculate the time course deterministically. COPASI is able to calculate the time course stochastically by means of one of two stochastic solvers: the *Gibson and Bruck* method or the *Tau-Leap* one. Depending on the solver you

have chosen, you can set several parameters in the Parameter value table that influence the way the method works. A detailed explanation of those parameters will follow in the methods part of this document.

In addition to purely deterministic or purely stochastic time course simulations, COPASI can also use a so called hybrid method to calculate a trajectory. This hybrid method splits the model in two segments according to the number of particles participating in a reaction. Reactions with many particles are simulated deterministically and those reactions with only a few particles are simulated stochastically. The boundaries of what is considered as to many or few particles can be set by the user. Depending on the nature of the model, the hybrid simulation can lead to significant simulation speedup compared to purely stochastic simulation while still being more accurate than purely deterministic simulation for small particle numbers. However, the method should still be considered experimental. For a more detailed description of the method and the attributes that you can set see the methods section.

The hybrid solver comes in two flavors. One uses LSODA for deterministic simulation and is called *Hybrid (LSODA)*, the other uses Runge-Kutta fourth order and is called *Hybrid (Runge-Kutta)*.

If you haven't created a report definition yet, you can use the output assistant to easily create one by clicking on the corresponding button at the bottom of the time course dialog. Once you have created a report definition, you have to associate this report definition with a file for COPASI to be able to write the results to that file. To do that, you click on the Report button.

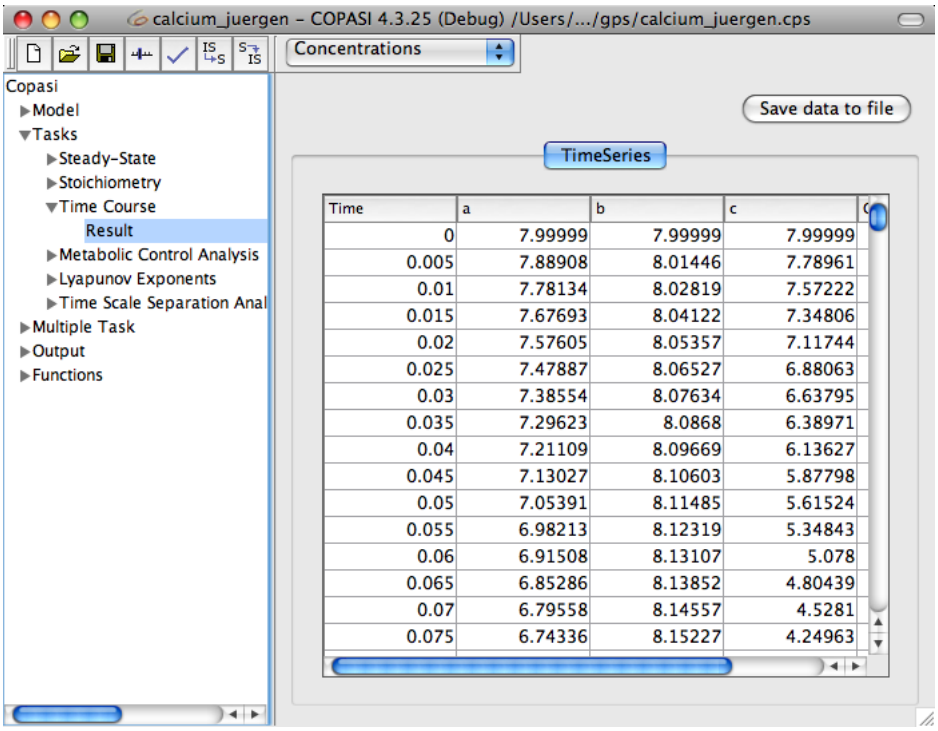


Dialog to associate a Report with a File

The dialog that pops up will let you choose the report you want to use (in case you created more than one) and lets you browse for a file to store the report to. Additionally, you can choose if you want to append the report to an already existing file. The default is to create a new file, or to overwrite an existing file. If you want to append to the selected file, you have to check the Append check box.

Once you made all the desired changes to the parameters, you can start the time course simulation by clicking on the Run button. COPASI will show a progress bar while running the simulation, which might take some time depending on several factors, like the hardware you are using, the simulation method you chose, and/or the size of your model. Once COPASI finishes the calculation, the results will be displayed in the report file you defined and/or in a separate result dialog, if you told COPASI to keep the results in memory.

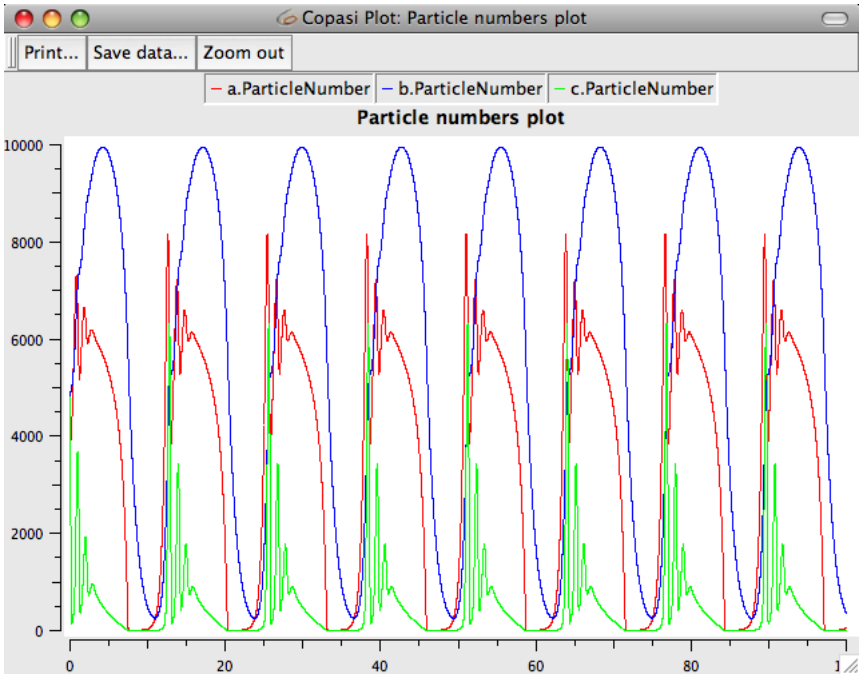
The Result dialog is located directly below the Time Course branch in the object tree. In this widget you can choose whether you want the results to be displayed as concentrations or as particle numbers, and you have the possibility to store the results to a file. The advantage that a report has rather than writing a file in the Result widget is that you can choose exactly which species concentrations you want to store whereas the Result dialog always stores all species concentrations. You also can't change the order in which the species concentrations are written, if you store the result from this dialog.



Trajectory Task Results

3.3.1 Working with Plots

If you have defined an active plot before calculating a trajectory, COPASI will draw the plot. The plot window has three elements. A tool bar at the top, that lets you print the plot or save the data into a file; the legend, which is interactive such that the drawing of certain curves can be toggled by clicking on the corresponding legend entry; and the actual plot.



Plot Window

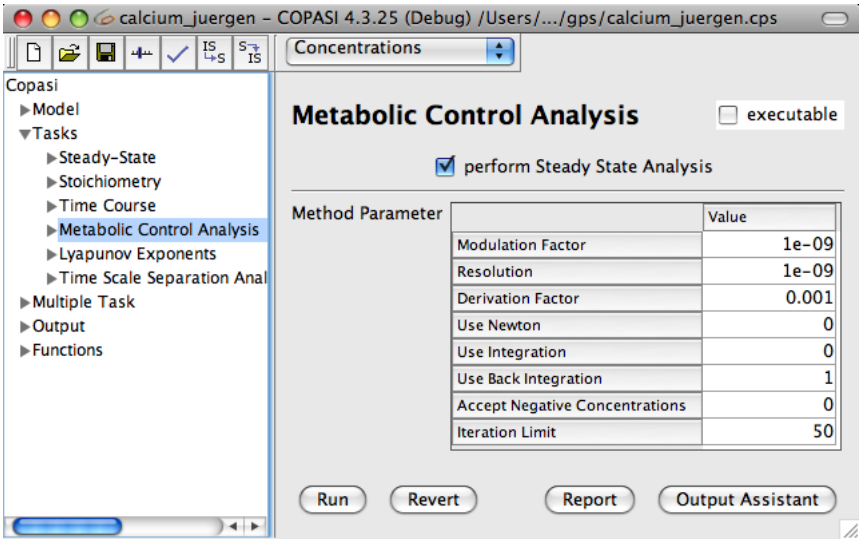
During moving the mouse cursor inside a plot widget, the mouse coordinates with respect to the coordinate system of the plot are displayed beside the mouse cursor.

In order to zoom further into a plot, you can select a rectangular area on the plot by clicking somewhere in the plot and dragging the pointer. The plot will now zoom into the area you just selected. To go back to the original plot, right click on the plot area.

Caution Macintosh users with single button mice should use CTRL-click.

3.4 Metabolic Control Analysis (MCA)

COPASI can also do a *Metabolic Control Analysis (MCA)* for your model. The MCA task is located under Tasks->Metabolic Control Analysis.



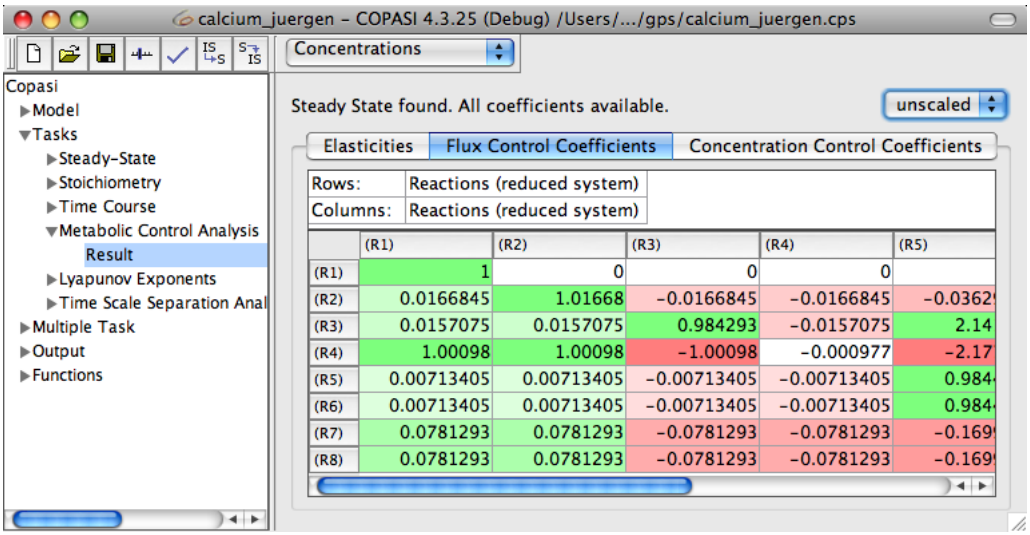
MCA Task Dialog

In order to do a full MCA (elasticities and control coefficients), COPASI needs to look for a Steady-State first, otherwise COPASI can only calculate the elasticities. If you did not already do a Steady-State analysis and updated the model so that the system already is in the Steady-State, you should enable the check box that tells COPASI to do a Steady-State calculation before calculating the MCA. Depending on whether COPASI needs to do a Steady-State analysis or not, you can change one or more parameters that influence the way the MCA and the Steady-State are calculated. The parameters for the Steady-State calculation are explained in the Steady-State analysis section.

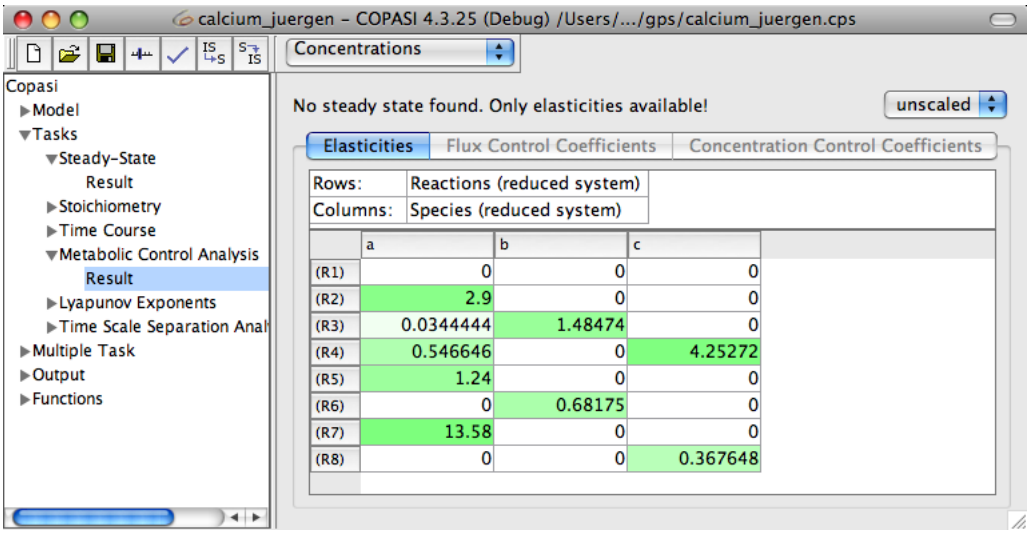
To start the calculation, you click the Run button. After the calculation is finished, COPASI will automatically switch to the Result dialog. The dialog shows three tabs that contain the results of the *Elasticities*, *Flux Control Coefficients* and *Concentration Control Coefficients*.

Depending on whether a steady state condition was found or not, only the *Elasticities* tab might be enabled. COPASI will state the status of finding a steady state in a label right above the tabs. For all of the results, you can choose if you want COPASI to display them scaled or unscaled.

In order to have an output from the MCA, you have to create a report as described in the section or you decide to use the default report. The default report will output all matrices that are calculated as well as the Steady-State, if steady state calculation is requested. All that is left to do in order to write the output to a specific file is to connect output definition with a file. This can be achieved by clicking on the Report button. This opens a dialog that lets you connect the report of a specific task to a file on your hard disk. First we choose a report that is suitable for the MCA task from the drop down list at the top of the dialog. The default report for MCA is called *Metabolic Control Analysis*. Next, we specify a file that will be used to store the report by clicking on the browse button and selecting the destination in the file dialog that opens. Per default, COPASI creates a new file or overwrites an existing file with the same name. Alternatively, you can tell COPASI to append the report to the end of an existing file by selecting the corresponding check box labeled Append at the bottom of the dialog. Once you are finished, you click on the Confirm button. If you now run the task, COPASI will write the output to the file you specified.



MCA Results at Steady-State



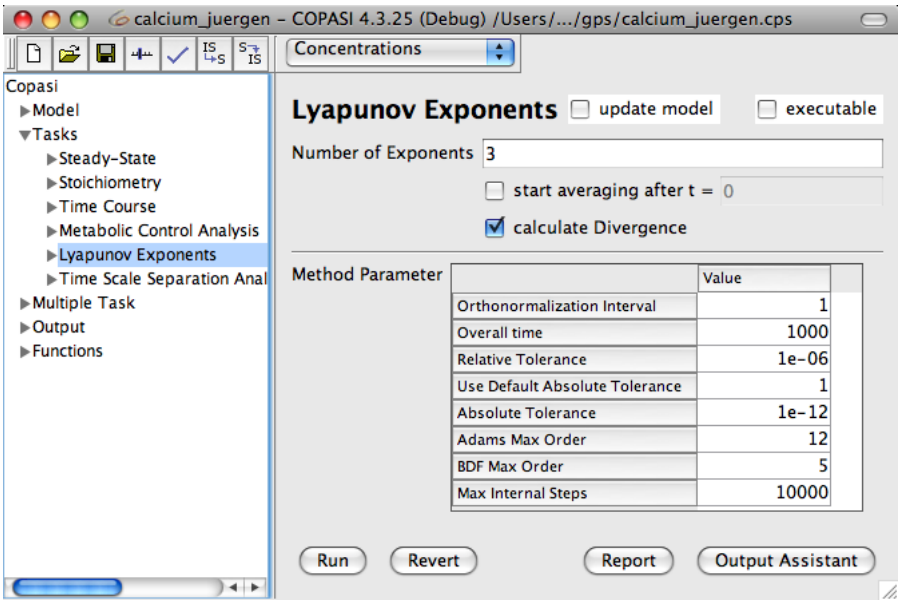
MCA Results when no Steady-State was found

3.5 Lyapunov Exponents

The widget to calculate Lyapunov exponents in COPASI is located in the Task branch of COPASI's tree view. If you select the Lyapunov Exponents item in the tree, the corresponding widget will be displayed. The first input field in the widget labeled Number of Exponents lets you specify how many Lyapunov exponents are to be calculated. This number should be between one and the number of independent variables in the system (that is the number of species that are not constant minus the number of mass conservation relations). If you specify a number that is higher than the number of independent variables in the system, COPASI will issue a warning telling you to lower the number and will also tell what the maximal number should be. During the calculation of the Lyapunov exponents, a time course simulation is carried out. If your model shows a long transient, you might want to exclude the beginning of the trajectory from the calculation of the Lyapunov exponents. For this, COPASI lets you specify at which time point the averaging for the Lyapunov exponents should start. You can specify this delay in the field labeled start averaging after t= ".

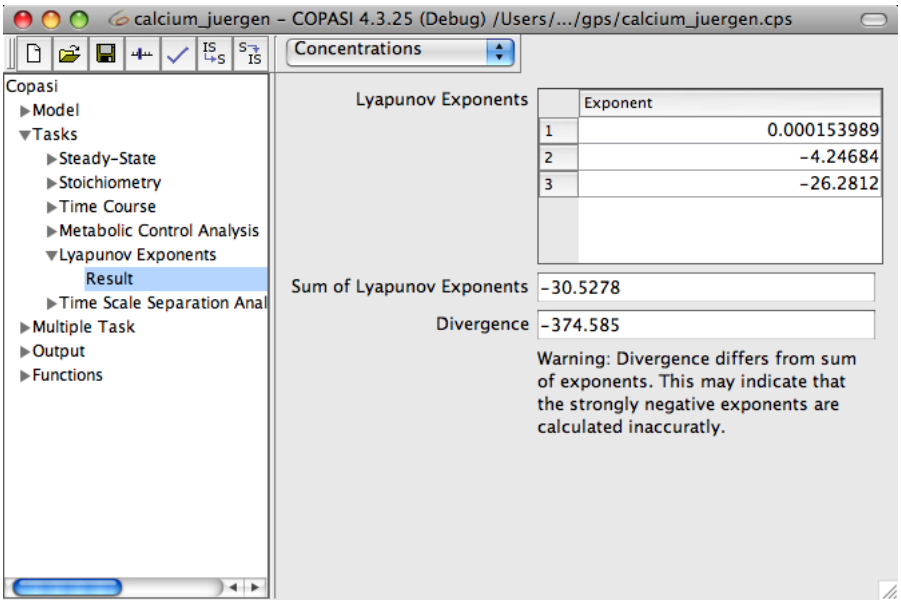
With the check box labeled calculate Divergence, you can activate the calculation of the average divergence. The divergence is calculated as the average over the trace of the Jacobian (see Lyapunov calculation method).

The method behind of calculating Lyapunov exponents is called *Wolf Method*. For a more detailed description of the method and its parameters please see the Lyapunov exponents methods.



Lyapunov Task Dialog

After clicking the Run button, COPASI will start the time course simulation in order to calculate the Lyapunov exponents. Once the calculation is finished, COPASI will jump to the Result window. The window shows the calculated Lyapunov exponents in a table and beneath the table it shows the sum of the calculated exponents. If you told COPASI to calculate the divergence as well, this will be shown underneath the Sum of Lyapunov Exponents display. If you told COPASI to calculate all Lyapunov exponents (as many as there are independent variables in the model), the sum of the exponents and the divergence should be equal; if it isn't COPASI will display a warning. Since the warning only makes sense if all exponents have been calculated, it will not be displayed otherwise. It is expected that the divergence and the sum differ in those cases.



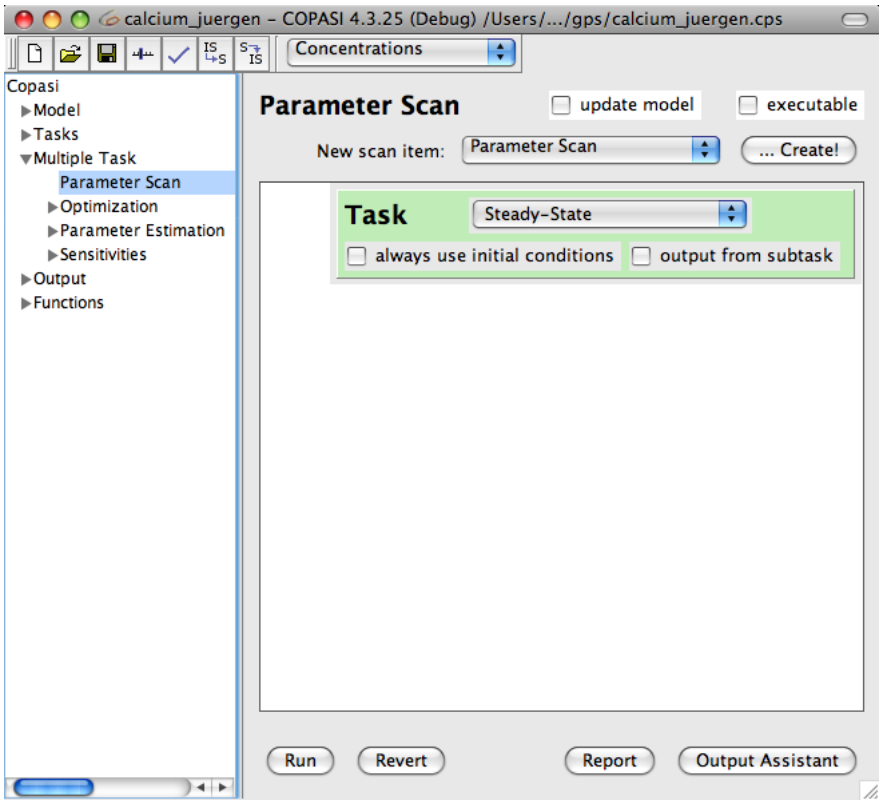
Results for the Lyapunov Exponent Calculation

3.6 Time Scale Separation

Caution This is under construction.

3.7 Parameter Scan

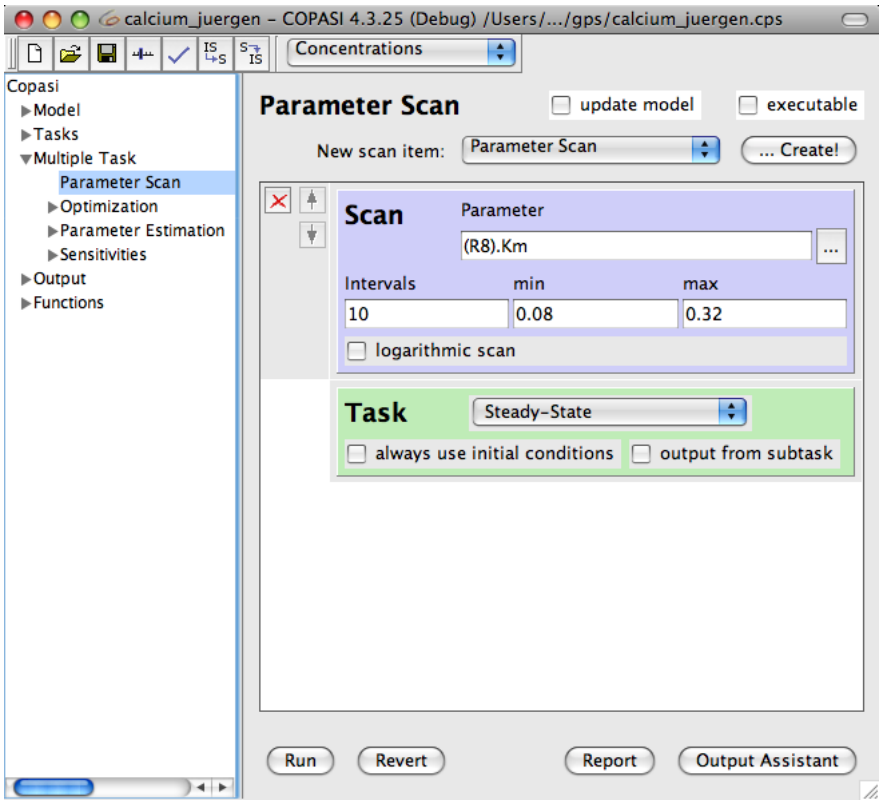
The leaf for Parameter Scan in the object tree is located under Multiple Tasks. At the top, the dialog displays a box called New scan item together with a ... Create! button. When the dialog is opened for the first time, the only item in the widget below is a green box called Task. This widget contains a drop down list with all the tasks that can be used in a scan. Additionally there are a check box called always use initial conditions and a check box called output from subtask.



Empty Scan Task Dialog

The tasks for which scans can be conducted are *Steady-State*, *Time Course*, *Metabolic Control Analysis*, and *Lyapunov Exponents*. So if you want to do a scan for a time course calculation, you should choose *Time Course* from the drop down list in the Task widget. We will ignore the two check boxes for the time being and take a look at the drop down list at the top of the Scan Task dialog. This drop down list contains three entries named *Parameter Scan*, *Repeat* and *Random distribution*. Those are additional elements that can be added to the main widget below to form complex tasks.

Let us look at one after the other and start with the *Parameter Scan*. As it is right now, the scan task does not do anything since we have not told it yet which parameter we want to scan. To define a parameter for the scan, you choose *Parameter Scan* from the drop down list at the top and press the ... Create! button. COPASI will add a new widget to the dialog that is called Scan. The Scan widget contains some empty line edit fields and a button beside the line edit labeled Parameter and a check box labeled logarithmic scan.

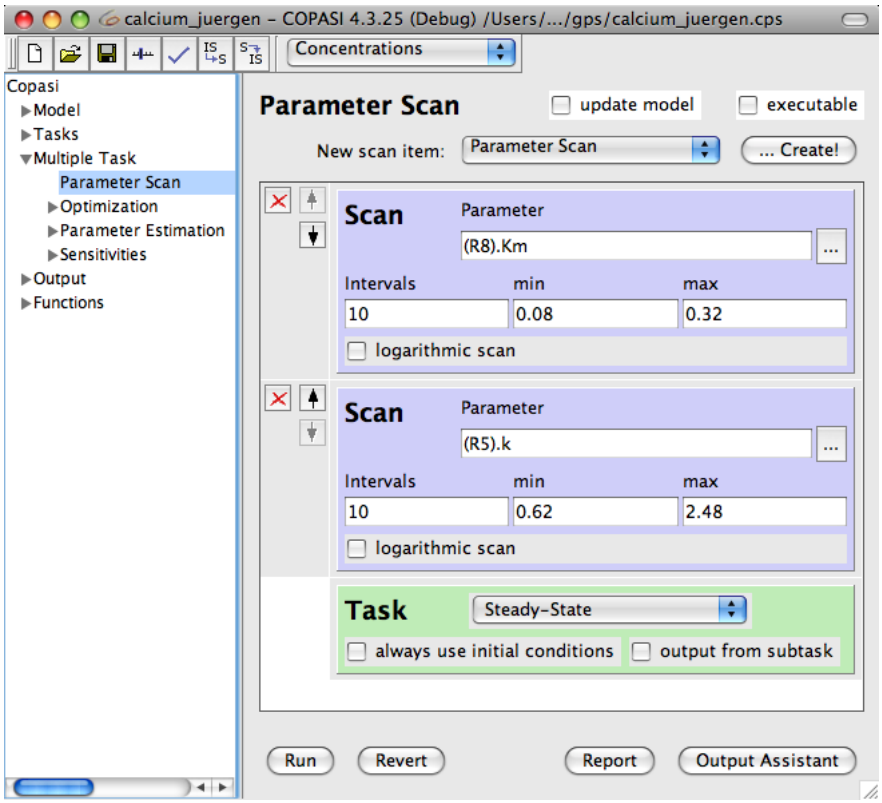


Scan Task Dialog with Scan Item

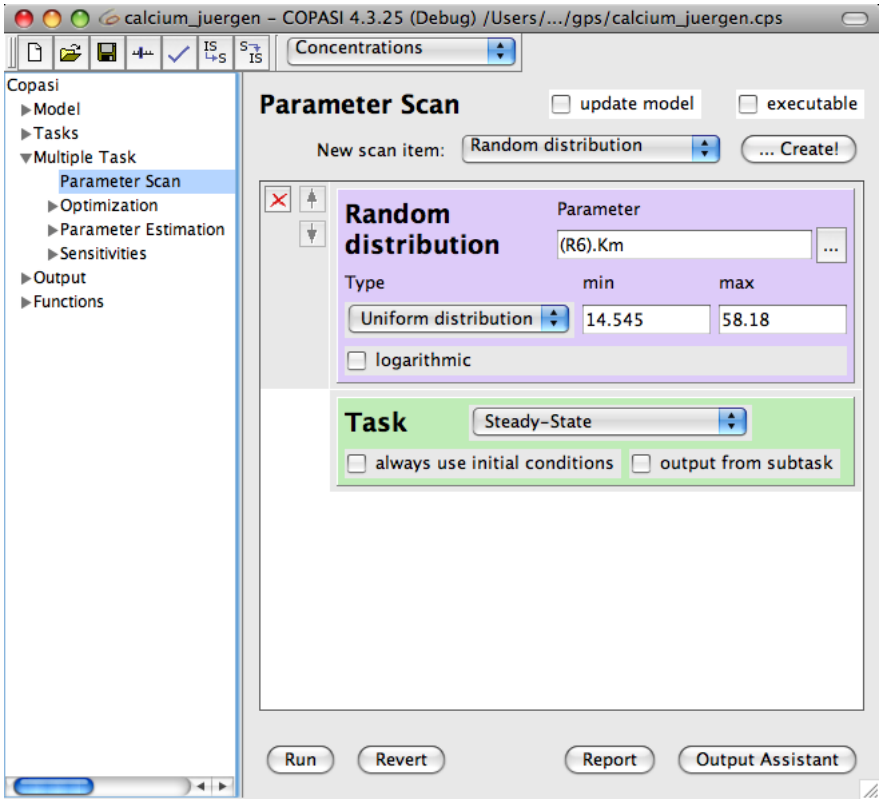
The first thing you have to do is to select the parameter for the scan. You do this by clicking on the ... button and selecting a parameter from the selection dialog that appears. Once you are finished with your selection and pressed the OK button, the name of the object you chose will appear in the line edit field beside the ... button. In the line edit fields below you can now specify the minimal value the parameter will have during the scan as well as the maximal value. After choosing the object, these values will be set to half the objects value for the minimum and double the objects value for the maximum. In the Intervals, you can specify how many intervals COPASI uses during the scan to raise the value from minimum to maximum. Last but not least, the logarithmic check box determines whether the value is raised in linear steps if the box is unchecked or in logarithmic steps if the box is checked. Now you are set to run your first simple scan by clicking on the Run at the bottom of the dialog. In order to actually see some result, you have to have some kind of output defined. The scan task can generate reports and, if a plot is defined, the scan task will do plotting while running the scan. If you did not change the number of intervals from the default, the scan task will run 10 time course simulations each with a different value for the chosen parameter and in turn you will see 10 plots overlaid in one plot window.

If you want to have output from the parameter scan, you have to create an output definition as described in the output section. The easiest way is probably to use the output assistant which you activate via the Output Assistant button. This is described in the output assistant section. All that is left to do in order to write the output to a specific file is to connect an output definition with a file. This can be achieved by clicking on the Report button. This opens a dialog that lets you connect the report of a specific task to a file on your hard disk. First we choose a report that is suitable for the parameter scan task from the drop down list at the top of the dialog. Next, we specify a file that will be used to store the report by clicking on the browse button and selecting the destination in the file dialog that opens. Per default, COPASI creates a new file or overwrites an existing file with the same name. Alternatively, you can tell COPASI to append the report to the end of an existing file by selecting the corresponding check box labeled Append at the bottom of the dialog. Once you are finished, you click on the Confirm button. If you now run the task, COPASI will write the output to the file you specified.

So far we have only scratched the surface of the scan dialog. E.g. if you want to do a two-dimensional scan, i.e. a scan where two parameters are independently scanned, you can add a second Scan widget by adding it to the main dialog just like you did for the first parameter. You select *Parameter Scan* in the drop down list and push the ... Create! button.



Scan Task Dialog with 2 Scan Items



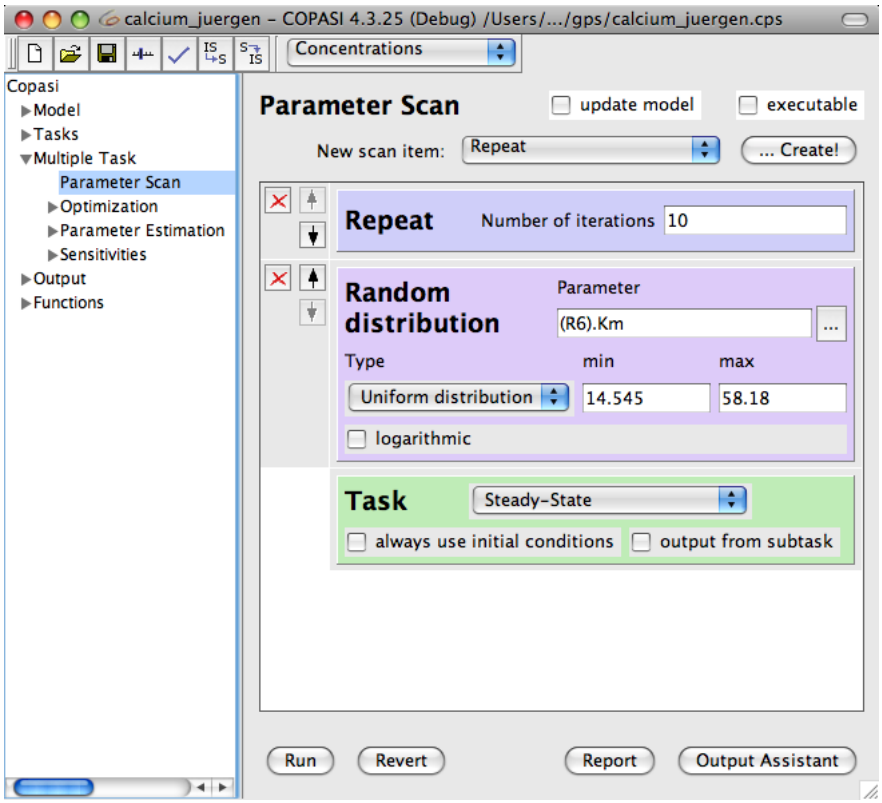
Scan Task Dialog with Random Distribution Scan Item

After adjusting the minimum and maximum as well as the number of intervals, you can run the scan task again. COPASI will now do a scan for the first parameter while holding the second parameter constant at the minimum. Next, COPASI will raise the second parameter and do another scan from minimum to maximum for the first parameter. This way a full scan for the first parameter is done for every value of the second parameter. You have to be careful because assuming that you chose 10 intervals

for both parameters, COPASI will run 100 time course simulations during this two-dimensional scan which can take a long time.

The *Random distribution* item is similar to the *Parameter Scan* item. With the random distribution, a parameter can be given a random value. After adding a Random distribution widget to the main dialog, you first have to choose a parameter for which a random value will be set. You can then choose from three distribution types to generate the random value. The three distributions are *Uniform distribution*, *Normal distribution* and *Poisson distribution*. Additionally, you have to set the bounds within which the random value will be set (uniform distribution) or the mean and average values (normal distribution) or just the mean value (Poisson distribution). If all has been set and you press the Run, COPASI will set the parameter value to a random value from the chosen distribution and run one time course simulation.

The *Repeat* item can be used to repeat a certain action several times. For example, if you add a Repeat widget above a Random distribution one, the parameter will be given a random value as many times as specified in the Repeat widget and time course simulation, or one of the other tasks specified in the Task one at the bottom of the main widget, is run with this value.



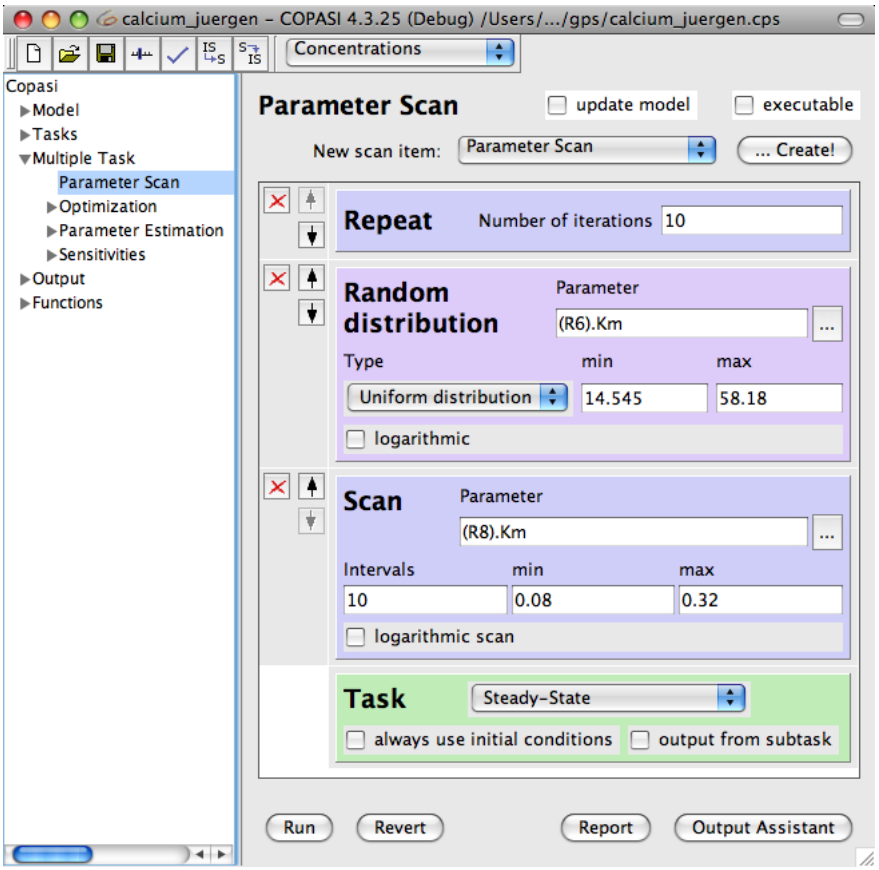
Scan Task Dialog with Random Distribution and Repeat Scan Items

Different items can be combined in many ways to achieve certain goals. For instance, you could run 10 parameter scans for a certain value, each with a different random value for another parameter, by combining a Scan widget with a Random distribution and a Repeat ones.

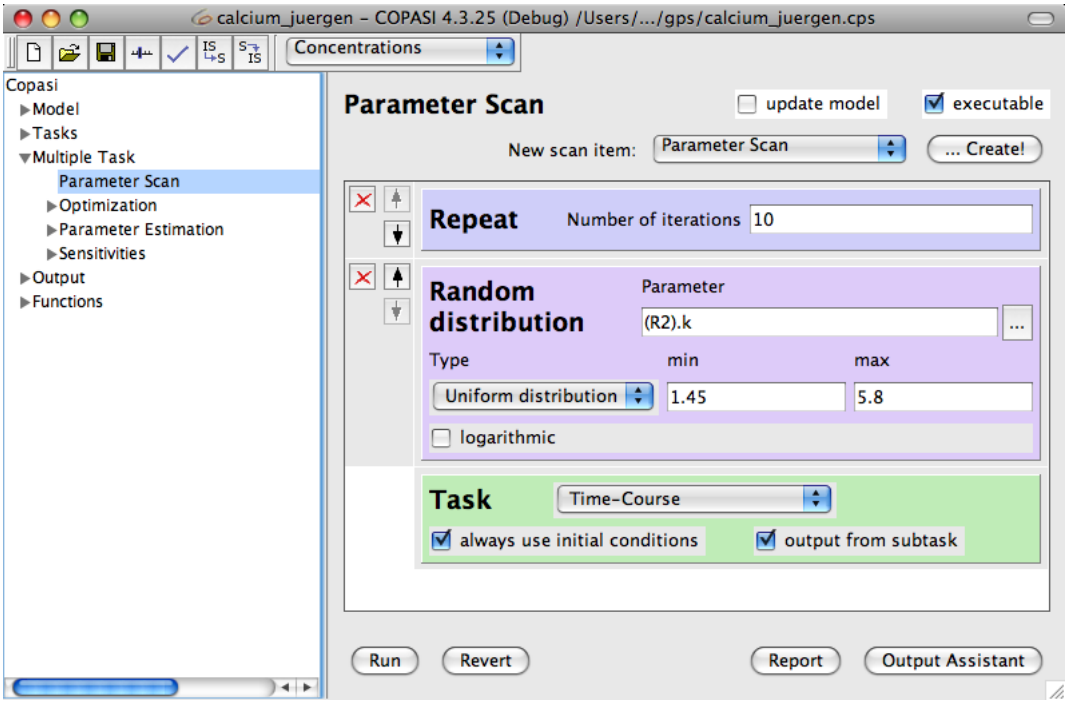
With the buttons on the left of each item, except the Task at the bottom, you can move the items up or down in the list or close an item you no longer need. The order of the items in the main widgets determines in which order COPASI will handle the actions. A widget is controlled by the widget above it. Thus, it can be rather important, if a Repeat acts on the widget directly below it.

Consider you have added a Repeat widget and a Random distribution one to the main widget. Now the Repeat widget can be either above the Random distribution widget or below it, and the results you get after pushing the Run button are very different.

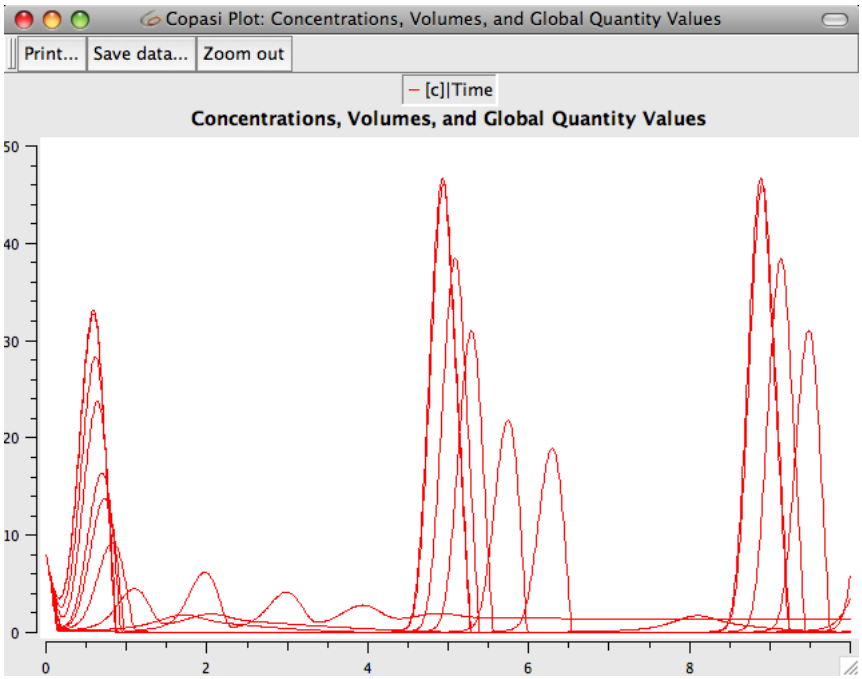
Lets first look at the case where the Repeat widget is above the Random distribution widget. If you push the Run button, COPASI will run the task as many times as you have specified in the Repeat widget, each time with a different random value for the parameter you have chosen in the Random distribution widget. Now lets assume the Repeat is below the Random distribution. If you push the Run button now, COPASI will also run the task as many times as you specified in the Repeat widget. The difference is that this time, the parameter value gets a random value before the repeat takes effect, i.e. all runs are done with the same random value for the chosen parameter. If you do this for a deterministic time course simulation and a concentration plot is defined, the difference becomes obvious because in the first case, you will probably get X different curves, whereas in the second case you may get X times the same curve.



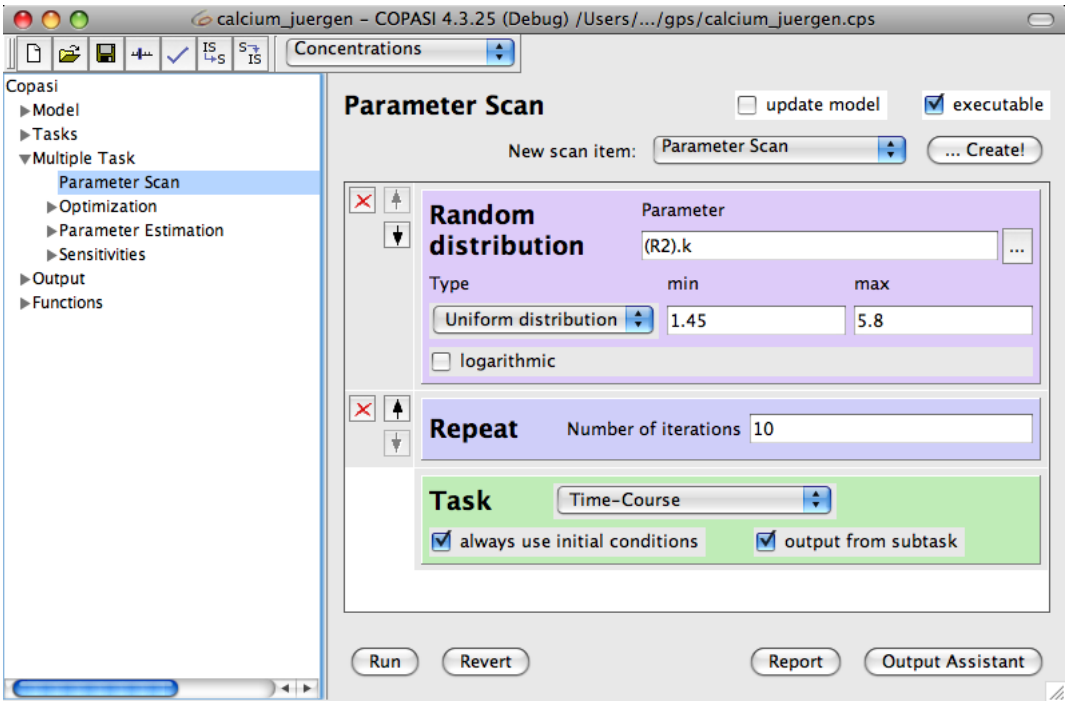
Scan Task Dialog with Combination of Actions



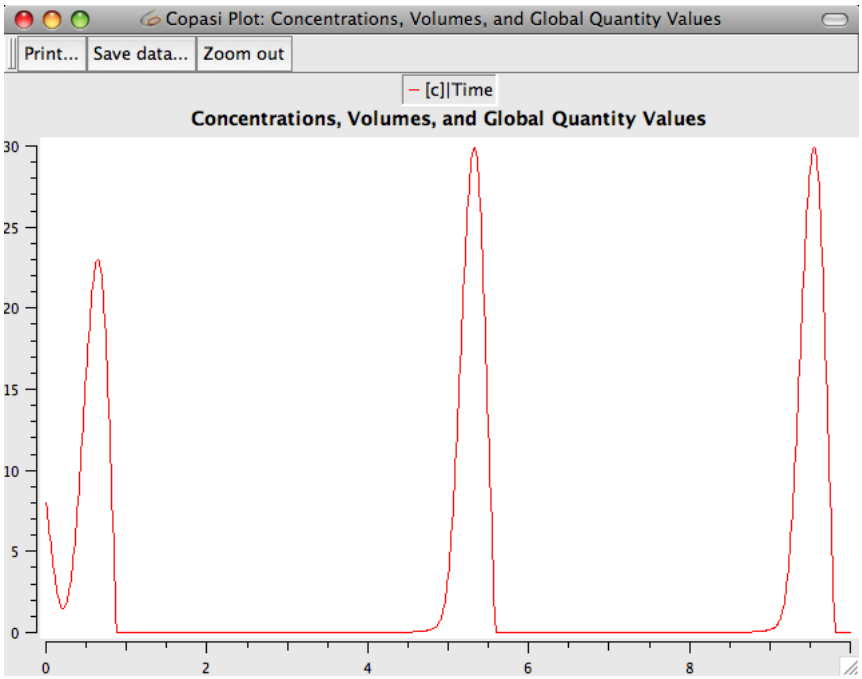
Scan Task Dialog with repeated Random Distribution



Plot with 10 different Curves



Scan task Dialog with with Random Distribution where Time Course Task is repeated



Plot with 10 identical Curves

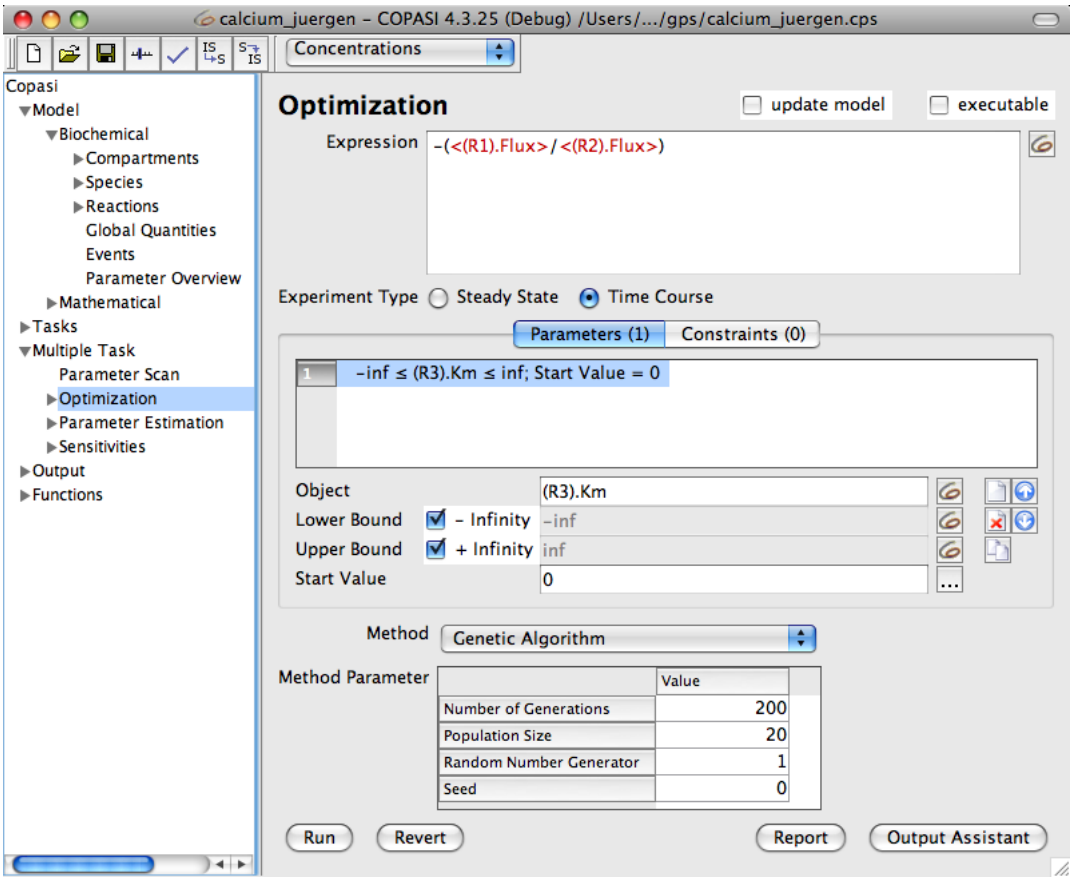
What we have left out so far are the two check boxes at the bottom of the Task widget. The first check box is labeled always use initial conditions. If this check box is checked, each task that is run runs with the same initial conditions. If the check box is not checked, only the first task that runs uses the initial conditions from the model, each subsequent task uses the conditions as they are after the preceding task has finished.

The second check box is labeled output from subtask. If it is checked, COPASI will plot or write the complete time series result each time a time series task runs. Correspondingly it will output intermediate results for the other tasks. So if during your scan, 10 time series are calculated, all ten are overloaded in the same window. If the check box is unchecked, COPASI will only plot the end result, i.e. the concentrations after the last step, of each time series. This applies to reports correspondingly. This feature is useful if you want to plot some calculation results as a function of the parameter that is scanned. So e.g. you could plot the end result of a Steady-State calculation on the y-Axis versus a kinetic parameter on the x-Axis.

3.8 Optimization

The optimization task lets you minimize a given objective function by scanning one or more parameters over a given range. This probably sounds rather cryptic, therefore, lets try to illustrate this with a simple example.

We assume you have a model that consists of several reactions and two of those reactions fluxes (R1, R2) depend on a certain parameter k (either directly or indirectly). Now you are interested in finding the optimal value of k so that the ratio of R1/R2 will be maximal. So lets see how you would do this in COPASI.



Optimization h Task Dialog

If you go to the optimization dialog which you can find in the tree under Multiple Task->Optimization, you will get a screen similar to the one above. At the top of the dialog, you can find an edit line called Expression where you have to input your objective function, that is the expression that COPASI will try to minimize during the optimization. In our case, we want to find the value of k where the ratio of R1 and R2 is maximal. At the beginning of this section, I stated that COPASI will minimize a given objective function, but we would like to maximize the objective function R1/R2. So how do we generate a function to be minimized out of the function R1/R2? This can be achieved in two ways, either we inverse the ratio in order to minimize R2/R1, or we add a "-" sign to our ratio R1/R2. (For the sake of simplicity, I will assume here that the fluxes R1 and R2 will always be positive.) For this example I will use the second possibility and therefore the objective function for COPASI to minimize will be -(R1/R2). Now you can't just enter this expression into the line edit field since COPASI does not associate the names R1 and R2 with the fluxes through your reactions. In order to build this expression, you have to start typing "-(" into the line edit field. Now, you have to press the button to the right of the line edit field which opens the object browser. In this object browser, you select the flux (particle or concentration) which belongs to R1 and press the OK button. Now you will get a string that corresponds to this flux object right after the part of the expression you already typed. You can now go on by typing "/" followed by the selection of the flux for R2. You end the expression by typing ")". Please be aware that you are allowed to edit the expression, but only those parts of the expression that do not belong to object representation strings. That means everything that has been inserted via selection from the object browser may not be modified. You may however delete complete identifier expressions.

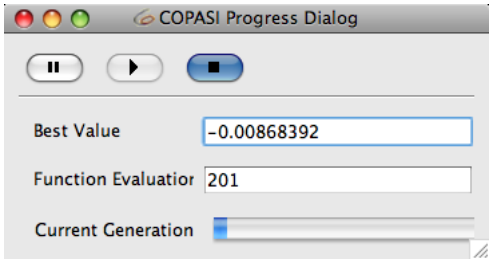
You can run an optimization task with several different methods which can be selected via the drop down list below the input field for the objective function. Each of those methods has a number of parameters which are documented in the methods chapter.

The optimization task can also be run on either the "Time Course" or the "Steady-state" subtask. Which one is used for the optimization run can be selected via the correspondingly named check boxes. So for our example we would choose the "Time Course" subtask.

The only thing we have not done yet is to tell COPASI which parameter(s) it should scan in order to minimize the objective function. In the middle of the widget, there is another line edit field right after Object where you can specify which parameter is to be scanned. The selection is again done via clicking on the button to the right of the edit field and selecting the correct parameter from the object browser. Please note that is possible to create multiple parameters at once. Below the Object you can specify the upper and lower bounds for the parameter during the optimization. Those bounds can also be expressions depending on other parameters. Per default, the check boxes for -Infinity and +Infinity are selected as the boundaries. Since a computer

can't handle infinitely small or large numbers, the search will effectively proceed from the lowest possible to the largest possible double precision number. If you want to specify your own range, you first have to deselect the +Inf or -Inf check box and then you can set your own bounds. As a matter of convenience you may enter -X% or +X% as the lower and upper limits. This instructs COPASI to calculate the limits based on the start value. The start value is the initial parameter value used by COPASI in any fitting attempt. Per default COPASI selects the current model value of the parameter to be estimated as the starting value. You may manually override this default or use the ... button to reset it to model values, randomize it, or set it to the last estimated values. Please note, if the start value of a parameter is outside the boundaries specified, COPASI will force it to the nearest boundary during the optimization. If you want to delete a parameter from the scan list, you just select it and click on the delete button which you find at the right side. If you want to scan more than one parameter, you just add more by clicking on the new button and add as many parameters as you need. Please note, you may select multiple parameters and edit them simultaneously. You can also change the order in which the parameters are scanned by moving them up or down in the list with the corresponding buttons at the right side of the widget. This has only an effect if the parameter boundaries of one parameter depend on another parameter. COPASI does not currently determine such a dependency and it is left to the user to order the parameters appropriately.

In addition to parameters COPASI knows also about constraints. Constraints are applied to the solution. i.e., they are evaluated after the simulation which can currently be either a time course or a Steady-State calculation. A possible constraint could be that the Steady-State concentration of a species has to be within a certain range that can be specified by the user. The widget for specifying these constraints can be found under the tab called Constraints and the constraints are specified in the same way as the parameters to be scanned. Each constrained consists of an object that is to be constrained and an upper and lower bound that define the constraint for the object. Just like for the parameters, several constraints can be specified.

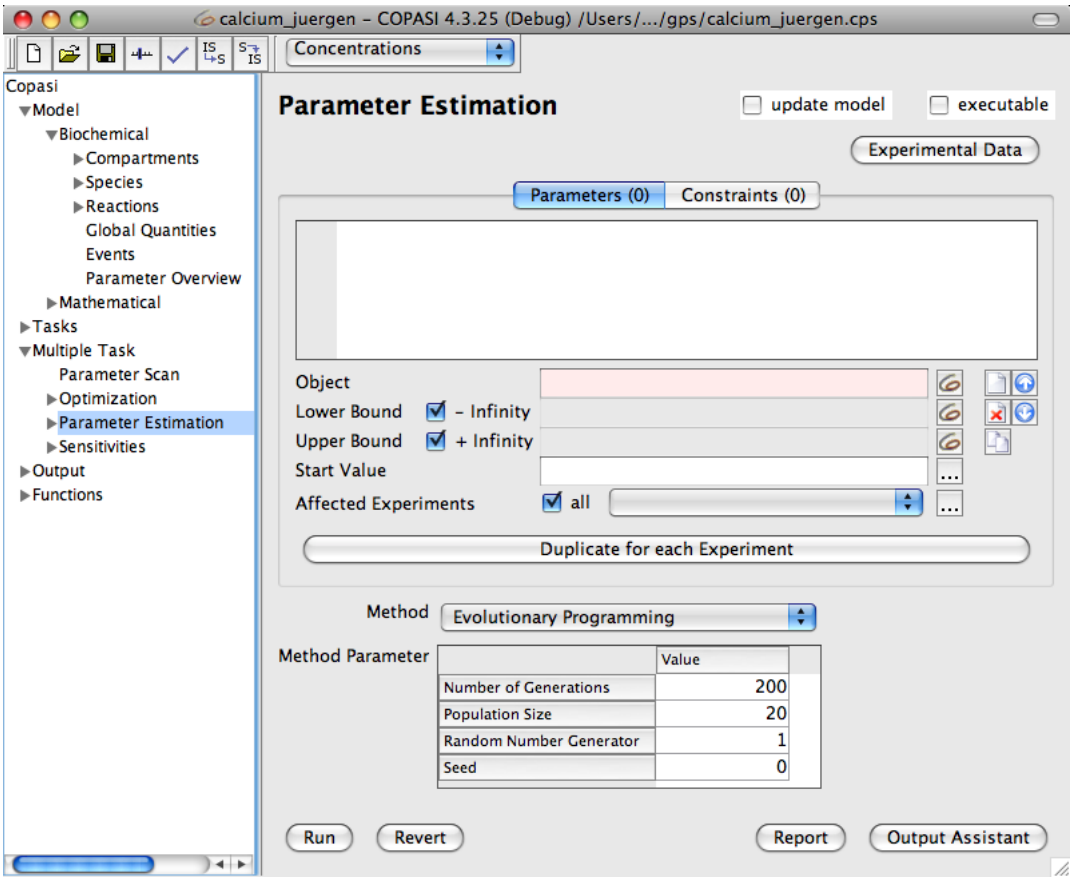


Optimization Progress Dialog

Once you have set up all the parameters, you are ready to run the optimization task by clicking on Run. COPASI will now display a progress dialog which informs you about the progress of the calculation and about the best (smallest) value found for the objective function up to this step. Since this progress dialog will close once the optimization task has run, you probably also want to define a report to be written during the calculation. In many cases using the default report name "Optimization" is sufficient. The default report outputs a description of all the settings you provided for the optimization run. It prints intermediate results during the calculation every time the target value has improved. In the end it prints a summary of the result. The easiest way to define a customized report is to use the output assistant. Alternatively, you can create a report manually as described in the output section. Once you have created a report definition for the optimization task, you click on the report button at the bottom of the dialog. In the dialog that opens, you select the report you just created from the drop down list labeled Report Definitions and then you select a filename where the report is to be stored in the field labeled Target. You can either type a filename manually, or you can select one by clicking on the browse button. When you are finished, you click the Confirm button. Now the next time you run the optimization, a report will be stored in the location you specified.

3.9 Parameter Estimation

Parameter estimation is the process of trying to calculate model parameters based on a dataset. This dataset can be the result of time course or steady-state experiments or both. COPASI reads a dataset, which may be comprised of several files each including possibly multiple experiments. After the load of the dataset COPASI tries to fit one or more parameters that are specified by the user to that dataset. The methods COPASI uses to estimate good parameter values are the same as in the optimization task. For a description of the different methods, you should read the methods section of this document.



Parameter Task Estimation Dialog

The dialog for the parameter estimation task can be activated by selecting the branch called Parameter Estimation under the Multiple Tasks branch of the tree view on the left side of the user interface. First you can define which parameters COPASI shall try to fit. Each parameter to be fitted can be added like in the Optimization. To do this, click on the button beside the line edit labeled Object, this will open a selection dialog where you can choose the parameter. Additionally you can specify an upper and a lower bound for the parameter. COPASI will only try to fit the parameter within those bounds. Per default, the upper and lower bound are + Infinity and - Infinity respectively. If you want to set your own bounds, disable the check boxes and enter your own value in the edit field. The value for the lower bound goes into the correspondingly labeled edit field, likewise for the upper bound. As a matter of convenience you may enter -X% or +X% as the lower and upper limits. This instructs COPASI to calculate the limits based on the start value. You can also specify other objects from the model as bounds for the parameter. To choose the value of another object as a bound for the parameter, click on the button beside the edit field and choose the object from the tree. The start value is the initial parameter value used by COPASI in any fitting attempt. Per default COPASI selects the current model value of the parameter to be estimated as the starting value. You may manually override this default or use the ... to reset it to model values, randomize it, or set it to the last estimated values. Please note, if the start value of a parameter is outside the boundaries specified, COPASI will force it to the nearest boundary during th parameter estimation. Additionally, you can restrict the effect of a parameter to a subset of the experiments you are attempting to fit. To do this select the ... to the right of Affected Experiments. A possible application is to fit different initial values for each time course experiment. To help you in such a case the Duplicate for each Experiment button will create a copy of the current parameter for each specified experiment.

3.9.1 Experimental Data

Before you can execute a parameter estimation task you need to specify the dataset which COPASI will use to fit the parameters you have specified. Each experiment of your dataset contributes to the objective function with the following weighted sum of squares:

$$E(P) = \sum_{i,j} \omega_j \cdot (x_{i,j} - y_{i,j}(P))^2$$

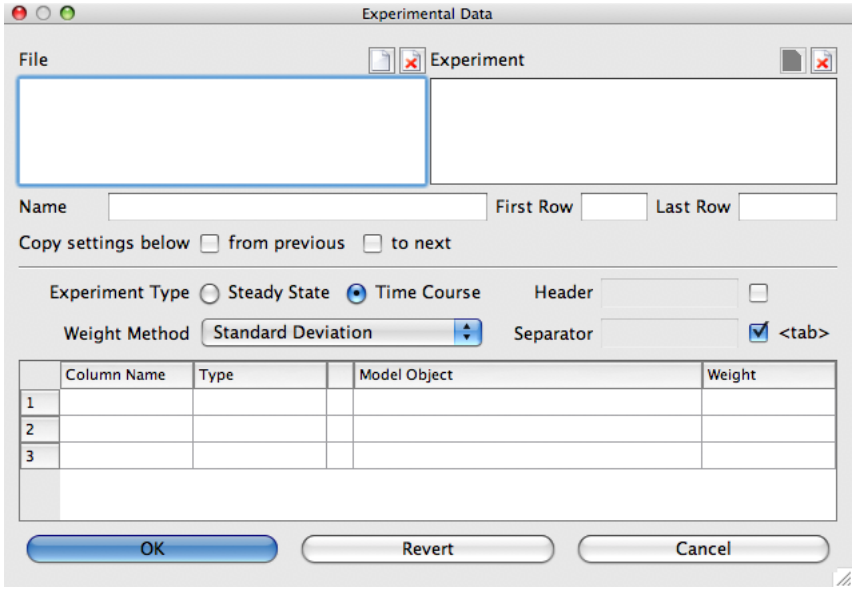
Where P is the currently tested parameter set, $x_{i,j}$ is a point in the dataset, and $y_{i,j}(P)$ the corresponding simulated value. The indices i and j denote rows and columns in the dataset. The weight for each data column is given by ω_j . COPASI provides 3

methods shown in the table below to calculate the weights for you. After applying the method chosen COPASI scales the weights so that for each experiment the maximal occurring weight is 1. In case that the weights calculated are not satisfactory you are able to manually override them individually.

Name	Formula
Mean	$\omega_j = \frac{1}{\lvert<x_j>\rvert}$
Mean Square	$\omega_j = \frac{1}{\sqrt{<x_j^2>}}$
Standard Deviation	$\omega_j = \frac{1}{<x_j^2>-<x_j><x_j>}$

Weight Calculation Methods

To specify the experimental data you click on the Experimental Data button at the top right of the parameter estimation dialog. A new dialog opens that lets you enter experimental data.



Experimental Data Dialog

To read a data file, click on the open button beside the label File at the top of the dialog and choose a file that contains experimental data from the file dialog. The data file should contain experimental data grouped in experiments. To support automatic detection of experiments these must be separated by one or more empty lines. But manually definition of experiments is allowed. The data for an experiment must be a table of values. The columns of the table are separated by a user specifiable separation character. The default and recommended character is the <tab>-character. The first line of each experiment is treated as the row containing the column headings. However, this is only a default and the header row can be specified by the user. The header row may be anywhere in the file the data is contained. The purpose of the header row is to ease the interface to the data file and may be omitted. To tell COPASI that no header row is included uncheck the box next to the header. Once COPASI has read a file, you have to specify some information for each experiment included in the file. To select an experiment you choose it from the right selection box. The first thing you need to specify is whether the data belongs to a Steady-State analysis or to a time course simulation. You also have to associate the individual columns of input data to elements of the model. For this, you click on the ... button in each row and select the corresponding object in the selection dialog. It is mandatory that COPASI knows about the meaning of each data column. The data in a column can have four different types, which are:

ignored Values in columns marked ignored are not taken into account during parameter fitting. Columns of this type may not be associated with elements of the model.

independent Independent data is data which needs to be set for the correct simulation of the experiment row. Possible model elements are initial concentrations or kinetic parameters. Note, for a time course experiment only the independent data in the first data row is set before the start of the simulation. Columns of this type must be associated with elements of the model.

dependent The dependent data is the data, which COPASI tries to fit by minimizing the sum of squares between the simulated data and the experimental data. Columns of this type must be associated with elements of the model.

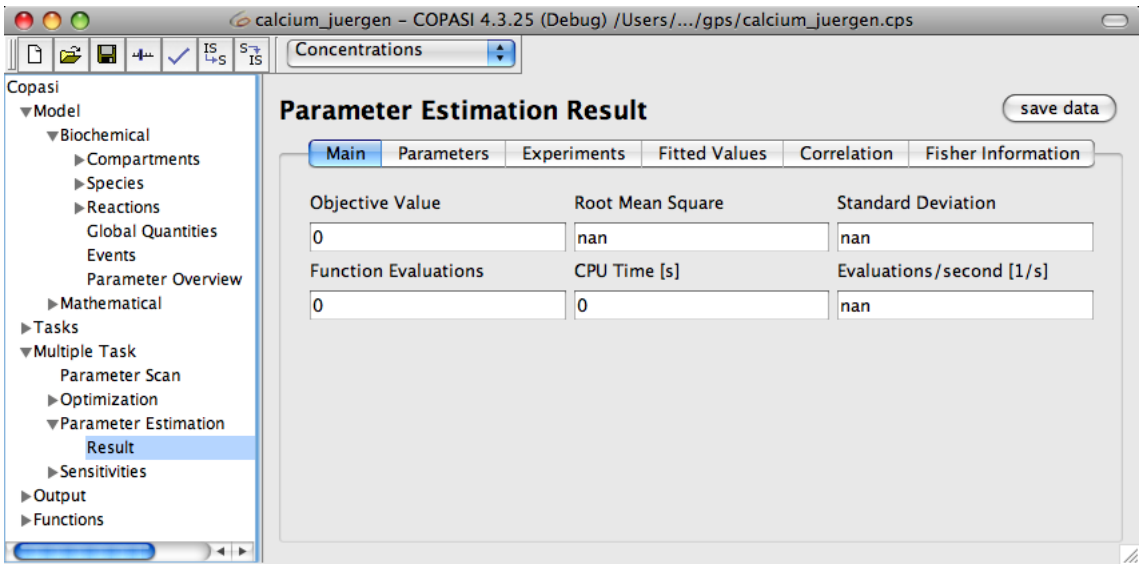
Time This column type is only available for time course experiments. Obviously only one column with this data type may exist. COPASI attempts to automatically identify the column containing the time by looking at the column headers. You may correct COPASI's guess. This column may not be mapped to any model elements.

If you don't want COPASI to use the whole dataset of an experiment, but only a subset, you can specify the start and end line for this subset. You also may delete experiments completely. If you do so, you may notice that the New Document will be enabled. Pressing it will add the first not used experiment of the currently active file. Since it is commonly the case that all experimental data within one file has the same format, COPASI allows you to copy information of experimental data from the previous to the current or form the current to the next experiment within a file by selecting from previous and to next. If COPASI detects that experimental data descriptions are identical it will automatically set the from previous check box and disable editing the current experiment. Should you want to modify it you will have to unmark the check box first.

If you have more than one file, you can load additional data files and process them in the same manner. Once you are finished defining your datasets for the fitting, you close the data dialog with the OK button. Before you can start the parameter estimation process, you have to choose the method by which the fitting will be done and maybe set some method parameters. Most of the time, the default parameter values should do. The method choosing is done at the bottom of the dialog by selecting the method from the drop down list. For an explanation of the individual methods, please consult the methods section.

3.9.2 Result

If you want to have output from the parameter estimation task, you have to create an output definition as described in the output section or you choose the default report named "Parameter Fitting". The default reports prints a description of the settings you provided for this parameter fitting run, intermediate results every time the target function improves, and a detailed result at the end. The easiest way to get a customized output is probably to use the output assistant which you activate via the Output Assistant button. All that is left to do in order to write the output to a specific file is to connect an output definition with a file. This can be achieved by clicking on the Report button. This opens a dialog that lets you connect the report for a specific task to a file on your hard disk. First we choose a report that is suitable for the parameter estimation task from the drop down list at the top of the dialog. Next, we specify a file that will be used to store the report by clicking on the browse button and selecting the destination in the file dialog that opens. Per default, COPASI creates a new file or overwrites an existing file with the same name. Alternatively, you can tell COPASI to append the report to the end of an existing file by selecting the corresponding check box labeled Append at the bottom of the dialog. Once you are finished, you click on the Confirm button. If you now run the task, COPASI will write the output to the file you specified.



Parameter Estimation Results

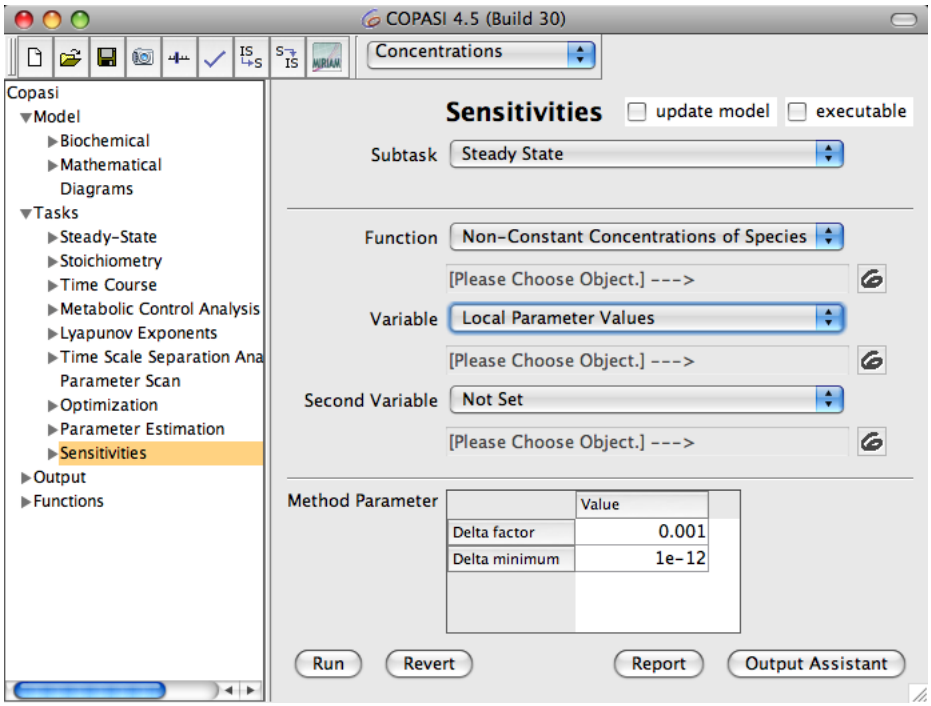
After running a Parameter Estimation task the result may be viewed by selecting the Result widget. This result widget contains multiple tabs. The overall fit and performance statistic are displayed in the Main and detailed information about parameters, experiments, and fitted values can be found under the corresponding tabs. In addition, you may look at the correlation matrix of the parameters or the Fisher information matrix.

3.10 Sensitivity Analysis

COPASI also allows the calculation of sensitivities of the model with respect to various parameters. Generally a sensitivity is a measure for how much a specific "observable" (this means any number that can be obtained by numerical analysis of the model) changes when a given parameter is changed.

In COPASI it is possible to calculate whole arrays of sensitivities for lists of "observables" with respect to lists of parameters. The settings for the sensitivities calculation can be found under "Tasks->Sensitivities". Let´s look at an example:

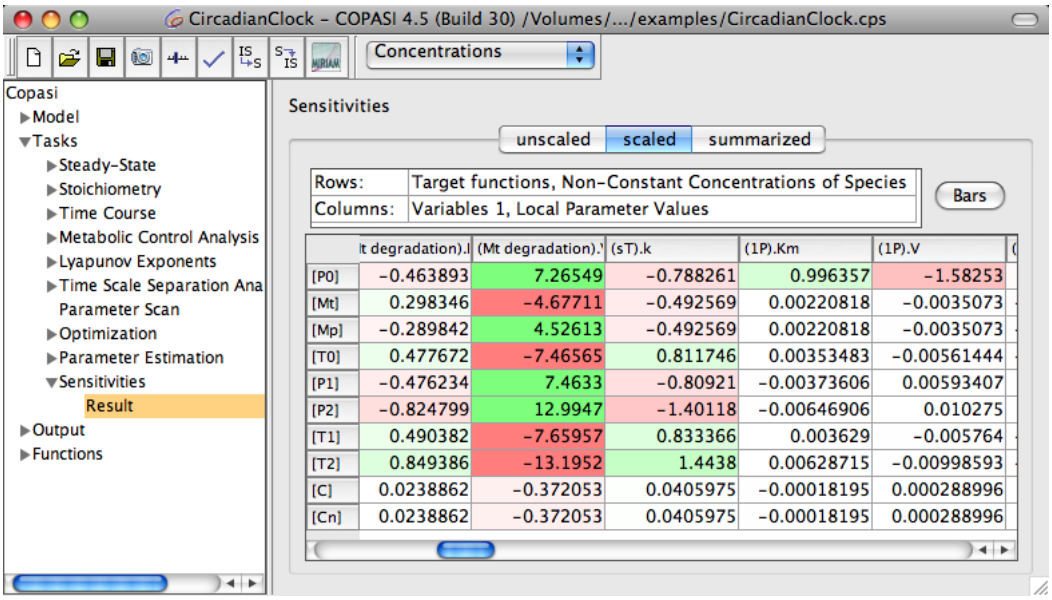
If you have a mode for which you can calculate a steady state you may be interested in how the steady state concentrations change with respect to different kinetic parameters. For this you could choose the following settings:



Sensitivity Analysis dialog

- For "Subtask" choose "Steady State". This means that you are interested in the sensitivities of a result of a steady state calculation.
- For "Function" choose "Non-Constant Concentration of Species". "Function" here indicates the observables, i.e. the values of which you want to know the sensitivities. We want to calculate the sensitivities of the steady state concentrations.
- Set "Variable" to "Local Parameter Values". Local parameters are all kinetic parameters that are directly specified for the reactions in the model.

To start the calculation just click on the run button.



Sensitivity Analysis result

The results will be displayed as color coded tables of numbers.

Chapter 4

Importing and Exporting

COPASI is one modeling and simulation tool among many and often people use more than one tool to create, investigate and simulate their models. In order to make it easier to exchange model files with other programs, COPASI supports reading and writing of several model file formats. Naturally COPASI can read and write it's own file format and it can read Gepasi files just as well although it will not write Gepasi files.

In addition to those two file formats, COPASI can import SBML Level 1 and 2 files and it can export models as SBML Level 2 files as well as C source code, XPPaut, or Berkeley Madonna files.

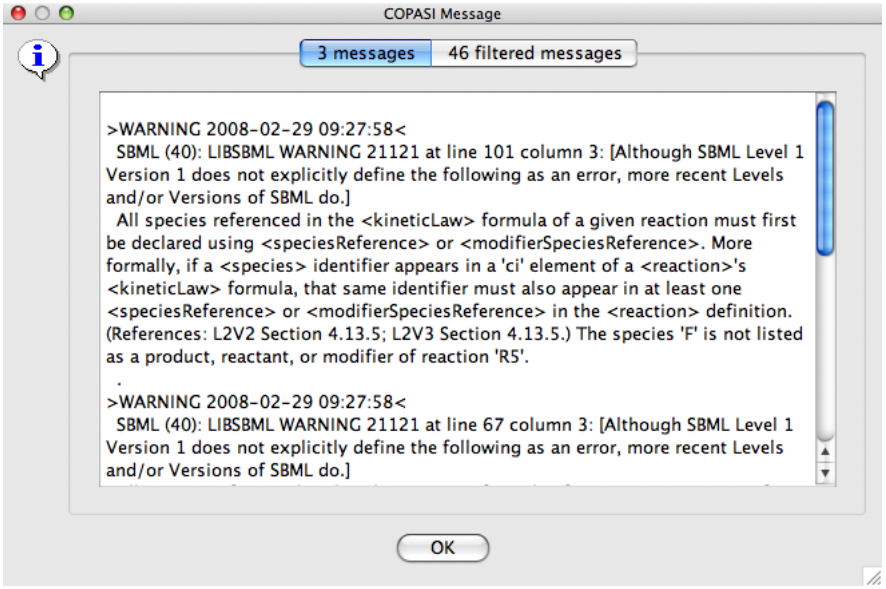
4.1 Importing and Exporting SBML files

COPASI is able to import SBML level 1 and level 2 files as well as export SBML level 2 files through the corresponding entries in the File menu. Export of SBML files to Level 1 Version 2 is also enabled, but since SBML Level 1 Version 2 does not support many features of SBML Level 2 and above, export to this level and version will often fail and COPASI should tell you why the export failed. For the import and export the SBML Model is read with libsbml and converted to the COPASI model structure and vice versa. On exporting, COPASI converts its native model structure to an SBML model that is again written out using libsbml. Since the SBML model structure is converted into the COPASI model structure upon import, some of the information in the SBML file gets lost because COPASI does not support the corresponding model elements. Examples of data that gets lost are e.g. algebraic rules rules.

Notes and Annotations from the original SBML file should be preserved when you import and reexport an SBML file as long as you do not delete the objects that contain the annotation in COPASI. But still if you have important annotations or notes in your SBML file make a backup of the file before you open it in COPASI since we can not guarantee that they will survive an import/export cycle.

Likewise, SBML does not support all of the elements of a COPASI model so some information from the COPASI model also gets lost when exporting an SBML file. For example tasks, reports and plot definitions are not exported to the SBML file. This is normally not a big problem since the essential parts of a model normally get imported or exported. A consequence however one should keep in mind is that if you import an SBML file into COPASI and later export it again, it might have lost some of the original information. So if your SBML model depends on events, it is not suitable for import into COPASI. This will change in future versions of COPASI.

Current versions of COPASI use a modified libsbml version 3.4.1 which supports SBML files up to Level 2 Version 4. Libsbml 3 also has a lot of built-in tests to check the validity of a model. Upon import COPASI will display most of those errors to the user. Libsbml 3 also gives a lot of warnings, especially concerning units in SBML files, which should help the user in generating better, more consistent models. Even so this is a good thing, it can happen that the real errors are hidden by the large amount of warnings. COPASI now uses a new error message dialog that splits the messages in filtered and unfiltered messages. The unfiltered messages are what the user sees when the dialog comes up. In order the see the filtered messages the second tab in the message dialog has to be selected. If there are no filtered messages, this tab is disabled. In the labels of the individual tabs, COPASI also shows you the number of messages that for the corresponding tab.



Error Message Dialog after import of an SBML File

In the current version, the errors that are filtered are hard coded and they are all warnings that have to do with unit inconsistencies within the model. In future version of COPASI, we will have a mechanism that will let the user manage which error messages should be filtered.

Although COPASI now uses libsbml 3, not all features found in the different versions of SBML are currently supported. If an unsupported feature is encountered during import, COPASI will usually notify the user. Depending on which parts of the model have been ignored, ignored, the result of time course simulations and other tasks might not be what you expect.

One should also be aware of the fact that in COPASI kinetic laws are always functions and a reaction contains a call to one of those functions. Since SBML allows arbitrary mathematical expressions as kinetic laws of reactions, those expressions are converted to user defined functions upon import and the reaction then calls this function. When reexporting such a reaction to SBML, the kinetic law expression will be exported as a function definition and a call to that function in the kinetic law of the reaction.

One notable feature from SBML that is currently not fully supported in COPASI is the delay function. SBML models with calls to the delay function can be imported in COPASI, however, the delay implementation in COPASI will always return NaN for any call to the delay function. In most cases this means that you do not get correct results when working with such a model in COPASI. The reason for importing those models anyway is to allow you to modify the model, meaning to get rid of the delay calls in which case you can work with the model in COPASI.

Another issue with the delay function in COPASI is the fact that COPASI does not allow a call to the delay function to appear in a function definition. This includes function definitions that are created from kinetic law expressions upon import (see above). If COPASI encounters a call to the delay function in a kinetic law expression, the delay call is converted to a global variable which is then passed as an extra argument to the kinetic function. If there are references to local variables within a call to the delay function, those local parameters are also converted to global parameters. If you reexport such a model to SBML, all those changes remain in the model. However, those changes do not change the results you get when you e.g. simulate the model.

4.2 Exporting C Source files

Sometimes it is of advantage to have the differential equations that make up your model in the form of source code for some programming language. This allows you to integrate your model into some experimental analysis software that you might have written, or some analysis software that expect the input as C source code like Auto2000.

So far COPASI only supports the export of C source code, source code for other programming languages might follow if there is need for it.

The C source file COPASI exports is split into several smaller parts which are encapsulated in `#ifdef` structures for the C preprocessor. The file consists of ten such sections which can be included in other files by defining the corresponding preprocessor constants at the place of inclusion. Example:

```
#define SIZE_DEFINITIONS
#include "SOURCE FILE"
#undef SIZE_DEFINITIONS
```

The file contains the following sections:

Name	Description
SIZE_DEFINITIONS	Contains size definitions for the individual model elements, e.g. N_COMPARTMENTS for the number of compartments or N_REACTIONS for the number of reactions. All definitions are declared with preprocessor #define statements.
SPECIES	Contains assignments for the species initial concentrations. The assignments are of the form y[INDEX]=VALUE where INDEX is the index of the species and VALUE is the initial concentration. This assumes that the including code has generated an array of double values of size N_METABS called y. N_METABS is part of the SIZE_DEFINITIONS section (see SIZE_DEFINITIONS above).
INDEP_SPECIES	Contains assignments for the independent species initial concentrations. The assignments are of the form x[INDEX]=VALUE where INDEX is the index of the independent species and VALUE is the initial concentration. This assumes that the including code has generated an array of double values of size N_INDEP_METABS called x. N_INDEP_METABS is part of the SIZE_DEFINITIONS section (see SIZE_DEFINITIONS above).
MOIETY	Contains assignments for the moieties. The assignments are of the form y[INDEX]=RSIDE where INDEX is the index of the species and RSIDE is the calculated value for the moiety, e.g. 0.2 - y[INDEX2]. y is the vector of species (see SPECIES above).
COMPARTMENTS	Contains assignments for the compartments initial volumes. The assignments are of the form c[INDEX]=VALUE where INDEX is the index of the compartment and VALUE is the initial volume. This assumes that the including code has generated an array of double values of size N_COMPARTMENTS called c. N_COMPARTMENTS is part of the SIZE_DEFINITIONS section (see SIZE_DEFINITIONS above).
GLOBAL_PARAMETERS	Contains assignments for the values of global kinetic parameters. The assignments are of the form gk[INDEX]=VALUE where INDEX is the index of the global parameter and VALUE is the value. This assumes that the including code has generated an array of double values of size N_GLOBAL_PARAMS called k. N_GLOBALPARAMS is part of the SIZE_DEFINITIONS section (see SIZE_DEFINITIONS above).
KINETIC_PARAMETERS	Contains assignments for the values of local kinetic parameters. The assignments are of the form k[INDEX]=VALUE where INDEX is the index of the local parameter and VALUE is the value. This assumes that the including code has generated an array of double values of size N_KIN_PARAMS called k. N_KIN_PARAMS is part of the SIZE_DEFINITIONS section (see SIZE_DEFINITIONS above).
KINETIC_FUNCTIONS_HEA	This section contains the declarations of function definitions. The section has to be included before the function definitions section itself and before the differential equations section is included.
KINETIC_FUNCTIONS	This section contains the implementation of the function definitions.
DIFFERENTIAL_EQUATIONS	This section contains the set of differential equations. The result of the right hand side of each differential equation is stored in a variable dxdt[INDEX]. This assumes that the including program has created an array of double values called dxdt of size N_INDEP_METABS. N_INDEP_METABS is part of the SIZE_DEFINITIONS section (see SIZE_DEFINITIONS above).

C Code Export Sections

Caution Some models may contains constants or functions that are not included in the ANSI C standard. In order to be able to use exported C source code files with those constants and functions, the user has to provide them, e.g. by including a separate header file that defines those constants and/or functions. Currently those are the constants for pi, Euler's number, TRUE, FALSE and infinity as well as the functions "asinh", "acosh", "atanh", "sec", "csc", "cot", "sech", "csch", "coth", "arcsec", "arccsc", "arccot", "asech", "acsch", "acoth", "factorial" and logical "xor".

In order to export the set of differential equations to a C source code file, you select the Export ODEs menu entry from COPASI's File menu. In the save dialog that shows up, you select *C Files (*.c)* from the File type drop down and specify the name of the file you want to write the ODEs to. After clicking on the Save the ODEs will be saved to the specified file.

4.3 Exporting Berkeley Madonna files

Berkeley Madonna is a widely used commercial modeling and simulation tool for the Windows and Mac OS X operating systems. The input to Berkeley Madonna is a file that basically consists of a set of parameters, initial values, function definitions and ordinary differential equations. COPASI can generate these ordinary differential equations from the reactions of the model and export them together with the needed parameters in a format suitable to be read into Berkeley Madonna.

Since the way ordinary differential equations are specified is similar between different programs that take such input, it should be rather easy to adjust the exported file for import in other programs as for example the free XPPaut program or even Mathematica.

Caution Berkeley Madonna does not support all mathematical functions available in COPASI if such a function is used in a model, COPASI will write the string "ILLEGAL FUNCTION" to the exported file instead of the name of the function. Naturally this will lead to an error message when one tries to load the file in Berkeley Madonna. Functions currently not supported by Berkeley Madonna are: sec, csc, cot, sech, csch, coth, arcsec, arccsc, arccot, arcsech, arccsch, arccoth, floor, ceil, factorial, modulus and logical xor.
The Constants for TRUE, FALSE, and Euler's number will be exported as numerical values where FALSE will be exported as 0 and TRUE will be exported as 1. Likewise infinity and "not a number" are exported as the strings "inf" and "nan" respectively.
COPASI adds comments to variables in the ODE by using the semicolon to separate the comment from the actual code. Since this feature has been added to Berkeley Madonna in Version 7.0 the exported code cannot be used with older versions of Berkeley Madonna.

In order to export the set of differential equations to a Berkeley Madonna file, you select the Export ODEs menu entry from COPASI's File menu. In the save dialog that shows up, you select *Berkeley Madonna Files (*.mmd)* from the File type drop down and specify the name of the file you want to write the ODEs to. After clicking on the Save the ODEs will be saved to the specified file.

4.4 Exporting XPPAUT files

XPPAUT is a program to to solve differential equations. It is freely available for Windows, MacOS X, Linux/Unix and probably others. Its input format is very similar to the one from Madonna and COPASI starting from Version 4.1 Build21 can export models as a set of differential equations for use in XPPAUT.

In order to export the set of differential equations to an XPPAUT file, you select the Export ODEs menu entry from COPASI's File menu. In the save dialog that shows up, you select *XPPAUT (*.ode)* from the File type drop down and specify the name of the file you want to write the ODEs to. After clicking on the Save the ODEs will be saved to the specified file.

Caution Just as in Berkeley Madonna, XPPAUT does not support all functions supported by COPASI. Functions that are not supported are exported as @. Among the unsupported functions are *sec*, *csc*, *cot*, *sech*, *csch*, *coth*, *arcsec*, *arccsc*, *arccot*, *arcsech*, *arccsch*, *arccoth*, *factorial*, *modulus* and *logical xor*.
The values of *TRUE* and *FALSE* are converted to *1* and *0* respectively upon export. *INFINITY* and *NAN* are exported as *INF* and *NAN* . Since both *INF* and *NAN* are not supported keywords in XPPAUT, those files have to be modified before they can be used in XPPAUT.
We also note that also *ceil* was mentioned in the XPPAUT documentation, files containing this function could not be loaded in XPPAUT.

Chapter 5

Diagrams

Starting with version 4.4.28, COPASI has the possibility to read and display layout information from COPASI and SBML files. An additional feature that comes with the diagram display is the possibility to display animations of time course data from simulations. Currently there is no way to create new diagrams within COPASI, but we hope to change that in the future. The layout information can either be obtained in the form of the SBML layout extension (see the SBML Layout Extension Specification for further information) or from a COPASI file that includes layout information. Since there is no way to create new diagrams in COPASI, the only way to come up with a COPASI file that includes layout information is to create an SBML file with layout information and convert it to a COPASI file. The layout in COPASI files is also stored in the form of the SBML layout extension. Only that the SBML object ids are replaced by the corresponding COPASI keys.

There are several ways to create layout information for an SBML model. The Sycamore web application allows you to load SBML files, create layout information for the model in the file and save the model with the layout information back to file. It provides different methods for generating the layout and the layout can be modified manually as well.

SABIO-RK is a reaction kinetics database that allows you to create SBML models from the entries in the database. The model can be saved as an SBML file and it can also create layout information for the model.

Another web application that allows the user to add layout information to an SBML file is Frank Bergmanns SBML Layout Viewer. It allows the user to influence the layout creation by setting some parameters.

Franks Bergmanns layout creation tool is also part of SBW and since the latest Versions of COPASI do include some SBW support, it is possible to send a model from COPASI to the layout tool and get back a model with layout information.

If the model file you loaded contains layout information, the diagram table contains an entry for every layout that is contained in the file. In order to display a specific layout, you have to either click on the show button beside the layout or double click on the layout entry in the table.

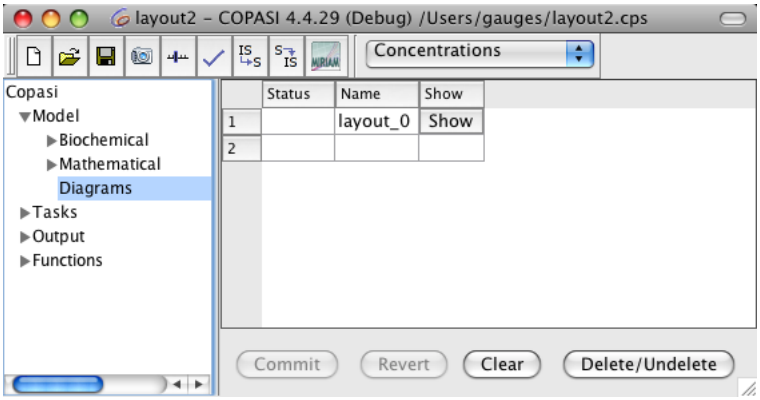


Diagram Table with one Diagram Entry

In the layout window that comes up, you can see some controls on the left side and on the right side you see the actual layout. When the layout is first opened, all species are displayed as rectangles.

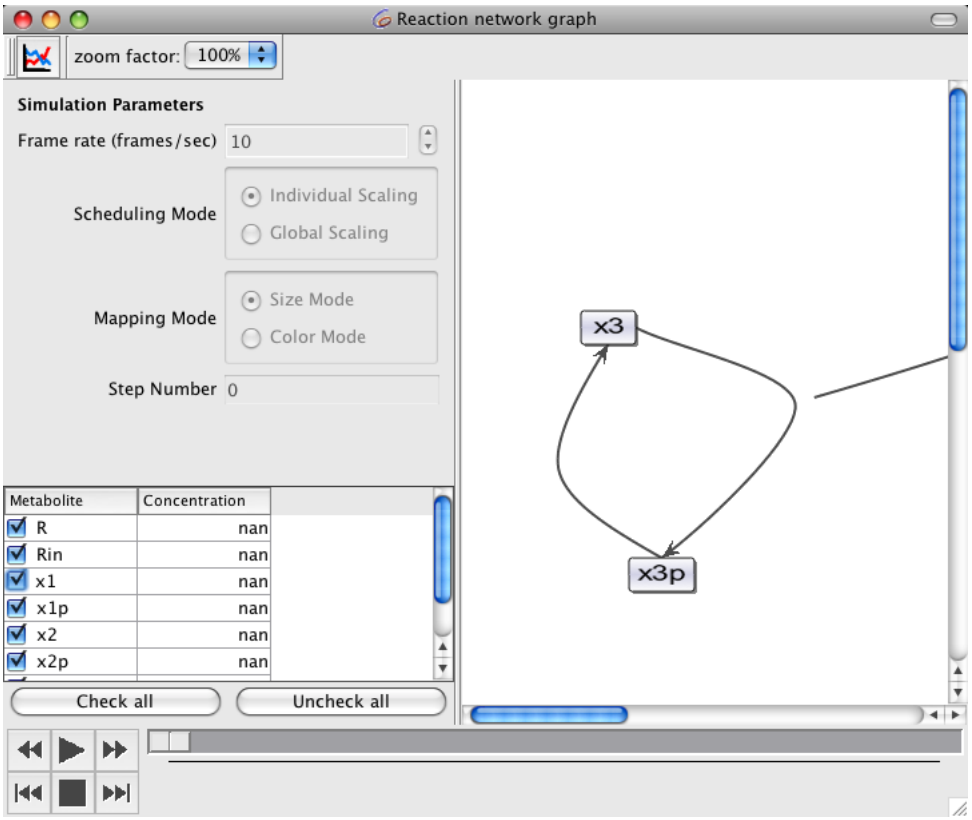


Diagram before loading Simulation Data. Some Controls are disabled.

Currently the menu bar at the top contains two control elements. The first one is a button that allows you to read time course data from COPASI if any has been calculated. If a time course has already been calculated when the layout window is opened, this time course data is imported automatically. Each time you rerun the time course simulation in COPASI, you have to reimport it to the layout if you want to display it.

Since the time course data is copied when it is imported to the layout, you have to take care that you have enough memory in your machine if you either have very large simulations or if you want to animate the data in many different layout windows. In the future we will try to handle this more intelligently.

The second control element in the toolbar is for zooming in and out of the layout view.

If there is no time course data yet, many of the controls on the right side are disabled since they only make sense in the context of animating simulation data. Once simulation data has been loaded via the button in the toolbar, those controls are enabled.

If you don't need the controls, but just want to look at the diagram, the individual controls can be hidden via the View menu in the menu bar.

In order to animate your simulation data, you can use the player controls at the bottom of the screen. The controls look like the controls on a CD player and it should be easy to figure out what the individual buttons do. There are buttons to start and stop the animation, as well as buttons to forward and rewind or single step through the animation. You can also drag the slider beside the buttons to go to a certain time point within the animation.

During the animation, the species are displayed as colored balls. The speed of the animation can be controlled by the Frame rate element at the top of the controls.

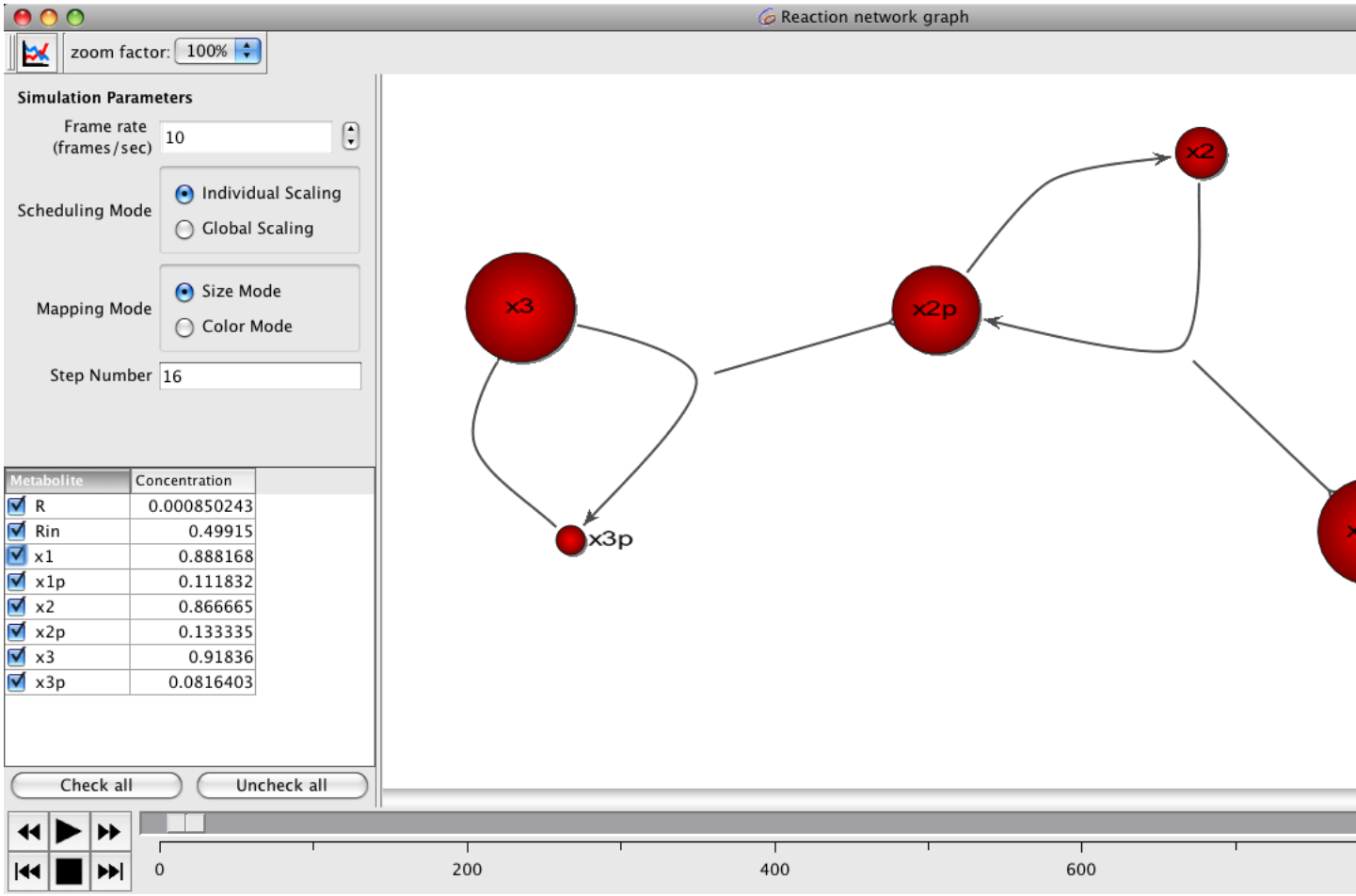


Diagram Window with paused Animation in Size mode.

Depending on the Mapping mode that is controlled by the radio button group with the same name, either the size of the balls or the color changes to reflect the concentration of the species at a given time point.

The radio button group above the Mapping group, called Scheduling Mode determines how the size or color of a species is determined during an animation.

If the control is set to *Individual Scaling*, the size (or color) of a species is determined by the minimum and maximum value of that species. If the control is set to *Global Scaling*, the size or color is determined by the minimum and maximum values of all species in the model.

The last control element is the table with the names and concentrations of all species contained in the layout. The concentration displayed is the concentration of the species at the given time point of the simulation. All species that are checked vary their size or color during the animation, species tha are unchecked don't.

There is also the possibility to export the current layout as a bitmap. In order to do this, you select the Create Image menu item from the View menu. Currently, this only allows you the create a bitmap of the layout as it is displayed. Parts that are not displayed because the window is to small are not written to the bitmap. So in order to make a bitmap from large layouts, you might have to hide the controls as described above, make the layout window as large as possible and change the zoom factor until the full layout is displayed in the layout window. We also hope to make this more convenient in the future.

Since this is an early version of the diagram display, there certainly are still many issues, but we hope that it might be helpful nevertheless. If you have suggestions or complaints, please tell us, preferably via the User Support Forum or if you think it is a bug via our Issue Tracker.

Chapter 6

The Model in COPASI

This section discusses the general properties of the model in COPASI and its mathematical interpretation. Generally a model consists of species which are placed in a compartment. Species are produced or consumed by reactions which happen with a speed given by kinetic functions. This model is interpreted mathematically in different ways in COPASI.

6.1 Compartments

Every species in COPASI must be placed in a compartment. The compartment has to have a well defined volume because COPASI relies on being able to calculate particle numbers from concentrations and vice versa. COPASI does not really support models without concentrations (i.e. dealing only with amounts of substance or particle numbers) but you can work around this limitation by setting the volume to be 1.0. But if you do this keep in mind that the dimensions of the different parameters as displayed in the COPASI GUI are not correct. The volumes of compartments can be fixed (the default) or determined by an assignment or an ODE.

If the (chemically) same species is present in several compartments it is treated as several species in COPASI. E.g. if you have a model that contains glucose inside a cell and outside a cell these are two species in the model. They can however have the same name.

6.2 Species

A species is characterized by how much of it is present at a given time. This can be expressed as a particle number or as a concentration. COPASI makes sure that those two are consistent at any time. An initial value can be provided for concentration or particle number of each species. This initial value will be used as a starting point for simulations or other calculations. A species can have four different modes (called "types" in the GUI).

Species determined by reactions This is the default case. The value (concentration/particle number) of these species changes according to the reactions in which the species participate as substrates or products.

Fixed species "Fixed" species have constant particle number. This means that if the volume of the compartment containing the species is not changing, the concentration of the species will also be constant. For variable compartment volume the concentration will change. "Fixed" species can participate in a reaction, and can influence the rate of this reaction, but their concentration/particle number will not be changed by the reaction.

Species with assignment The Concentration of a species can also be determined by an explicit mathematical expression. In this case the concentration of the species is determined by evaluating the mathematical expression (which could be constant, time-dependent, or dependent on other variables of the model) and the particle number is then calculated from the concentration. The particle numbers/concentrations of such species will not be influenced by reactions in which they participate.

Species with ordinary differential equation The modeler can also explicitly provide the right hand side of an ordinary differential equation for the species. The expression for the differential equation has the dimension of concentration per time, but it is important to note that it does *not* specify the rate of change of the concentration. Instead, the expression is internally multiplied by the compartment volume and then interpreted as the rate of change of the amount of substance of the species.

Associated to the species is also a value called transition time. It is calculated as the current particle number of the species divided by either the sum of all reaction fluxes going into the species or the sum of all reaction fluxes going out of the species, whichever is smaller (the fluxes are also expressed in number of particles per second). The transition time gives a rough heuristic measure for how long a particle of this species will exist on average before it is consumed. A short transition time in a stable steady state may indicate that the equilibrium is a "fast" equilibrium which could be used for a simplification of the model. Note however that this calculation is inexact if any reversible reactions are involved.

6.3 Global Quantities

The model also contains a list of global quantities. Roughly speaking the global quantities can be used to describe values that are not necessarily a concentration or a volume. But to be more specific: The global quantities can have three different modes (called "types" in the GUI).

Fixed quantities "Fixed" quantities always take a constant value. While the GUI still displays an initial value and a transient value those are always the same. This can be used if the same numerical value is used in several parts of the model (e.g. several reactions use a common kinetic parameter) but you want to be able to change this value in a single place.

Quantities with assignment Global quantities can also have an assignment rule. This means a mathematical expression is specified (which can involve other variable or constant values from the model) and the value of the global quantity is alway the value of this expression. Circular definitions are not allowed. Global quantities with assignment do not have an initial value (since their value is always, also at the beginning of a simulation, determined by the mathematical expression). These assigned quantities could be used for output only (e.g. if you want to plot the sum of some concentrations) or can be used in the model itself.

Quantities with ordinary differential equation If global quantities are set to the "ode" mode they become true variables of the model. A mathematical expression has to be specified and this expression is interpreted as the right hand side of an ordinary differential equation for this variable. An initial value has to be provided for these quantities.

Since there is no obvious probabilistic interpretation of a general ode stochastic simulation is currently disabled for models containing global quantities of type "ode".

6.4 Reactions

A reaction is a process by which species will be consumed or produced. It is characterized by a description of how fast this happens. It contains a list of substrates, i.e. the species that are consumed if the reaction takes place, along with the information about how many molecules of each substrate are consumed when the reaction event happens once (the stoichiometry). Correspondingly there is a list of products with the respective stoichiometries. Reactions without substrates are possible, as well as reactions without products. However reactions without both substrates and without products are not allowed. In addition so called modifiers can be specified which are neither produced nor consumed in the reaction but which have influence on the speed of the reaction.

The speed of the reaction is always specified by a reference to a kinetic function. The kinetics can depend on the concentrations of the substrates, products, and modifiers, on the volume of a compartment, on local or global parameters, and on the simulation time. The difference between local and global parameters is that local parameters only specify a numerical value of a kinetic parameter for one specific reaction. Global parameters can be used in several reactions.

Reactions can be reversible or irreversible. Kinetic functions for irreversible reactions should always be positive. Also they should not depend on the concentration of the products (only on the concentrations of the substrates and modifiers). While all built in kinetic functions satisfy these conditions, they are not enforced for user defined functions.

6.5 Functions

A function in COPASI is a mathematical expression that calculates a value from a given list of other values. Generally functions can be used in two ways in COPASI: As a kinetic function specified for a reaction or as a function that is called from another function or expression.

The function contains a detailed list of parameters that specifies how many values (and what kind of values) need to be passed to the function. If a function is called from another function or expression only the number of values is checked, not the kind of value. In the case that a function is used as a kinetic function for a reaction COPASI ensures consistency about the roles of the parameters in the model. Each parameter has one of six different roles: "Substrate", "Product", "Modifier", "Volume", "Time", "Parameter". The first three must be connected with the concentration of a species that has the respective role in the reaction. "Volume" must be connected with a compartment, "Time" is always the simulation time of the model. "Parameter" must be connected with a global parameter or a local value in the reaction.

A function also specifies if it is appropriate for a reversible or irreversible reaction. This information is not automatically inferred from the mathematical description of the function, and COPASI does not enforce restrictions like irreversible kinetics having to be strictly positive.

6.6 Deterministic Interpretation of the Model

One possible mathematical interpretation of the model is to convert it into a set of ordinary differential equations. The variables of the equation are the particle numbers of the species in the model. The right hand side of the differential equation are constructed as follows: The particle numbers are converted to concentrations taking into account the unit for amounts of substance and dividing by compartment volume. These concentrations are used to calculate the reaction fluxes. The kinetic functions, as they are defined in COPASI give as result a value that is a concentration rate (for single compartment reactions) or an amount of substance rate (for multi-compartment reactions), respectively. So for single compartment reactions the value of the kinetic function is multiplied by the compartment volume; kinetics for multi-compartment reactions are assumed to already be expressed in units of amount of substance (e.g. moles) per time. The resulting value is then multiplied by a factor to convert amount of substance per time to particle numbers per time. Linear combinations of these values, using the stoichiometries as coefficients, result in particle number rates for all species. These form the right hand side of the differential equations.

COPASI automatically performs an analysis of the model by which conserved values are found. The conserved values COPASI is looking for are linear combinations of particle numbers that do not vary during the time evolution of the system. Each conservation relation can be used to eliminate one variable of the system, leading to a reduced system with a smaller number of variables. These variables are called the independent variables of the system; the dependent variables are defined as linear combinations of independent variables. COPASI handles this model reduction transparently, but it is displayed in the GUI which species are treated as independent or dependent variables. Technically finding the conservation relation means finding rows in the stoichiometry matrix that can be expressed as linear combinations of other rows. COPASI uses Householder QR factorization [Vallabhajosyula06] to do this.

6.7 Stochastic Interpretation of the Model

An alternative interpretation is to consider the model as a stochastic process. In this case the reaction kinetics are not considered to describe the rates of change for the concentrations of involved species, but rather as a specification about the probability that a reaction event happens. If a reaction event happens the particle numbers of the involved species are updated according to their stoichiometries. That means particle numbers are always integer numbers and change discretely.

Specifically the value of the kinetic function is interpreted as a so called propensity, that is a differential probability density that a reaction event will happen in the next infinitesimal time interval. However there are subtle differences between reaction rates and reaction propensities. One of those differences that only matters for rather small particle numbers is that e.g. the rate of a second order mass action reaction is described as $k\left(\frac{S}{V}\right)^2$, while the propensity of the same reaction is $k\frac{S}{V}\frac{S-1}{V}$. COPASI will apply this kind of corrections automatically. In cases where these corrections have already been done by the modeler explicitly COPASI needs to be told not to apply this correction. This is described in the general model settings.

Another issue modelers should be aware of is that the rate laws for enzymatic reactions that are derived using the steady state approximation are not necessarily valid for stochastic simulation. In many cases they are, but the underlying assumptions for using them are not exactly the same as for deterministic simulations.

Chapter 7

Methods

7.1 Time Course Calculation

With the time course simulation, you can calculate the trajectory for the species in your model over a given time interval. There are different methods to calculate such trajectories and depending on your model, one or several of them may be appropriate to do a time course simulation of your model.

COPASI supports three different methodologies to calculate a trajectory. The first method is to do a deterministic time course simulation of your model using the LSODA [Petzold83] algorithm. For systems with small particle numbers, it is sometimes better to do a stochastic simulation rather than a deterministic one. COPASI supports a method for the stochastic calculation of time series, which is called stochastic and uses the next reaction method described by Gibson and Bruck [Gibson00].

Since the deterministic simulation is inappropriate for some systems but on the other hand, the stochastic simulation is too time consuming, there are some methods that try to combine the advantages of both deterministic and stochastic simulation. Most of those methods are termed hybrid methods. COPASI also includes such a hybrid method which in some systems where deterministic simulation would lead to incorrect results will give the correct time series but is still computationally less demanding than a pure stochastic simulation.

7.1.1 Deterministic Simulation

7.1.1.1 Deterministic (LSODA)

The default method in COPASI to calculate a time course is LSODA [Petzold83]. LSODA is part of the ODEPACK library [Hindmarsh83]. LSODA was written by Linda R. Petzold and Alan C. Hindmarsh.It solves systems $\frac{dy}{dt} = f(t,y)$ with a dense or banded Jacobian when the problem is stiff, but it automatically selects between non-stiff (Adams) and stiff (BDF) methods. It uses the non-stiff method initially, and dynamically monitors data in order to decide which method to use. Options for LSODA

Integrate Reduced Model This parameter is a boolean value to determine whether the integration shall be performed using the mass conservation laws, i.e., reducing the number of system variables or to use the complete model. A value of '1' (the default) instructs COPASI to make use of the mass conservation laws, whereas a value of '0' instructs COPASI to determine all variables through ODEs.

Relative Tolerance This parameter is a numeric value specifying the desired relative tolerance the user wants to achieve. A smaller value means that the trajectory is calculated more accurate. The default value is $1.0 \cdot 10^{-6}$. Please note that best achievable relative tolerance is approximately $2.22 \cdot 10^{-6}$.

Absolute Tolerance This parameter is a positive numeric value specifying the desired absolute tolerance the user wants to achieve. Please note that for species the absolute tolerance is applied to the concentration value. The default value is $1.0 \cdot 10^{-12}$.

Adams Max Order This parameter is a positive integer value specifying the maximal order the non-stiff Adams integration method shall attempt before switching to the stiff BDF method. The default and maximal order is ’12’.

BDF Max Order This parameter is a positive integer value specifying the maximal order the stiff BDF integration method shall attempt before switching to smaller internal step sizes. The default and maximal order is ’5’.

Max Internal Steps This parameter is a positive integer value specifying the maximal number of internal steps the integrator is allowed to take before the next desired reporting time. The default value is ’10000’.

7.1.2 Stochastic Simulation

7.1.2.1 The Next-Reaction-Method

This stochastic simulation method utilizes the algorithm developed by Gibson and Bruck [Gibson00]. For each reaction a putative stochastic reaction time is calculated and the reaction with the shortest reaction time will be realized. The set of reactions is organized in a priority queue to allow for the efficient search for the fastest reaction. In addition, by using a so-called dependency graph only those reaction times are recalculated in each step, that are dependent on the reaction, which has been realized. This simulation method requires all the reactions to be irreversible. However, COPASI provides a tool, that converts all reversible reactions into irreversible ones. Because the algorithm internally works on discrete particle numbers rather than concentrations, the particle numbers in the system must not exceed a value of approximately 264.

The current implementation of the Next Reaction Method is very inefficient when the model contains assignment rules, which leads to extended calculation times.

There is a restriction regarding global quantities: If a differential equation is provided for a global quantity (the quantity is of type "ode") the model cannot be simulated stochastically with the current version of COPASI. If the global quantity is of type "assignment" stochastic simulation is possible but not as efficient as for models without assignments. No restrictions apply for "fixed" global quantities.

Options for the Next-Reaction-Method

Max Internal Steps This parameter is a positive integer value specifying the maximal number of internal steps the integrator is allowed to take before the next desired reporting time. The default value is ’1000000’.

Subtype This parameter is ignored in the current version of COPASI.

Use Random Seed This flag can be ’0’ or ’1’ and determines if the user-defined random seed should be used for the calculation. The default is ’0’ meaning that the random seed is set to a random value before each run and consecutively calculated trajectories will be different. If the value of this flag is set to ’1’, the user-defined random seed will be used and each calculated trajectory will be the same for the same value of the given random seed.

Random Seed This unsigned integer is used as random seed in the calculations, if the flag Use Random Seed is set to ’1’. The default value is ’1’.

7.1.3 Hybrid Simulation

7.1.3.1 Hybrid (Runge-Kutta)

This hybrid simulation method developed by us combines a deterministic numerical integration of ODEs with a stochastic simulation algorithm. The whole biochemical network is partitioned into a deterministic and a stochastic subnet internally. The deterministic subnet contains all reactions, in which only species with high particle numbers take part. All reactions with at least one low-numbered species are in the stochastic subnet, because here stochastic effects are expected. Which particle numbers are considered low or high can be specified by the user with the Lower Limit and the Upper Limit parameters (Species with particle numbers between those limits do not change their status. This leads to a hysteresis-like behavior and avoids many unnecessary swaps, if the particle numbers fluctuate in the middle range). The partitioning of the biochemical network can change dynamically

during the simulation. After a certain number of steps, which the user can define using the parameter Partitioning Interval, the partitioning is recalculated using the current particle numbers in the system. During one run the deterministic subnet and the stochastic subnet are simulated in parallel. A 4th-order Runge-Kutta method is used to numerically integrate the deterministic part of the system. For the stochastic part the simulation method by Gibson and Bruck [Gibson00] is utilized. The reaction probabilities of the stochastic subnet are approximated as constant during one stochastic step, even though in theory they can change due to the effects of the deterministic subnet.

The current implementation of the Hybrid Runge Kutta Method is very inefficient when the model contains assignment rules, which leads to extended calculation times.

Options for Hybrid (Runge-Kutta)

Max Internal Steps This parameter is a positive integer value specifying the maximal number of internal steps the integrator is allowed to take before the next desired reporting time. The default value is '1000000'.

Lower Limit This parameter is a double value specifying the lower limit for particle numbers. Species with a particle number below this value are considered as having a low particle number. The lower limit cannot be higher than the upper limit. The default value is '800'.

Upper Limit This parameter is a double value specifying the upper limit for particle numbers. Species with a particle number above this value are considered as having a high particle number. The upper limit cannot be lower than the lower limit. The default value is '1000'.

Runge Kutta Stepsize This positive double value is the step size of the Runge-Kutta solver for the integration of the deterministic part of the system. The default value is '0.001'.

Partitioning Interval This positive integer value specifies after how many steps the internal partitioning of the system should be recalculated. The default is '1', i.e. after every step the partitioning of the system is checked.

Use Random Seed This flag can be '0' or '1' and determines if the user-defined random seed should be used for the calculation. The default is '0' meaning that the random seed is set to a random value before each run and consecutively calculated trajectories will be different. If the value of this flag is set to '1', the user-defined random seed will be used and each calculated trajectory will be the same for the same value of the given random seed.

Random Seed This unsigned integer is used as random seed in the calculations, if the flag Use Random Seed is set to '1'. The default value is '1'.

7.2 Steady State Calculation

The steady state is the state in which the state variables of the model, e.g. the species concentrations do not change in time. Mathematically this is expressed by setting the differential equations that describe the time evolution of the metabolic system to zero. This forms a system of algebraic non-linear equations. To solve them, COPASI can use a series of strategies using more than one numerical method.

All calculations are done based on particle numbers and particle number rates rather than concentrations internally. The reduced model (see Deterministic Interpretation of the Model) is used. The Jacobian (which is used in the Newton method and when eigenvalues of the Jacobian are requested) is calculated using finite differences. The Eigenvalues of the Jacobian are calculated using LAPACK.

7.2.0.2 Options for Steady State Analysis

Use Newton This parameter is a Boolean value to determine whether to use the damped Newton method on the non-linear algebraic equations defining the steady-state. The initial concentrations set by the user are taken as guesses for the solution. A value of '1' (the default) indicates that COPASI shall use the damped Newton method.

The damped Newton method is a variant of the famous Newton method for the solution of systems of non-linear equations. The solution is obtained from an iterative procedure that refines an initial guess until the residual error is smaller than required. If a limit number of iterations is reached without an acceptable solution, the method halts without a solution.

The iteration of the plain Newton method is:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$$

In the damped method if x_{i-1} has a larger residual error than x_i one looks at:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})} \cdot 2^{-n} \text{ where } n = 0, \ldots, 32$$

and accepts the first such value that has a smaller residual error than x_i . If none is found, the procedure halts without a solution (because it is at a local minimum).

Use Integration This parameter is a Boolean value to determine whether to use the deterministic ODE solver to follow the time course defined by the differential equations until a steady state is reached. If at 10^{10} units of time no steady state has been reached the method halts with no solution. If Use Newton is '1' an attempt to find the Steady-statevia the damped Newton method is made at each intermediate time point. A value of '1' (the default) indicates that COPASI shall use integration.

Use Back Integration This parameter is a Boolean value to determine whether to use the deterministic ODE solver to reverse the time course (going backwards in time) defined by the differential equations until a steady state is reached. If at 10^{10} units of time no Steady-State has been reached the method halts with no solution. If Use Newton is '1' an attempt to find the steady-state via the damped Newton method is made at each intermediate time point. A value of '1' indicates that COPASI shall use back integration.

Accept Negative Concentrations This parameter is a boolean value to determine whether to accept a steady-state, which contains negative concentrations. A value of '1' indicates that negative concentrations are acceptable whereas a value of '0' (the default) indicates that such states are discarded.

Iteration Limit This parameter is a positive integer to determine the maximum number of iterations the damped Newton method shall perform before it fails. The default is '50'.

Derivation Factor This is a numeric value to determine the step size used to calculate $f'(x_{i-1})$. The default is '0.001'.

Resolution This is a positive numeric value to determine the resolution used to decide whether the current state is acceptable as a steady-state. If the absolute change of each state variable is smaller than the resolutions the state is accepted. The default is 10^{-9} .

Note, this value is interpreted as a concentration value, even though the calculation internally uses particle numbers. The reason for that is purely heuristic: In many cases the modeler will choose the units in a way that concentration values are neither extremely large nor extremely small numerically so that the default value for this parameter leads to useful results. However generally it is not save to just keep the default value without checking.

7.3 Metabolic Control Analysis

Metabolic control analysis (MCA) is a sensitivity analysis of metabolic systems. In MCA one studies the relative control exerted by each step on the system's variables (e.g. fluxes and species concentrations). This control is measured by applying a perturbation to the step being studied and then measuring the effect on the variable of interest after the system has settled to a new steady state.

7.3.1 Options for MCA

Modulation Factor This parameter is ignored in the current version of COPASI.

The rest of the options is described in the sections for Steady-State calculation and deterministic simulation.

7.3.2 Control Coefficients

A control coefficient is a relative measure of how much a perturbation on a parameter affects a system variable (e.g. fluxes or concentrations). It is defined [**?**] as:

$$C_{v_i}^{A} = \frac{\partial A}{\partial v_i}\frac{v_i}{A}$$

where A is the variable, i the step (enzyme) and v the steady-state rate of the perturbed step. The most common control coefficients are those for fluxes and species concentrations, but any variable of the system can be analyzed with MCA and have control coefficients defined by equations analogous to equation 1. In fact, there is no need even for the system to be in a steady state. COPASI only calculates directly the steady-state concentration- and flux-control coefficients, those for other variables can still be estimated by simulating small perturbations.

7.3.3 Summation Theorem

A very important property of steady-state metabolic systems was uncovered with the MCA formalism. This concerns the summation of all the flux control coefficients of a pathway. By various procedures [**?**] it can be demonstrated that for a given reference flux the sum of all flux-control coefficients (of all steps) is equal to unity:

$$\sum_i C_{v_i}^{J} = 1$$

For a given reference species concentration the sum of all concentration-control coefficients is zero:

$$\sum_i C_{v_i}^{[M]} = 0$$

where the summations are over all the steps of the system.

According to the first summation theorem, increases in some of the flux-control coefficients imply decreases in the others so that the total remains unity. As a consequence of the summation theorems, one concludes that the control coefficients are global properties and that in metabolic systems, control is a systemic property, dependent on all of the system's elements (steps).

7.3.4 Enzyme Kinetics and the Elasticity Coefficients

In enzyme kinetics the behavior of isolated enzymes is studied through the dependence of the initial rates of reaction with the concentration of the substrate(s). Enzyme kinetic studies are centered on derivation of rate equations and the determination of their kinetic constants such as Michaelis constants or limiting-rates or even on the elementary rate constants of a specific reaction mechanism.

In metabolic control analysis the properties of each (isolated) enzyme are measured in a way very similar to the flux-control properties: using a sensitivity, known as the elasticity coefficient [**?**]. In this case, one has to consider the effect of perturbations of a reaction parameter on the local reaction rate. By local one means that this sensitivity refers to the isolated reaction which has the same characteristics (effector and enzyme concentrations, temperature, and so on) as in the whole system at the operating point (steady state) of interest. The elasticity coefficients are defined as the ratio of relative change in local rate to the relative change in one parameter (normally the concentration of an effector). Infinitesimally, this is written as:

$$\varepsilon_{p}^{v_i} = \frac{\partial v_i}{\partial p}\frac{p}{v_1}$$

where v is the rate of the enzyme in question and p is the parameter of the perturbation. Each enzyme has as many elasticity coefficients as the number of parameters that affect it. One can immediately recognize the concentration of the reaction substrates, products and modifiers as parameters of the reaction. Unlike control coefficients, elasticity coefficients are not systemic properties but rather measure how isolated enzymes are sensitive to changes in their parameters. The elasticity coefficients can be obtained from the kinetic functions by partial derivation. Again like the control coefficients, the elasticity coefficients are not constants, they are dependent on the value of the relevant parameter and so are different for each Steady-State.

7.3.5 Connectivity Relations

A particularly useful and important feature of MCA is that it can relate the kinetic properties of the individual reactions (local properties) with (global) properties of the whole intact pathway. This is done through the connectivity theorems [Kacser73] that relate the control coefficients and the elasticity coefficients of steps with common intermediate species.

The connectivity theorem for flux-control coefficients [Kacser73] states that, for a common species S, the sum of the products of the flux-control coefficient of all (i) steps affected by S and its elasticity coefficients towards S, is zero:

$$\sum_i C^J_{v_i} \varepsilon^{v_i}_{[S]} = 0$$

For the concentration-control coefficients, the following two equations apply [Westerhoff84]:

$$\sum_i C^{[A]}_{v_i} \varepsilon^{v_i}_{[S]} = 0 \text{ , where } A \neq S$$

$$\sum_i C^{[A]}_{v_i} \varepsilon^{v_i}_{[S]} = -1$$

The first equation applies to the case in which the reference species A is different from the perturbed species S. Whereas the second applies to the case in which the reference species is the same as the perturbed species.

The connectivity theorems allow MCA to describe how perturbations on species of a pathway propagate through the chain of enzymes. The local (kinetic) properties of each enzyme effectively propagate the perturbation to and from its immediate neighbors.

7.3.6 Scaling

COPASI calculates (non-normalized) elasticity coefficients by numerical derivation with finite differences. To calculate control coefficients from steady-state data, COPASI applies the method described in [Reder88]. This method works with the reduced system where some variables are eliminated using conservation relations (see Deterministic Interpretation of the Model). All coefficients are obtained unscaled by this method and are scaled with the appropriate steady state concentrations and fluxes (the same with the elasticities). Both scaled and unscaled coefficients and elasticities are displayed and available for output.

7.4 Optimization Methods

The optimization methods described in this chapter attempt to minimize a given objective function. There are several ways to do this and COPASI supports many different methods for the minimization of an objective function.

7.4.1 Evolutionary Programming

Evolutionary programming (EP) [**?**] is a computational technique that mimics evolution and is based on reproduction and selection. An EP algorithm is composed of individuals that reproduce and compete, each one is a potential solution to the (optimization) problem and is represented by a "genome" where each gene corresponds to one adjustable parameter. At each generation of the EP, each individual reproduces asexually, i.e. divides into two individuals. One of these contains exactly the same "genome" as the parent while the other suffers some mutations (the parameter values of each gene change slightly). At the end of the generation, the algorithm has double the number of individuals. Then each of the individuals is confronted with a number of others to count how many does it outperform (the number of wins is the number of these competitors that represent worse solutions than itself). All the individuals are ranked by their number of wins, and the population is again reduced to the original number of individuals by eliminating those which have worse fitness (solutions).

7.4.1.1 Options for Evolutionary Programming

Number of Generations The parameter is a positive integer value to determine the number of generations the algorithm shall evolve the population. The default is '200'.

Population Size The parameter is a positive integer value to determine the size of the population, i.e., the number of individuals that survive after each generation. The default is '20'. Random Number Generator The parameter is an enumeration value to determine which random number generator this method shall use. COPASI provides two random number generators R250 [Maier91] (selected through the value 0) and the Mersenne Twister [Matsumoto98] (selected through the value 1 (default)).

Seed The parameter is a positive integer value to determine the seed for the random number generator. A value of zero instructs COPASI to select a "random" value.

7.4.2 Evolutionary Strategy (SRES)

Evolutionary Strategies with Stochastic Ranking (SRES) [Runarsson00] is similar to . However, a parent has multiple offsprings during each generation. Each offspring will contain a recombination of genes with another parent and additional mutations. The algorithm assures that each parameter value will be within its boundaries. But constraints to the solutions may be violated. Whenever this happens the square of the size of the violation is summed up, i.e., we calculate

$$\varphi = \sum_{c_j<l_{c_j}} (l_{c_j} - c_j)^2 + \sum_{c_j>u_{c_j}} (c_j - u_{c_j})^2$$

where the constraints are given by $c_j \in (l_{c_i}, u_{c_i})$. The value φ is used within the selection, which is performed as described in Genetic Algorithm SR.

7.4.2.1 Options for Evolutionary Strategy (SRES)

Number of Generations The parameter is a positive integer value to determine the number of generations the algorithm shall evolve the population. The default is '200.'

Population Size The parameter is a positive integer value to determine the size of the population, i.e., the number of individuals that survive after each generation. The default is '20'.

Random Number Generator The parameter is an enumeration value to determine which random number generator this method shall use. COPASI provides two random number generators R250 [Maier91] (selected through the value 0) and the Mersenne Twister [Matsumoto98] (selected through the value 1 (default)).

Seed The parameter is a positive integer value to determine the seed for the random number generator. A value of zero instructs COPASI to select a "random" value.

Pf This parameter is a numerical value in the interval (0, 1) determining the chance that individuals either outside the parameter boundaries or violating the constraints are compared during the selection. The default is '0.475'.

7.4.3 Genetic Algorithm

The genetic algorithm (GA) [**?**] is a computational technique that mimics evolution and is based on reproduction and selection. A GA is composed of individuals that reproduce and compete, each one is a potential solution to the (optimization) problem and is represented by a "genome" where each gene corresponds to one adjustable parameter. At each generation of the GA, each individual is paired with one other at random for reproduction. Two offspring are produced by combining their genomes and allowing for "cross-over", i.e., the two new individuals have genomes that are formed from a combination of the genomes of their parents. Also each new gene might have mutated, i.e. the parameter value might have changed slightly. At the end of

the generation, the algorithm has double the number of individuals. Then each of the individuals is confronted with a number of others to count how many does it outperform (the number of wins is the number of these competitors that represent worse solutions than itself). All the individuals are ranked by their number of wins, and the population is again reduced to the original number of individuals by eliminating those which have worse fitness (solutions).

Many features of a GA may be varied. The details of this particular implementation of the GA for optimization of biochemical kinetics are:

- Parameters are encoded in genes using floating-point representation, rather than the more usual binary representation.
- Mutation is carried out by adding to the gene a random number drawn from a normal distribution with zero mean and a standard deviation of 10% of the parameter value. Whenever this makes the parameter (gene) exceed one boundary, it is set to that boundary value.
- Cross-over is always performed at gene boundaries so that no gene is ever disrupted. The number of cross-over points is a random number between zero and half the number of adjustable parameters (uniform distribution).
- Selection is done by a tournament where each individual competes with a number of others equal to 20% the population size. The competitors are chosen at random.
- The initial population contains one individual whose genes are the initial parameter values, the genes of all other individuals are initialized to a random value between their boundaries. If the boundaries span two orders of magnitude or more, the random distribution is exponential, otherwise normal.
- Whenever the fittest individual has not changed for the last 10 generations, the 10% less fit individuals are replaced by individuals with random genes. When the fittest individual has not changed for 30 generations, the worse 30% are substituted by individuals with random genes. When the fittest individual has not changed for 50 generations, the worse 50% are substituted by individuals with random genes. This procedure helps the algorithm escape local minima and is somewhat equivalent to increasing the mutation rate when the population has become uniform.

7.4.3.1 Options for Genetic Algorithm

Number of Generations The parameter is a positive integer value to determine the number of generations the algorithm shall evolve the population. The default is ’200’.

Population Size The parameter is a positive integer value to determine the size of the population, i.e., the number of individuals that survive after each generation. The default is ’20’.

Random Number Generator The parameter is an enumeration value to determine which random number generator this method shall use. COPASI provides two random number generators R250 [Maier91] (selected through the value 0) and the Mersenne Twister [Matsumoto98] (selected through the value 1 (default)).

Seed The parameter is a positive integer value to determine the seed for the random number generator. A value of zero instructs COPASI to select a "random" value.

7.4.4 Genetic Algorithm SR

The genetic algorithm with stochastic ranking is very similar to the before described with tournament selection. With two exception which are the mutations are not forced to be within the boundaries and the selection is done through a bubble sort with a random factor as described in [Runarsson00].

- Parameters are encoded in genes using floating-point representation, rather than the more usual binary representation.
- Mutation is carried out by adding to the gene a random number drawn from a normal distribution with zero mean and a standard deviation of 10% of the parameter value. Parameters may exceed boundaries. Whenever this happens or a constraint to the solution is violated the square of the size of the violation is summed up, i.e., we calculate

$$\varphi = \sum_{p_i<l_{p_i}}(l_{p_i}-p_i)^2+\sum_{p_i>u_{p_i}}(p_i-u_{p_i})^2+\sum_{c_j<l_{c_j}}(l_{c_j}-c_j)^2+\sum_{c_j>u_{c_j}}(c_j-u_{c_j})^2$$

where the parameters are given by $p_i \in (l_{p_i}, u_{p_i})$ and the constraints by $c_i \in (l_{c_i}, u_{c_i})$ The value φ is used within the selection.

- Cross-over is always performed at gene boundaries so that no gene is ever disrupted. The number of cross-over points is a random number between zero and half the number of adjustable parameters (uniform distribution).
- Selection is done by the bubble sort described in [Runarsson00]. This sort incorporates a probability to compare objective values for individuals with a $\varphi \neq 0$. The pseudo code for the sort is:

```
// Here sweepNum is optimal number of sweeps from paper, i.e., TotalPopulation
for (i = 0; i < sweepNum; i++)
  {
    wasSwapped = false;

    for (j = 0; j < TotalPopulation - 1; j++)
      {
        // within bounds or random chance
        if ((phi(j) == 0 and phi(j + 1) == 0) or UniformRandom(0, 1) < Pf)
          {
            // compare objective function values
            if (Value(j) > Value(j + 1))
              {
                swap(j, j + 1);
                wasSwapped = true;
              }
          }
        else // phi != 0
          {
            // individual j further outside then j + 1
            if (phi(j) > phi(j + 1))
              {
                swap(j, j + 1);
                wasSwapped = true;
              }
          }
      }

    // if no swap then break
    if (wasSwapped == false) break;
  }
```

- The initial population contains one individual whose genes are the initial parameter values, the genes of all other individuals are initialized to a random value between their boundaries. If the boundaries span two orders of magnitude or more, the random distribution is exponential, otherwise normal.
- Whenever the fittest individual has not changed for the last 10 generations, the 10% less fit individuals are replaced by individuals with random genes. When the fittest individual has not changed for 30 generations, the worse 30% are substituted by individuals with random genes. When the fittest individual has not changed for 50 generations, the worse 50% are substituted by individuals with random genes. This procedure helps the algorithm escape local minima and is somewhat equivalent to increasing the mutation rate when the population has become uniform.

7.4.4.1 Options for Genetic Algorithm SR

Number of Generations The parameter is a positive integer value to determine the number of generations the algorithm shall evolve the population. The default is '200'.

Population Size The parameter is a positive integer value to determine the size of the population, i.e., the number of individuals that survive after each generation. The default is '20'.

Random Number Generator The parameter is an enumeration value to determine which random number generator this method shall use. COPASI provides two random number generators R250 [Maier91] (selected through the value 0) and the Mersenne Twister [Matsumoto98] (selected through the value 1 (default)).

Seed The parameter is a positive integer value to determine the seed for the random number generator. A value of zero instructs COPASI to select a "random" value.

Pf This parameter is a numerical value in the interval (0, 1) determining the chance that individuals either outside the parameter boundaries or violating the constraints are compared during the selection. The default is '0.475'.

7.4.5 Hooke & Jeeves

The method of Hooke and Jeeves [**?**] is a direct search algorithm that searches for the minimum of a nonlinear function without requiring (or attempting to calculate) derivatives of the function. Instead it is based on a heuristic that suggests a descent direction using the values of the function calculated in a number of previous iterations.

7.4.5.1 Options for Hooke & Jeeves

Iteration Limit This parameter is positive integer determining the maximum number of iterations the algorithm shall perform. The default is '50'.

Tolerance This parameter is a positive value determining the tolerance with which the solution shall be determined. If the improvement between two steps is less than the tolerance the algorithm stops. The default is '10^{-5} '.

Rho This parameter is a value in (0, 1) determining the factor with which the steps size is reduced between iterations. The default is '0.2'.

7.4.6 Levenberg - Marquardt

Levenberg-Marquardt [Levenberg44], [Marquardt63] is a gradient descent method. It is a hybrid between the steepest descent and the Newton methods.

The Newton optimization method searches for the minimum of a nonlinear function by following descent directions determined from the function's first and second partial derivatives. The steepest descent method searches for a minimum based only on the first derivatives of the function. While the Newton method converges quadratically towards the minimum in its vicinity, it may not converge at all if it is far away from it. On the other hand the steepest descent method only converges linearly but is guaranteed to converge.

Levenberg first suggested an improvement to the Newton method in order to make it more robust, i.e. to overcome the problem of non-convergence. His suggestion was to add a factor to the diagonal elements of the Hessian matrix of second derivatives when not close to the minimum (this can be judged by how positive definite the matrix is). The effect when this factor is large compared to the elements of Hessian is that the method then becomes the steepest descent method. Later Marquardt suggested that the factor should be multiplicative rather than additive and also defined a heuristic to make this factor increase or decrease. The method known as Levenberg-Marquardt is thus an adaptive method that effectively changes between the steepest descent to the Newton method.

The original suggestions of Levenberg and Marquardt were effective to enhance the Gauss-Newton method, a variant of the Newton method specifically for minimizing least-squares functions. In this case the advantage is also that the second derivatives do not need to be calculated as they are estimated from the gradient of the residuals. Subsequently Goldfeld et al. [Goldfeld66] extended the method to the case of general non-linear functions.

7.4.6.1 Options for Levenberg - Marquardt

Iteration Limit This parameter is positive integer determining the maximum number of iterations the algorithm shall perform. The default is '200'.

Tolerance This parameter is a positive value determining the tolerance with which the solution shall be determined. If the improvement between two steps is less than the tolerance the algorithm stops. The default is '10^{-5} '.

7.4.7 Nelder - Mead

This method also known as the simplex method is due to Nelder and Mead [Nelder65]. A simplex is a polytope of *N+1* vertices in *N* dimensions. The objective function is evaluated at each vertex. Dependent on these calculated values a new simplex is constructed. The simplest step is to replace the worst point with a point reflected through the centroid of the remaining *N* points. If this point is better than the best current point, then we can try stretching exponentially out along this line. On the other hand, if this new point isn't much better than the previous value then we are stepping across a valley, so we shrink the simplex towards the best point.

7.4.7.1 Options for Nelder - Mead

Iteration Limit This parameter is a positive integer to determine the maximum number of iterations the method is to perform. The default value is '200'.

Tolerance This parameter is a positive number and provides an alternative termination criteria. If the variance of the values of the objective function at the vertices of the current simplex is smaller than the tolerance the algorithm stops. The default is '10^{-5} '.

Scale This parameter is a positive number and determines the size of the initial simplex. The edges of the polytope are inverse proportional to the scale, i.e., a larger value makes the initial simplex smaller. The default is '10'.

7.4.8 Particle Swarm

The particle swarm optimization method suggested by Kennedy and Eberhart [Kennedy95] is inspired by a flock of birds or a school of fish searching for food. Each particle has a position Xi and a velocity Vi in the parameter space. Additionally, it remembers its best achieved objective value O and position Mi. Dependent on its own information and the position of its best neighbor (a random subset of particles of the swarm) a new velocity is calculated. With this information the position is updated. The pseudo code for the algorithm for each particle is:

```
N =  best neighbor's position

for i = 0 to number of parameters do
  R1 = uniform random number in [0, 1]
  R2 = uniform random number in [0, 1]

  V[i]= w * V[i]
        + C * R1 * (M[i]- X[i])
        + C * R2 * (N[i]- X[i])
  X[i] = X[i] + V[i]
enddo

current_O = evaluate objective function

if current_O < O then do
  O = current_O
  M = X
enddo
```

7.4.8.1 Options for Particle Swarm

Iteration Limit This parameter is a positive integer to determine the maximum number of iterations the method is to perform. The default value is '2000'.

Swarm Size This parameter is a positive integer specifying the number of particles in the swarm. The default value is '50'.

Std. Deviation This parameter is a positive number and provides an alternative termination criteria. If the standard deviation of the values of the objective function of each particle and the standard deviation of the best positions is smaller than the provided value the algorithm stops. The default value is 10^{-6} .

Random Number Generator The parameter is an enumeration value to determine which random number generator this method shall use. COPASI provides two random number generators R250 [Maier91] (selected through the value 0) and the Mersenne Twister [Matsumoto98] (selected through the value 1 (default)).

Seed The parameter is a positive integer value to determine the seed for the random number generator. A value of zero instructs COPASI to select a "random" value.

7.4.9 Praxis

Praxis is a direct search method (for a review see [Swann72]) that searches for the minimum of a nonlinear function without requiring (or attempting to calculate) derivatives of that function. Praxis was developed by Brent [Brent72] after the method proposed by Powell [Powel64]. The inspiration for Praxis was the well-known method of minimising each adjustable parameter (direction) at a time - the principal axes method. In Praxis directions are chosen that do not coincide with the principal axes, in fact if the objective function is quadratic then these will be conjugate directions, assuring a fast convergence rate.

This implementation of the Praxis method was originally written in FORTRAN at the Stanford Linear Accelerator Center (dated 3/1/73). The original FORTRAN code is available from http://www.netlib.org/opt/praxis. The original code was translated automatically by the F2C program from AT&T Bell Labs (available at http://www.netlib.org/f2c/). The C code was then enhanced to suite COPASI's reentrant optimization method interface.

7.4.9.1 Options for Praxis

Tolerance Convergence stopping criterium: the method stops when two consecutive estimates differ by less than Tolerance. This parameter is a positive integer to determine the number of parameter sets to be drawn before the algorithm stops. The default value is 10^{-5} .

7.4.10 Random Search

Random search is an optimization method that attempts to find the optimum by testing the objective function's value on a series of combinations of random values of the adjustable parameters. The random values are generated complying with any boundaries selected by the user, furthermore, any combinations of parameter values that do not fulfill constraints on the variables are excluded. This means that the method is capable of handling bounds on the adjustable parameters and fulfilling constraints.

For infinite number of iterations this method is guaranteed to find the global optimum of the objective function. In general one is interested in processing a very large number of iterations.

7.4.10.1 Options for Random Search

Number of Iterations This parameter is a positive integer to determine the number of parameter sets to be drawn before the algorithm stops. The default value is '100000'.

Random Number Generator The parameter is an enumeration value to determine which random number generator this method shall use. COPASI provides two random number generators R250 [Maier91] (selected through the value 0) and the Mersenne Twister [Matsumoto98] (selected through the value 1 (default)).

Seed The parameter is a positive integer value to determine the seed for the random number generator. A value of zero instructs COPASI to select a "random" value.

7.4.11 Simulated Annealing

Simulated annealing is an optimization algorithm first proposed by Kirkpatrick et al. [Kirkpatrick83] and was inspired by statistical mechanics and the way in which perfect crystals are formed. Perfect crystals are formed by first melting the substance of interest, and then cooling it very slowly. At large temperatures the particles vibrate with wide amplitude and this allows a search for global optimum. As the temperature decreases so do the vibrations until the system settles to the global optimum (the perfect crystal).

The simulated annealing optimization algorithm uses a similar concept: the objective function is considered a measure of the energy of the system and this is maintained constant for a certain number of iterations (a temperature cycle). In each iteration, the parameters are changed to a nearby location in parameter space and the new objective function value calculated; if it decreased, then the new state is accepted, if it increased then the new state is accepted with a probability that follows a Boltzmann distribution (higher temperature means higher probability of accepting the new state). After a fixed number of iterations, the stopping criterion is checked; if it is not time to stop, then the system's temperature is reduced and the algorithm continues.

Simulated annealing is a stochastic algorithm that is guaranteed to converge if ran for an infinite number of iterations. It is one of the most robust global optimization algorithms, although it is also one of the slowest. (Be warned that simulated annealing can run for hours or even days!).

This implementation of simulated annealing is based on the code of Corana et al. [Corana87]. This implementation has tuned some of the parameters of the original implementation as follows:

- Each temperature cycle takes $10 \cdot p \cdot \max(5 * p, 100)$ random steps, with p being the number of optimization parameters.
- At each step, a new candidate solution is accepted if it fulfills either of the following:
  1. it reduced the objective function value, or
  2. with a probability equal to $e^{-\frac{\Delta f}{T}}$, where Δf is the increase in objective function value, and T is the current temperature.
- The stopping criterion is applied to the last two temperature cycles (i.e. the change in objective function between this temperature and the previous must have been smaller than the Tolerance in the last two cycles).

7.4.11.1 Options for Simulated Annealing

Start Temperature Initial temperature of the system. The higher the temperature, the larger the probability that a global optimum is found. Note that the temperature should be very high in the beginning of the method (the system should be above the "melting" temperature). This value has the same units as the objective function, so what represents "high" is different from problem to problem. The default is '1'.

Cooling Factor Rate by which the temperature is reduced from one cycle to the next, given by the formula: Tnew=Told*"Cooling Factor". The simulated annealing algorithm works best if the temperature is reduced at a slow rate, so this value should be close to 1. (but values closer to 1 will also cause the algorithm to run longer). The default is '0.85'.

Tolerance Convergence stopping criteria: the method stops when the change in the objective function has been smaller than this value in the last two temperature steps. The default is 10^{-6} . (This is an absolute tolerance)

Random Number Generator The parameter is an enumeration value to determine which random number generator this method shall use. COPASI provides two random number generators R250 [Maier91] (selected through the value 0) and the Mersenne Twister [Matsumoto98] (selected through the value 1 (default)).

Seed The parameter is a positive integer value to determine the seed for the random number generator. A value of zero instructs COPASI to select a "random" value.

7.4.12 Steepest Descent

Steepest descent [Fogel92] is an optimization method that follows the direction of steepest descent on the hyper-surface of the objective function to find a local minimum. The direction of steepest descent is defined by the negative of the gradient of the objective function.

7.4.12.1 Options for Steepest Descent

Iteration Limit This parameter is positive integer determining the maximum number of iterations the algorithm shall perform. The default is '100'.

Tolerance This parameter is a positive value determining the tolerance with which the solution shall be determined. If the improvement between two steps is less than the tolerance the algorithm stops. The default is 10^{-6} .

7.4.13 Truncated Newton

The Truncated Newton method is a sophisticated variant of the Newton optimization method. The Newton optimization method searches for the minimum of a nonlinear function by following descent directions determined from the function's first and second partial derivatives. The Truncated Newton method does an incomplete (truncated) solution of a system of linear equations to calculate the Newton direction. This means that the actual direction chosen for the descent is between the steepest descent direction and the true Newton direction. A more complete description of the method can be found in [Gill81] and [Nash84].

The particular implementation of the Truncated Newton method used here takes into account simple bounds on the optimization parameters. Like the Newton method itself, the Truncated Newton variant converges quadratically when in the vicinity of the minimum. On average this method is one of the best gradient descent methods.

This implementation of the Truncated Newton method was originally written in FORTRAN by Stephen Nash (Operations Research and Applied Statistics Dept., George Mason University, Fairfax, VA 22030, USA). The original code is available from http://www.netlib.org/opt/tn). The original code was translated automatically to C by the F2C program from AT&T Bell Labs (available at http://www.netlib.org/f2c/). The C code was then enhanced to suite COPASI's reentrant optimization method interface.

7.4.13.1 Options for Truncated Newton

None The algorithm is not tunable.

7.5 Lyapunov Exponents Calculation

COPASI allows the calculation of Lyapunov exponents of a trajectory as well as the average divergence of the system. The exponents are calculated for the reduced system (see), so the maximum number of exponents that can be calculated is the number of independent variables. If less than this number of exponents is requested, the largest exponents are calculated.

COPASI uses the well known algorithm proposed by Wolf et al. [Wolf85]. This algorithm integrates one reference trajectory and simultaneously N difference trajectories (if N is the number of exponents requested) in a system linearized around the reference trajectory. This integration is carried out for a short time interval (the "Orthonormalization interval", see below) and then the difference vectors are reorthonormalized. The exponents for this time interval are calculated from how much each of the difference trajectories converged or diverged from the reference trajectory during the interval. This calculation is repeated and the "local" exponents are averaged over the the whole trajectory.

The divergence is calculated (if requested) as the average of the trace of the Jacobian. Although it is not numerically necessary the divergence is also calculated for the same short intervals that are used for the Lyapunov exponents. This allows comparing the local values of the divergence with the local exponents.

If you are only interested in the end result of the Lyapunov exponents and the average divergence you can just use the default report that is provided by COPASI (or just look at the result in the GUI). If you want to have access to the "local" results for the single orthonormalization intervals however, you will have to define a plot or report manually. In this version of COPASI you will need to used the "expert" feature of the object selection dialog to access the exponents. They are located in the "Lyapunov Exponents" branch under the "Task List" entry. Output takes place after each reorthonormalization interval. The output can contain each of the ten largest exponents, both the local value from the last interval and the and the average value of all intervals calculated so far. Correspondingly the divergence can be output both as an average over the last interval or as an average over the whole trajectory.

The Jacobian that is used for both Lyapunov exponents and divergence calculation is calculated using finite differences. The integration of the reference and difference trajectories is done using LSODA [Hindmarsh83].

7.5.1 Options for Lyapunov exponents calculation

Orthonormalization interval This is the time interval after which an orthonormalization of the difference trajectories takes place. This parameter is critical for the accuracy of the Lyapunov exponents. Smaller values generally lead to more accurate results, but take longer time (since the numerical integration needs to be restarted for many short calculations). One way to judge the adequacy of this parameter is to compare the sum of exponents with the divergence of the system. Those two values should be the same (only if you request the calculation of all exponents), and since the calculation of the divergence is very robust, a mismatch would typically mean that the orthonormalization interval needs to be smaller. Note that this parameter mostly affects the accuracy of the exponents with the largest absolute values. Since large positive exponents are unusual, this means that the largest negative exponents suffer from accuracy problems related to this parameter. If you don't need the exact values of the strongly negative exponents you can chose a larger value for this parameter and enjoy a much faster calculation. The default value is 1.0.

Overall time This parameter specifies the overall time of the calculation. The integration will be repeated in small steps given by the "Orthonormalization interval" parameter until the overall time is reached. This value is also critical for the accuracy of the exponents. Since COPASI cannot guess how fast the exponents converge, no save default value can be given for this parameter. One indication would be that if the system does not run into a steady state one of the exponents should be zero. If this is not the case in the result, probably the overall time was to short to allow the exponents to converge to their average value. The default value is 1000.

The rest of the options apply for the LSODA numerical integrator that is used for the calculation. They are described in the section for deterministic simulation.

7.6 Sensitivities Calculation

COPASI allows the calculation of generalized sensitivities of a model. This is done by numerical differentiation using finite differences. The user specifies a list of functions to be differentiated and one or two lists of variables. The differentiation is performed with respect to these variables. Since every function in the list is differentiated with respect to every variable in the list the result will generally be a two-dimensional matrix. If also a list of variables for second derivatives is given then all the first derivatives are again differentiated with respect to all the variables in the second list. The result will be a three-dimensional array of second derivatives.

7.6.1 Options for sensitivities calculation

Delta factor This is used to determine the delta value for the finite difference numerical differentiation. The delta is calculated as the product of the delta factor and the current absolute value of the variable. If the resulting value is smaller than the "Delta minimum" parameter, it is discarded and the "Delta minimum" value is used instead. Default value is 10^{-6}

Delta minimum The minimal delta for numerical differentiation. Default value is 10^{-12} .

7.7 Time Scale Separation Methods

In this chapter we describe the time scale separation methods for reducing of biochemical systems. Both methods rely on the presence of a wide range of characteristic time scales in biological systems and are based on the local analysis of the Jacobian, which is partitioned into fast and slow components at the initial point of user chosen interval .

7.7.1 Common parameters of the time scale separation methods

Intervals The user specifies the number of intervals to which the method is applied. COPASI performs the analysis at each initial point of these intervals. Please note that the interval size should be large in comparison with the time scale interesting for the user.

Deuflhard Tolerance This parameter is a positive numeric value specifying the maximal tolerated error of slow modes (see Deuflhard and Heroth, 1996). The default value is $1 \cdot 10^{-6}$. Please note that reasonable values of Deuflhard tolerance should be of the same order as the time scale the user is interested in.

7.7.2 ILDM (Deuflhard)

The ILDM (Deuflhard) method utilizes the algorithm developed by Deuflhard and Heroth (see [Deuflhard96], [Zobeley05], [Surovtsova09] for details).

7.7.2.1 Basic concept of decomposition into "slow" and "fast" modes

The block slow — fast decomposition of the Jacobian is performed in two steps:

- First a real Schur decomposition yields a block upper triangular matrix.
- In a second step the desired decoupled structure of the transformed Jacobian is obtained by solving a Sylvester equation.

This basis procedure results in a transformation of state vectors in new modes, which are then separated in slow and fast modes. Accordingly, the system dynamics of a full reaction system comprising n ODEs is reduced to a DAE system consisting of n_{slow} ordinary differential equations and $n - n_{\text{slow}}$ algebraic equations. The number n_{slow} of slow variables is calculated iteratively using the tolerance criterion of Deuflhard and Heroth. The method implemented in COPASI has an additional focus on the reduction of the underlying biochemical network and not only on the reduction of the mathematical equations. For this purpose COPASI performs the analysis of transformation matrices derived after solving the Sylvester equation.

7.7.2.2 The implementation in COPASI

The Jacobian is calculated by using finite differences. The Schur transformation and the solution of Sylvester equation are performed using CLAPACK. Results in COPASI are displayed in four matrices and four vectors. The matrix "Species" provides the contribution of each metabolite to every mode, whereas the table "Modes" summarizes the mode distribution for each metabolite. Vector "Slow space" ("Fast space" respectively) displays the contribution of each concentration variable to the set of all slow (fast) modes. The matrices "Reactions contributions to modes" and "Reactions distribution between modes" are the product of stoichiometric matrix and transformation matrix normed by column and row respectively. Two vectors "Reactions slow space" and "Reactions fast space" provide the contribution of each reaction to the slow and fast spaces. The metabolites with largest contribution to the fast space could be supposed to be "fast" and thus, its ordinary differential equation is replaced by the corresponding algebraic equation (i.e. with the same right-hand side). In the specific case that a subset of species does not contribute to the slow space (but contributes only to the fast space), the time-scale decomposition results in a dissection of the reaction network. The reactions dominating in the fast modes are fast reactions.

7.7.3 Modified ILDM

The algorithm of modified ILDM method has been developed by our group. The main goal is to distinguish between "slow" and "fast" metabolites — instead of "modes". So the final result of the analysis can be formulated without linear transformation of the reaction system.

7.7.3.1 The implementation in COPASI

For proposing a selection of fast metabolites, COPASI uses essentially the same numerical tools of local Schur decomposition of Jacobian as for the ILDM (Deuflhard) method.

The vector "Fast space" specifies the metabolites making the largest contributions to the set of fast modes. So it provides the next candidate of concentration which is then tested whether its additional nomination as fast metabolite still satisfies the criterion of error tolerance.The result will be two matrices and two vectors, which describe the contribution of metabolites to slow (fast) modes (spaces). However, the corresponding analysis is performed by using Schur transformation matrices. The corresponding numbers of slow and fast components now refer to the "real" slow and fast metabolites (not to modes).

Chapter 8

Error Messages

This section contains a list of COPASI error messages and an explanation when and/or why this error might have occurred. This is work in progress and not all error messages might have been fully documented here yet.

Also keep in mind that the vast majority of error messages listed here are internal to COPASI and users will probably never see them at all. If you see one of those, you know that there is a bug in COPASI and you should provide a bug report that is as detailed as possible. At the very least, the bug report should include a description of what you did when the bug occurred, the operating system you are running COPASI on and the version of COPASI you were using. Naturally, if possible, we would also like to have some file that shows the error reproducibly since this makes finding bugs a lot easier for us. Error messages that you should never encounter will be marked INTERNAL in the description. So if you encounter any error message marked as INTERNAL please provide as with a bug report so we can fix it. There are two ways to provide a bug report. The first one is to go to COPASI's bugzilla web-page and create a new account if you don't already have one and enter a full bug report into the bugzilla database. The other way to report a bug is to send an email to bugs@copasi.org with the information mentioned above.

8.1 Memory Allocation

This error message comes from allocation memory problem.

Message

Identifier CVector

Message Memory allocation failed for 'NBYTES' bytes.

Description INTERNAL.

8.2 Gepasi File Reader

The CReadConfig messages report problems when reading a Gepasi file.

Message

Identifier CReadConfig (1)

Message Variable 'VARIABLENAME' not found in 'FILENAME(LINENUMBER)'.

Description COPASI expects the variable VARIABLENAME in the file FILENAME at line LINENUMBER. Often LINENUMBER will be the last line of the file as COPASI searches for the variable.

Message

Identifier CReadConfig (2)

Message Cannot open file 'FILENAME'.

Description COPASI cannot open the file FILENAME.

Message

Identifier CReadConfig (3)

Message Cannot read file 'FILENAME'.

Description COPASI cannot read the file FILENAME.

Message

Identifier CReadConfig (4)

Message Cannot close file 'FILENAME'.

Description COPASI cannot close the file FILENAME.

Message

Identifier CReadConfig (5)

Message Invalid type 'TYPE' for Variable 'VARIABLENAME'.

Description COPASI expects the variable VARIABLENAME to be of a different type than TYPE.

8.3 Kinetic Function

These error messages are related to problems of kinetic function.

Message

Identifier CKinFunction (1)

Message Cannot find identifier 'IDENTIFIERNAME'.

Description COPASI cannot find the identifier IDENTIFIERNAME.

Message

Identifier CKinFunction (2)

Message Missing operant in function 'FUNCTIONNAME'.

Description COPASI expects operant in function FUNCTIONNAME.

8.4 Range

Error message about invalid range.

Message

Identifier CRange (1)

Message Invalid range ('STARTRANGE', 'ENDRANGE').

Description The range (STARTRANGE, ENDRANGE) is invalid.

8.5 COPASI Vector

These error messages are from COPASI's vector class which is sued in many places. If you see those error messages, who very likeley found a bug in COPASI and you should provide us with a bug report.

Message

Identifier CCopasiVector (1)

Message Object 'OBJECTNAME' not found.

Description INTERNAL

Message

Identifier CCopasiVector (2)

Message Object 'OBJECTNAME' already exists.

Description INTERNAL

Message

Identifier CCopasiVector (3)

Message Index 'INDEX' out of range (0, 'LASTINDEX').

Description INTERNAL

8.6 Function Parameter

Error messages from COPASI's internal function representation.

Message

Identifier CFunctionParameters (1)

Message The usage 'TYPE' is not unique for a vector type parameter ('VECTORTYPE').

Description INTERNAL

Message

Identifier CFunctionParameters (2)

Message No parameter with usage 'TYPE' with index >= 'INDEX'.

Description INTERNAL

8.7 Mass Action

COPASI uses a separate class for representing mass action kinetics in reaction. These error messages are from that class.

Message

Identifier CMassAction (1)

Message The function Mass Action reversibility must be either TRUE or FALSE.

Description INTERNAL

8.8 COPASI Method

Classes derived from CCopasiMethod provide the individual methods for the different task. If you see one of those error messages, you have probably found a bug in COPASI and we would like you to send us a bug report.

Message

Identifier CCopasiMethod (1)

Message No parameter list found for name 'PARAMETERNAME' and type 'PARAMETERTYPE'.

Description INTERNAL

Message

Identifier CCopasiMethod (2)

Message Problem is not set.

Description INTERNAL

Message

Identifier CCopasiMethod (3)

Message Model is not set in problem

Description INTERNAL

8.9 Reaction

CReaction is COPASIs internal reaction representation. Many of the error messages in this class are related to SBML import/export.

Message

Identifier CReaction (1)

Message Function 'FUNCTIONNAME' not found.

Description INTERNAL. A reaction tries to use a function as a kinetic law that is unknown.

Message

Identifier CReaction (2)

Message In Reaction 'REACTIONNAME' the compartment could not be guessed.

Description !!!UNDOCUMENTED!!!

Message

Identifier CReaction (3)

Message Reaction 'REACTIONNAME' has no substrates and no products.

Description COPASI found a reaction with neither substrates nor products. A reaction needs to have at least one substrate or one product.

Message

Identifier CReaction (4)

Message Only Species, Compartments or Parameter object nodes are allowed in kinetic functions.

Description In kinetic functions, only references to compartments, species and/or parameters are allowed. For explicit time dependent models, time can also be referenced from a kinetic function. If this error message occurs, COPASI has found a reference that is neither a reference to a compartment, a species, a parameter or the time.

Message

Identifier CReaction (5)

Message Nodes of type 'TYPENAME' are not implemented yet.

Description INTERNAL

Message

Identifier CReaction (6)

Message Nodes of type VARIABLE must not appear in an expression.

Description INTERNAL

Message

Identifier CReaction (7)

Message Species object 'OBJECTNAME' is neither substrate, product nor modifier to reaction 'REACTIONNAME' but it is used in the kinetic law.

Description During SBML import COPASI found a name in a kinetic law that is neither a parameter, a substrate, product or modifier to the reaction. The most likely cause for this that a modifier has not been declared in the listOfModifiers of the reaction.

Message

Identifier CReaction (8)

Message Could not find variable with name 'VARIABLENAME'.

Description INTERNAL

Message

Identifier CReaction (9)

Message Could not find object for key 'KEY'.

Description INTERNAL

Message

Identifier CReaction (10)

Message Parameter 'PARAMETER' is a vector.

Description INTERNAL

Message

Identifier CReaction (11)

Message Reaction 'REACTIONNAME' refers to unusable Function 'FUNCTIONNAME'.

Description !!!UNDOCUMENTED!!!

8.10 Chemical Equation

These error messages come from chemical equation problem.

Message

Identifier CChemEq (1)

Message No Substrates and no Products.

Description !!!UNDOCUMENTED!!!

Message

Identifier CChemEq (2)

Message Substrates in different Compartments.

Description !!!UNDOCUMENTED!!!

Message

Identifier CChemEq (3)

Message No Substrates and Products are in different compartments.

Description !!!UNDOCUMENTED!!!

8.11 Method Parameter

Errors on defining method parameters are reported here.

Message

Identifier CCopasiParameter (1)

Message Invalid value 'VALUE' for 'PARAMETERNAME'

Description !!!UNDOCUMENTED!!!

Message

Identifier CCopasiParameter (2)

Message Elevation failed, since no parameter is provided.

Description INTERNAL

Message

Identifier CCopasiParameter (3)

Message Elevation failed, since the parameter is not of the required source type.

Description INTERNAL

Message

Identifier CCopasiParameter (4)

Message Elevation failed, since elevation of children failed.

Description INTERNAL

Message

Identifier CCopasiParameter (5)

Message Elevation failed, since the parameter is not member of this group.

Description INTERNAL

8.12 Trajectory Method

These error messages can occur when running a trajectory task. All of them depend on the method that has been chosen for the trajectory, e.g. stochastic, since the error message is generated by the method and not the task itself.

Message

Identifier CTrajectoryMethod (1)

Message Hybrid simulation not applicable, since the stoichiometry contains a non-integer.

Description Hybrid simulations can not be run on models that contain reactions with non integer stoichiometries.

Message

Identifier CTrajectoryMethod (2)

Message Hybrid simulation not applicable, since reversible reactions exists.

Description Hybrid simulations can not be run on models with reversible reactions. You can try to convert all reactions to irreversible reactions via the "Convert to irreversible" menu item in the tools menu.

Message

Identifier CTrajectoryMethod (3)

Message Hybrid simulation not applicable, since more than one compartment is involved.

Description Hybrid simulations only work on single compartment models.

Message

Identifier CTrajectoryMethod (4)

Message Lower Limit 'LOWERLIMIT' is greater than Upper Limit 'UPPER LIMIT'.

Description Check the limits for the hybrid method. The LOWERLIMIT parameter has to be smaller than the UPPERLIMIT parameter.

Message

Identifier CTrajectoryMethod (6)

Message Deterministic integration failed. LSODA reported:LSODA_ERROR_MESSAGE Please see result for indications of numerical instability.

Description The deterministic integration failed. This is a strong indication of numerical problems. A look at the time series calculated up to the point the problem occurred is often helpful in resolving the problem.

Message

Identifier CTrajectoryMethod (7)

Message Problem is not set.

Description INTERNAL

Message

Identifier CTrajectoryMethod (8)

Message Problem is not a trajectory problem.

Description INTERNAL

Message

Identifier CTrajectoryMethod (9)

Message Negative time steps not possible with stochastic simulation.

Description The duration of the trajectorytime course is negative. This is not allowed for stochastic integration.

Message

Identifier CTrajectoryMethod (10)

Message The tau-Leap Method encountered numerical problems. You can try to reduce the tau-value.

Description Choose a lower value for tau and rerun the time course task.

Message

Identifier CTrajectoryMethod (11)

Message Invalid tau-value ('TAU'). Tau must have a positive value.

Description Values for tau must be positive numbers.

Message

Identifier CTrajectoryMethod (12)

Message Internal step limit exceeded.

Description The internal step limit is exceeded. Reducing the accuracy requirements or increasing the step limit may resolve the problem.

Message

Identifier CTrajectoryMethod (13)

Message Runge Kutta Step size must be positive in hybrid method.

Description The stepsize for the Runge Kutta solver within the hybrid method must be positive, since the hybrid method does not support backward integration in time.

Message

Identifier CTrajectoryMethod (14)

Message Use Random Seed should be 0 or 1 since it is a boolean parameter.

Description The value of the parameter called "Use Random Seed" should be 0 or 1 since it is a boolean parameter.

Message

Identifier CTrajectoryMethod (15)

Message Max Internal Steps needs to be positive.

Description The maximal number of internal steps must be at least 1.

Message

Identifier CTrajectoryMethod (16)

Message Numerical Error encountered.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTrajectoryMethod (17)

Message At least one reaction is necessary to perform stochastic simulation.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTrajectoryMethod (18)

Message The model contains a global quantity with an ODE rule. Stochastic simulation is not possible.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTrajectoryMethod (19)

Message The model contains a global quantity with an assignment rule. The value of the quantity is used in the model. Stochastic simulation of such models is not possible with this version of COPASI.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTrajectoryMethod (20)

Message The model contains a species with an ODE rule. Stochastic simulation is not possible.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTrajectoryMethod (21)

Message The model contains a compartment with an ODE rule. Stochastic simulation is not possible.

Description !!!UNDOCUMENTED!!!

8.13 XML Reader

These error messages can occur when reading COPASI files. Many of the error messages of the XML file reader are not serious. Since the COPASI file format has changed during development, current versions of COPASI might not recognize some elements used in older versions of the file format. And vice versa if you try to read a COPASI file generated with a version newer than the one you are currently using, the file might contain new elements unknown to the older version of COPASI. In the worst case some task settings might be lost.

Message

Identifier XML (1)

Message Required attribute 'ATTRIBUTENAME' not found (line: 'LINENUMBER').

Description The required attribute ATTRIBUTENAME is missing at line LINENUMBER. Please check the XML file.

Message

Identifier XML (2): XML error (line: 'LINENUMBER', column: 'POSITION')

Message 'MESSAGE'.

Description Incorrect XML encountered at line LINENUMBER and character POSITION. The message gives further details about the error. This error is also shown when libsbml finds an error in an SBML file.

Message

Identifier XML (3)

Message Unknown element 'ELEMENTNAME' encountered at line 'LINENUMBER'.

Description The XML contained an unknown element ELEMENTNAME at line LINENUMBER. This element is ignored.

Message

Identifier XML (4)

Message Unknown parameter 'PARAMETERNAME' encountered at line 'LINENUMBER'.

Description The XML contained an unknown parameter PRAMETERNAME at line LINENUMBER. This parameter is ignored.

Message

Identifier XML (5)

Message Unknown task type 'TASKTYPE' encountered at line 'LINENUMBER'.

Description The XML contained an unknown task of type TASKTYPE at line LINENUMBER. This task is ignored.

Message

Identifier XML (6)

Message Invalid function 'FUNCTIONNAME' encountered at line 'LINENUMBER'.

Description The XML contains an invalid function FUNCTIONNAME at line LINENUMBER. This error should never occur, if it occurs with a file generated by COPASI, please send a bug report.

Message

Identifier XML (7)

Message Unknown function 'FUNCTIONNAME' in reaction 'REACTIONNAME' encountered at line 'LINENUMBER'.

Description The reaction REACTIONAME uses a call to an unknown function named FUNCTIONNAME in line LINENUMBER. Please check the XML file.

Message

Identifier XML (8)

Message Unknown variable 'VARIABLENAME' in function 'FUNCTIONNAME' encountered at line 'LINENUMBER'. A possible reason is that the variable is a reserved string within the function description.

Description Please check the file if a variable of the function FUNCTIONNAME at line LINENUMBER is a reserved keyword. For a list of reserved keywords see the section on

---

Message

Identifier XML (9)

Message The file 'FILENAME' is written in a newer version 'COPASIVERSION' of the COPASI file format. This file might include features your version of COPASI does not support. To assure full compatibility please download the newest version at http://www.copasi.org.

Description Self explanatory.

Message

Identifier XML (10)

Message Invalid element '<ELEMENTNAME>' expecting '<ELEMENTNAME>' encountered at line 'LINENUMBER'.

Description While reading a file, COPASI has encountered an element where it actually expected a different element.

Message

Identifier XML (11)

Message Invalid closing element '<ELEMENTNAME>' expecting '<ELEMENTNAME>' encountered at line 'LINENUMBER'.

Description While reading an XML file, COPASI encountered the end element tag for an element, but actually expected another element to end.

Message

Identifier XML (12)

Message Order 'INDEX' out of range for variable 'VARIABLENAME' in function 'FUNCTIONNAME' encountered at line 'LINENUMBER'.

Description !!!UNDOCUMENTED!!!

Message

Identifier XML (13)

Message Unrecognized format in file 'FILENAME'.

Description !!!UNDOCUMENTED!!!

Message

Identifier XML (14)

Message Local reaction parameters may no longer be used in any expression in the model. The following automatic corrections have been applied: AUTOMATIC_CORRECTIONS Please note: Tasks, reports, and plots may have been affected by these changes and may no longer work as expected.

Description !!!UNDOCUMENTED!!!

8.14 Message Handling

Error message related to message handling.

Message

Identifier Message (1)

Message No more messages.

Description Self explanatory

8.15 Configuration File

These error messages are related to reading and writing COPASIs configuration file. (see)

Message

Identifier Configuration (1)

Message COPASI directory is not set. Some features might not be working correctly. Please set the environment variable COPASIDIR or use the command-line options -c COPASIDIR or –copasidir COPASIDIR to point to the COPASI installation directory.

Description Self explanatory

Message

Identifier Configuration (2)

Message Configuration file 'FILENAME' found but is not readable.

Description COPASI found the configuration file but could not read it. Please check if you have sufficient permissions on the file FILENAME.

8.16 Optimization Task

These are error messages that can occur when running an optimization task.

Message

Identifier Optimization (1)

Message Object 'OBJECTNAME' not found.

Description The object specified by OBJECTNAME can not be found. This normally indicates that the model was changed and the parameters or constraints must be adjusted.

Message

Identifier Optimization (2)

Message Lower Bound 'OBJECTNAME' not found.

Description The lower bound specified by OBJECTNAME can not be found. This normally indicates that the model was changed and the parameters or constraints must be adjusted.

Message

Identifier Optimization (3)

Message Upper Bound 'OBJECTNAME' not found.

Description The upper bound specified by OBJECTNAME can not be found. This normally indicates that the model was changed and the parameters or constraints must be adjusted.

Message

Identifier Optimization (4)

Message Empty Interval ('LOWERBOUND', 'UPPERBOUND') specified.

Description The interval is invalid as the lower bound is larger than the upper bound. Please correct the problem.

Message

Identifier Optimization (5)

Message Invalid Objective Functions.

Description The objective function is invalid, i.e, COPASI can not interpret it. Please correct the problem.

Message

Identifier Optimization (6)

Message No adjustable Parameters specified.

Description Since no parameter are specified no optimization can be performed.

Message

Identifier Optimization (7)

Message No Task Type specified.

Description The optimization task type is not specified. Supported types are currently time course and steady state.

Message

Identifier Optimization (8)

Message 'NUMFAILS' Function Evaluation out of 'NUMEVALS' failed.

Description A high percentage of failed function evaluation indicates that the parameter range is not appropriate or the objective function is ill defined. For parameter estimation problems please check also whether the experimental data is correctly specified in accordance with provided data files.

8.17 SBML

The following errors are generated by the SBML importer and exporter. If you encounter errors related to SBML import, please check your SBML file with the SBML On-line Validator at http://sbml.org/tools/htdocs/sbmltools.php before you file a bug report.

Message

Identifier SBML (1)

Message SBML Level 2 Version 1 does not support initial times different from 0. This information will be lost in the exported file.

Description The initial time of your model is set to a value different than 0. This information can not be stored in SBML Level2 Version1 and will therefore not be present in the exported model.

Message

Identifier SBML (2)

Message The SBML document contains no model.

Description This error occurs if you try to load an SBML file that does not contain a model.

Message

Identifier SBML (3)

Message The SBML document contains algebraic rules that were ignored. Entities determined by those rules are probably undetermined now.

Description COPASI does not support SBML algebraic rules yet, so if you load an SBML file that contains algebraic rules, COPASI will warn you that the rules have been ignored. Please be aware that this will very likely change simulation results.

Message

Identifier SBML (4)

Message The SBML document contains events that were ignored.

Description COPASI does not support SBML events yet, so if you load an SBML file that contains events, COPASI will warn you that the events have been ignored. Please be aware that this will very likely change simulation results.

Message

Identifier SBML (5)

Message Compartment COMPARTMENTID referenced by species SPECIESID does not exist.

Description In the SBML file a the species with id SPECIESID references the compartment with id COMPARTMENTID, but a compartment with this id does not exist.

Message

Identifier SBML (6)

Message Annotations in SBase element of SBML Level 1 files, this is invalid and has been ignored.

Description Some programs add annotations to the SBase element of the file which is incorrect. Since it doesn't affect the mode, COPASI reads the model anyway but ignores the annotation. If you want to keep the annotation please open the SBML file in a text editor and move it to e.g. the model element.

Message

Identifier SBML (7)

Message No initial value or initial assignment set for compartment "COMPARTMENTID".

Description In order to simulate a model in COPASI it has to be complete, meaning all initial values (or, initial assignment) and parameter values have to be set. SBML allows the user to specify incomplete models. This warning tells the user that some values in the model have not been set and that the model is incomplete. Most tasks in COPASI will not work with this model until all values have been set. Unset values will be shown as NaN in COPASI graphical user interface. The fastest way to find unset values is to use the . This warning can also be the result of an ignored rule that would normally set the value of this model entity.

Message

Identifier SBML (8)

Message Expression tree for kinetic law of reaction 'REACTIONID' could not be converted.

Description The kinetic formula for the reaction with id REACTIONID could not be converted into COPASI's internal function representation. If you think the formula is correct, please send as a bug report with the SBML file included so we can have a look at it.

Message

Identifier SBML (9)

Message Could not set function from expression in reaction 'REACTIONID'.

Description Somehow COPASI was able to generate in internal representation for a function formula, but could not use it with a reaction. If you get such an error, please send as a bug report with the SBML file so we can look at it.

Message

Identifier SBML (11)

Message Function 'FUNCTIONID' does not start with a Lambda element.

Description Function definitions in MathML have to start with a Lambda element. If you see this error, the function with the id FUNCTIONID is very likely incorrect MathML because it is missing this element.

Message

Identifier SBML (12)

Message Function 'FUNCTIONID' contains invalid parameter list.

Description The error message means that the parameter list for the function definition with id FUNCTIONID contains an invalid type, e.g. a number instead of a parameter name.

Message

Identifier SBML (13)

Message Could not read function definition for function 'FUNCTIONID'.

Description COPASI could not read the function definition with id FUNCTIONID. If you think this function definition is correct, please send us a bug report and include the SBML file.

Message

Identifier SBML (14)

Message Could not import function with id 'ID'.

Description This error message is very similar to the one above but the error here already occurred in libsbml while reading the function definition.

Message

Identifier SBML (15)

Message Could not find function with id 'ID'.

Description COPASI could not find the function with id ID while exporting a model to SBML. If you get this error message please save the model as a COPASI file and send us a bug report with this file.

Message

Identifier SBML (16)

Message Loop found in function. That means a function calls itself either directly or indirectly.

Description COPASI has found a loop in a function call while exporting your model to SBML. This should never happen so if you see this error message please save the model as a COPASI file and send us a bug report with this file.

Message

Identifier SBML (17)

Message One or more single compartment reactions found where the kinetic law does not include a multiplication by the volume: REACTIONID.

Description This is a warning only. Please see the COPASI FAQ question 'Why are some rate laws in SBML files exported by COPASI multiplied by the compartment volume?' at http://www.copasi.org/. If COPASI finds a species reference in a reaction that has the hasOnlySubstanceUnits flag set, the warning is not generated for that reaction.

Message

Identifier SBML (18)

Message Support for 'hasOnlySubstanceUnit' flag on species is experimental. Species in compartments of dimension 0 are also considered to have this flag set.

Description Dealing with species in SBML files that have the "hasOnlySubstanceUnits" flag set has changed in recent versions of COPASI. We are pretty sure that the way it is handled now is correct. If you think that you found a case where COPASI does it incorrectly, please let us know.

Message

Identifier SBML (19)

Message Spatial size units on species "SPECIESID" is not the same as the volume unit in the model and has been ignored.

Description COPASI does not support the spatialSizeUnits attribute for species. The species with id SPECIESID in the current file seems to use this attribute however.

Message

Identifier SBML (20)

Message Setting an initial concentration on species SPECIESID which has the 'hasOnlySubstanceUnits' flag set is not allowed.

Description The SBML file seems to contain a species with id SPECIESID that has the hasOnlySubstanceUnits flag set to true but also has an initialConcentration set. This is incorrect, please specify an initial amount instead.

Message

Identifier SBML (21)

Message Setting an initial concentration on species 'SPECIESID' which is in a compartment with spatial dimensions 0 is not allowed.

Description Having a compartment with a spatial dimension of 0 implies that all species within this compartment have the hasOnlySubstanceUnits flag set to true. Therefore it is incorrect to specify an initial concentration for those species. Specify an initial amount instead. Current version of COPASI refuse to read models with compartments that have their spatial size units set to 0 so this error message will never appear.

Message

Identifier SBML (22)

Message Current versions of COPASI only supports three dimensional compartments. 'COMPARTMENTID' will be converted to be three dimensions.

Description COPASI only supports three dimensional compartments so far, so all compartments are converted to three dimensions no matter what is specified in the SBML file. Independent of this is the handling of spatial dimensions that are set to 0 in the SBML file (see error SBML (21) above). Current version of COPASI refuse to read models with compartments that have their spatial size units set to 0 so this error message will never appear.

Message

Identifier SBML (23)

Message Compartment 'COMPARTMENTID' has spatial dimensions of 0, setting dimensions to 3. Considering all species in that compartment to have "hasOnlySubstanceUnits" flag set.

Description If the compartment in an SBML file has a spatial dimension of 0, all species in the compartment are considered to have the "hasOnlySpeciesUnits" flag set. This means that if such a species is referenced within a mathematical expression, the amount of the species is referenced and not the concentration. Since COPASI only knows threedimensional compartments, the dimensionality of the compartment is ignored, this can lead to incorrect display of units within COPASI, but it should not affect numerical values of the kodel of of calculations done on the model Since COPASI only knows threedimensional compartments, the dimensionality of the compartment is ignored, this can lead to incorrect display of units within COPASI, but it should not affect numerical values of the kodel of of calculations done on the model.

Message

Identifier SBML (24)

Message Units for some compartments were ignored. Units might be displayed incorrectly. Compartments: COMPARTMENTSID.

Description If COPASI finds a compartment that has units different from the default volume units of the model, the units are ignored. This might lead to the display of incorrect units in the GUI.

Message

Identifier SBML (25)

Message Units for some species were ignored. Units might be displayed incorrectly. Species: SPECIESID.

Description If COPASI finds a species that has units different from the default substance units of the model, the units are ignored. This might lead to the display of incorrect units in the GUI.

Message

Identifier SBML (26)

Message Units for some parameters were ignored. Units might be displayed incorrectly. Parameters: PARAMETERSID.

Description If COPASI finds a parameter that has units set, the units are ignored. This might lead to the display of incorrect units in the GUI.

Message

Identifier SBML (27)

Message Error in kinetic law for reaction 'REACTIONID'.

Description Some error occurred while creating the kinetic law for the reaction with id REACTIONID. If you think the kinetic law in the SBML file is OK, please send a bug report and include the SBML file so that we can have a look at it.

Message

Identifier SBML (28)

Message Error in function definition with id 'ID'.

Description Somehow COPASI was not able to convert the function definition with id ID into it's internal format. If you think the function definition is correct, please send a bug report and include the SBML file.

Message

Identifier SBML (29)

Message Unable to handle reactions with the 'fast' flag set. The flag has been set to false.

Description COPASI is not yet able to handle reactions that are set to fast correctly the flag will be automatically set to false on import. In some cases simulation results for these models can be incorrect.

Message

Identifier SBML (30)

Message Can't handle units of type item with scale set to 1. If this file was created with COPASI RC1 or older please see the COPASI FAQ.

Description COPASI prior to and including RC1 had a bug in the SBML exporter. If the default concentration units were set to items, COPASI would set the scale to 1 instead of 0 on the exported substance unit definition. You can correct those files by opening them in a text editor and setting the scale for the substance units to 0. If the file was not created with COPASI, the scale was probably meant to be 1 and COPASI can not import it since it does not do unit conversion yet.

Message

Identifier SBML (31)

Message Assigment/ODE-Rules are currently only supported for global parameters.

Description (OBSOLETE) COPASI currently only supports assignment and ODE rules for global parameters, if an SBML file contains such rules for species or compartments, they are ignored. Working with such a model in COPASI will most likely lead to incorrect results.

Message

Identifier SBML (32)

Message Error in Assignment/ODE-Rule's variable id 'ID' does not specify a compartment, species or global parameter.

Description Rules in SBML files are only allowed on compartments, species and global parameters if the id referenced from a rule does not correspond to one of those entity types, it is an error.

Message

Identifier SBML (33)

Message Error: Assignment/ODE-Rule is not allowed for local parameter 'PARAMETERID'.

Description COPASI encountered a rule for a local parameter which is not allowed.

Message

Identifier SBML (34)

Message Error: Assignment/ODE-Rule is not allowed for local constant 'CONSTANTNAME' identified by id 'ID'

Description Ĩ!!!UNDOCUMENTED!!!

Message

Identifier SBML (35)

Message Error: Only one AssignmentRule or RateRule is allowed for id 'ID'.

Description COPASI has encountered an entity for which more than one rule has been specified.

Message

Identifier SBML (36)

Message COPASI does not support time delays. Calculations on this model will most likely lead to unusable results.

Description COPASI currently does has no possibility to calculate time delays as they are supported in SBML models. If COPASI encounters a model with such a time delay, the delay is converted to a normal function call that evaluates to NaN (Not a number). This might lead to calculation results containing a large amount of NaN values instead of numerical values. If you want to work with the model, you have to remove all calls to the delay function from the mathematical expressions in the model.

Message

Identifier SBML (37)

Message The id 'ID' is used in the expression of a rule, although it is later defined by a rule itself.

Description An object is referenced in the expression of a rule but is later defined by a rule itself. This is not allowed, please change the order of rules in the SBML file so that the rule for object with id is listed before the rule ID is referenced in. Also make sure that this does not lead to circular dependencies.

Message

Identifier SBML (38)

Message Only references to compartment volumes, species concentrations, global parameter values or the time are allowed in SBML rule expressions.

Description Objects referenced from the expression of a rule in SBML can only be compartments, species or global parameters.

Message

Identifier SBML (39)

Message Object with id "ID" referenced in kinetic law, but no object with that id found in model.

Description An id is referenced in the expression of a kinetic law, but COPASI could not find an object with that id in the model.

Message

Identifier SBML (40)

Message LIBSBML ERRORTYPE ERRORNUMBER at line LINENUMBER column COLUMNNUMBER: ERRORMESSAGE.

Description This error message actually encapsulates all errors, informations and warnings libSBML finds while checking a file it loads. The error message will tell you the severity of the message, the error number, where it occured in the file and the actual message from libSBML. Right now libSBML will probably give you many warnings and errors for your files which you can either ignore or depending on the error, use to fix the model file. If you think that an error message is actually incorrect, please let the authors of libSBML know so that they can have a look at it.

Message

Identifier SBML (41)

Message No initial value set for species "SPECIESID". Setting initial concentration to 1.0.

Description In order to simulate a model in COPASI it has to be complete, meaning all initial values and parameter values have to be set. SBML allows the user to specify incomplete models. This warning tells the user that some values in the model have not been set and that the model is incomplete. Most tasks in COPASI will not work with this model until all values have been set. Unset values will be shown as NaN in COPASI graphical user interface. It is save to set the initial concentration to 1.0.

Message

Identifier SBML (42)

Message No initial value set for local parameter "PARAMETERID".

Description In order to simulate a model in COPASI it has to be complete, meaning all initial values and parameter values have to be set. SBML allows the user to specify incomplete models. This warning tells the user that some values in the model have not been set and that the model is incomplete. Most tasks in COPASI will not work with this model until all values have been set. Unset values will be shown as NaN in COPASI graphical user interface. The fastest way to find unset values is to use the . This warning can also be the result of an ignored rule that would normally set the value of this model entity.

Message

Identifier SBML (43)

Message No initial value set for global parameter "PARAMETERID".

Description In order to simulate a model in COPASI it has to be complete, meaning all initial values and parameter values have to be set. SBML allows the user to specify incomplete models. This warning tells the user that some values in the model have not been set and that the model is incomplete. Most tasks in COPASI will not work with this model until all values have been set. Unset values will be shown as NaN in COPASI graphical user interface. The fastest way to find unset values is to use the . This warning can also be the result of an ignored rule that would normally set the value of this model entity.

Message

Identifier SBML (44)

Message Substance unit in kinetic law for some reactions were ignored. Units might be displayed incorrectly. Reactions: REACTIONSID.

Description If COPASI finds substance units in a kinetic law that differ from the default substance units of the model, those units are ignored. This might lead to the display of incorrect units in the GUI.

Message

Identifier SBML (45)

Message Compartment "COMPARTMENTID" does not set the initial volume. Volume has been set to 1.0.

Description COPASI has found a compartment in the model that does not specify an initial volume. Since COPASI needs this information, it would normally not be able to read those models. In certain cases when all species of the compartment in question have the hasOnylSubstanceUnits flag set, it is save to set the volume of that compartment to 1.0 in order to be able to read the model.

Message

Identifier SBML (46)

Message COPASI has changed the following function definitions to take the time as an additional argument instead of the function being directly or indirectly dependent on time: "FUNCTIONNAME" .

Description Up to SBML Level 2 Version 2 it was legal to use the time symbol within a function definition. Since COPASI can not ahndle this internally, those function definitions have to be converted to take an extra parameter which is the time. All appearances of the time symbol within the function definition are replaced by the name of the parameter. COPASI also replaces all function calls to those functions by new function calls that have the extra parameter which corresponds to time. Since starting with SBML Level 2 Version 3 it is illegal to have a time object in a function definition, this also helps to generate SBML files that are easily converted to future versions.

Message

Identifier SBML (47)

Message COPASI found a call to the function "FUNCTIONID" which has not been defined.

Description In some MathML term in the SBML file, COPASI found a term that is used as a function, but where it does not correspond to a predefined MathML function or the id of any function definition in the file.

Message

Identifier SBML (49)

Message Constraints ignored because they are not supported yet.

Description The current versions of COPASI do not support SBML constraints yet. We hope to change this in the future.

Message

Identifier SBML (50)

Message Could not open file "FILENAME".

Description Out of some reason, COPASI could not open the file. Reasons for this can be for example that the file does not exist, the user does not have sufficient permissions to open the file or that the file isn't an SBML file.

Message

Identifier SBML (51)

Message The species "SPECIESID" is defined by a rate rule and its compartments volume is variable. COPASI will probably interpret this incorrectly.

Description COPASIs interpretation of species that are defined by a rate rule and that are located in a non-constant compartment differes slightly from the interpretation intended by the SBML specification. This difference might lead to calculation results that are not consistent with the given SBML file. We hope to fix this in future version of COPASI.

Message

Identifier SBML (52)

Message The species "SPECIESID" is defined by a rate expression and its compartments volume is variable. The way COPASI interprets this is differently from the way SBML does.

Description COPASIs interpretation of species that are defined by a rate rule and that are located in a non-constant compartment differes slightly from the interpretation intended by the SBML specification. If such a model is exported to SBML, the resulting SBML model is not identical to the original COPASI model. We hope to fix this in future versions of COPASI.

Message

Identifier SBML (53)

Message The time units of kinetic laws in some reactions were ignored. Units might be displayed incorrectly. Reactions: REACTIONSID.

Description If COPASI finds a kinetic law in an SBL file that has time units different from the global time units, they are ignored. This might lead to incorrect units being displayed in the GUI.

Message

Identifier SBML (54)

Message Error while importing volume unit with id "ID".

Description COPASI could not convert the given volume unit to one of its own volume units. This might lead to incorrect units being displayed in the GUI.

Message

Identifier SBML (55)

Message Could not find unit definition for unit with id "ID" used in "ATTRIBUTE" attribute of UNIT with id "ID".

Description COPASI found a reference to a unit definition that has not been defined in the file.

Message

Identifier SBML (56)

Message There was a problem with the kinetic law in reaction "REACTIONSID". Make sure the math element is not empty.

Description Out of some reason COPASI could not import the kinetic law of some reaction. Most often this means that the kinetic law has some error. If you think that the kinetic law is correct, please send us a bug report.

Message

Identifier SBML (57)

Message InitialAssignment defined for object with id "ID", but the corresponding object could not be found. Ignoring assignment.

Description The SBML file contains an initial assignment for an object that is not defined in the SBML file.

Message

Identifier SBML (58)

Message InitialAssignment for object with id "ID" does not set a mathematical expression. Ignoring assignment.

Description The SBML file contains an initial assignment that does not define a mathematical expression for the assignment.

Message

Identifier SBML (59)

Message Error while importing InitialAssignment for object with id "ID". Ignoring assignment.

Description If this error message comes up, the initial assignment is incorrect, or you have found a bug in COPASI. If you think the initial assignment is actually correct, please send us a bug report.

Message

Identifier SBML (60)

Message Error while exporting EXPRESSIONTYPE for ELEMENTTYPE with name "ELEMENTNAME".

Description This error means that there were problems when exporting a rule or an initial assignment. If you see this message, it is very likely that you have found a bug in COPASI. Please send us a bug report that includes the COPASI file for reproducing this error.

Message

Identifier SBML (61)

Message Error while expanding function calls in mathematical expression for EXPRESSIONID.

Description This error means that there were problems when exporting a mathematical expression the SBML Level 1. If you see this message, it is very likely that you have found a bug in COPASI. Please send us a bug report that includes the COPASI file for reproducing this error.

Message

Identifier SBML (62)

Message Error while replacing unsupported elements in mathematical expression for EXPRESSIONID.

Description This error means that there were problems when exporting a mathematical expression the SBML Level 1. Either the expression that was exported contained elements that could not be represented in SBML Level 1, or you have found a bug in COPASI. Since exporting to SBML Level 1 is a new feature, it would be nice, if you could provide us with a bug report if you see this message.

Message

Identifier SBML (63)

Message Initial assignment for ASSIGNMENTID "ASSIGNMENT" can not be exported to SBML Level 2 Version 1.

Description Initial assignment have been introduced in SBML Level 2 Version 2, so models that contain initial assignments can not be exporter to SBML Level 2 Version 1.

Message

Identifier SBML (64)

Message One or more stoichiometric expressions were evaluated and converted to constants values.

Description COPASI does not support stoichiometric expression, so when an SBML file contains a stoichiometric expression, the expression is evaluated and converted to a constant. This will lead to incorrect calculation results if the expression contained a variable element.

Message

Identifier SBML (65)

Message The stoichiometric expression for a species reference for species "SPECIESID" in reaction "REACTIONID" could not be evaluated. The value has therefore been set to 1.0..

Description COPASI was not able to evaluate the stoichiometric expression for some species reference. This either means that the expression was incorrect, or that you found a bug in COPASI. If you think that the expression is correct, please sned us a bug report.

Message

Identifier SBML (66)

Message COPASI was not able to import the global UNITTYPE unit. Unit has been set to UNIT.

Description This error means that COPASI has encountered some problem while importing one of the global units for either substance, time or volume. Most likely this is due to the fact that the unit in the SBML file could not be converted to a unit COPASI understands. The corresponding unit has therefore been set to a default unit. As a consequence, the units displayed in the GUI might be incorrect. It has however no influence on any calculations with the model or the numerical values generated during those calculations.

8.18 Trajectory Task

These error messages are generated by the trajectory task so they might occur whenever you run a trajectory task.

Message

Identifier CTrajectoryProblem (1)

Message Invalid step size = 'VALUE'.

Description Normally this occurs when the step size is 0 which obviously is not meaningful.

Message

Identifier CTrajectoryProblem (2)

Message The step number 'VALUE' exceeds the limit. The step number and step size have been adjusted to avoid over flow.

Description The step number exceed the limit supported by COPASI. It has been adjusted to the maximal acceptable value.

Message

Identifier CTrajectoryProblem (3)

Message The step size 'VALUE' is to small for the machine accuracy. The step number and step size have been adjusted.

Description The step size is so small that the step number exceeds the limit supported by COPASI. It has been adjusted to the maximal acceptable value.

8.19 Directory Entry

Message

Identifier DirEntry (1)

Message Directory entry 'ENTRY' already exists.

Description A directory COPASI tried to create already exists.

Message

Identifier DirEntry (2)

Message Directory entry 'ENTRY' is read-only.

Description COPASI tried to write to a directory where it does not have writing permissions. Please check the permissions.

8.20 MathML

These error messages can occur when COPASI finds an error while importing the MathML from an SBML file.

Message

Identifier MathML (1)

Message Unsupported element 'ELEMENT'.

Description COPASI does not support all MathML elements allowed in SBML files yet. E.g. the delay function is currently not supported.

Message

Identifier MathML (2)

Message Unknown element in MathML expression.

Description During SBML import an unknown MathML element was encountered. Please check the MathML expressions in the SBML file.

Message

Identifier MathML (3)

Message MINUS operator can only have one or two arguments.

Description During SBML import, COPASI has encountered a MathML expression where a minus operator has 0 or more than 2 arguments. Please check the MathML in the SBML file.

Message

Identifier MathML (4)

Message DIVIDE and POWER operator can only have 2 arguments.

Description The divide and the power operator in MathML must have exactly two arguments. During SBML import COPASI has found a MathML expression where this is not the case. Please check the MathML in the SBML file.

8.21 Function

COPASI uses a lexer and a parser generated by Bison to parse function strings. These error messages can occur when an incorrect function is entered or read from file.

Message

Identifier Function (1)

Message Parser error after position: 'POSITION'.

Description The grammatical function parser encountered an error parsing the function description after character POSITION.

Message

Identifier Function (2)

Message Lexer error after position: 'POSITION'.

Description The lexical function parser encountered an error parsing the function description after character POSITION.

Message

Identifier Function (3)

Message Compile error after position: 'POSITION'.

Description A function compile failed after character POSITION.

Message

Identifier Function (4)

Message Circular dependency detected.

Description The function definition results in circular dependencies and thus are invalid. This may happen if the function call another function.

8.22 Evaluation Node

Message

Identifier CEvaluationNodeObject (1)

Message Only references to compartment, species, parameters and reaction are allowed in expression.

Description INTERNAL. During export to SBML COPASI has found a reference in a kinetic law that is neither a reference to the time, a species, a compartment, a parameter or a reaction.

8.23 COPASI Task

All tasks in COPASI are derived from one base class called CCopasiTask. These error messages are generated by this base class and should never occur. If you encounter such an error message it is very likely that you found a bug in COPASI. Please file a bug report.

Message

Identifier CCopasiTask (1)

Message No problem defined for task 'TASK'.

Description INTERNAL

Message

Identifier CCopasiTask (2)

Message No model associated for task 'TASK'.

Description INTERNAL

Message

Identifier CCopasiTask (3)

Message No method defined for task 'TASK'.

Description INTERNAL

Message

Identifier CCopasiTask (4)

Message Error compiling model 'MODELNAME'.

Description This indicates a serious problem with the model. Please check your model. If you think your model is OK, please send a bug report.

Message

Identifier CCopasiTask (5)

Message No output file defined for report.

Description This means that there is a report but that COPASI does not know where to write it to since this has not been specified. All data from this report will be lost.

Message

Identifier CCopasiTask (6)

Message Requested output object: 'OUTPUTOBJECTID' not found. It will be ignored.

Description At least one of the objects used in the report could not be found. The report will generated despite the problem but might not be what you expected. This usually happens if you change the model after you defined the plots.

Message

Identifier CCopasiTask (7)

Message Problems compiling output.

Description !!!UNDOCUMENTED!!!

8.24 Steady State Calculation

These error messages can occur when a steady state is calculated.

Message

Identifier CSteadyState (1)

Message The model is explicitly time dependent. Therefore, the calculation of a steady state is not very meaning full.

Description One or more of the kinetic laws used in the model are explicitly time dependent a steady state can therefore not be calculated by COPASI.

8.25 Parameter Fitting

These errors messages can occur when doing a parameter estimation.

Message

Identifier CFitting (1)

Message Failed to determine work area size for matrix inversion.

Description INTERNAL

Message

Identifier CFitting (2)

Message Failed to invert Fisher information matrix.

Description This reports that the Fisher information matrix could not be inverted due to singularity or numerical problems. The parameter statistics will be incomplete.

Message

Identifier CFitting (3)

Message No column with type 'Time' specified for a time course experiment.

Description For a time course exactly on column of the data file must be mapped to the model time.

Message

Identifier CFitting (4)

Message Insufficient experimental data (columns requested 'COLUMNNUMBER', columns found 'COLUMNNUMBER').

Description The data specification contains more columns than the file provides. Calculation is stopped and the problem must be fixed first.

Message

Identifier CFitting (5)

Message Incomplete data mapping, column 'COLUMNINDEX' must be mapped.

Description A column which is not ignored is not mapped to an object within COPASI. Calculation is stopped and the problem must be fixed first.

Message

Identifier CFitting (6)

Message Object 'OBJECTNAME' in column 'COLUMNNINDEX' has no numeric value.

Description The object must represent a numerical value. Currently this must be a floating point. This object can currently not be used in parameter fitting.

Message

Identifier CFitting (7)

Message Insufficient experimental data (rows requested 'ROWNUMBER', rows found 'ROWNUMBER').

Description The data specification describes more data rows than the experimental data provides. Calculation is stopped and the problem must be fixed first.

Message

Identifier CFitting (8)

Message Failure reading file 'FILE'.

Description The file cannot be read. Calculation is stopped and the problem must be fixed first. Make sure all data files specified for the fitting are there and readable.

Message

Identifier CFitting (9)

Message Experiment 'EXPERIMENTID' has no data rows.

Description !!!UNDOCUMENTED!!!

Message

Identifier CFitting (10)

Message Experiment 'EXPERIMENTID' has no dependent data.

Description !!!UNDOCUMENTED!!!

Message

Identifier CFitting (11)

Message Missing independent data for Experiment 'EXPERIMENTID' in row 'ROWNUMBER'.

Description !!!UNDOCUMENTED!!!

Message

Identifier CFitting (12)

Message The Fisher information matrix is singular. Therefore, the correlation matrix can not be calculated.

Description !!!UNDOCUMENTED!!!

Message

Identifier CFitting (13)

Message Not enough memory available to calculate the Fisher Information Matrix.

Description !!!UNDOCUMENTED!!!

8.26 COPASI Object

CCopasiObject is the base class of almost all objects in COPASI. If you see one of these error messages, you have very likely found a bug in COPASI. Please provide a bug report.

Message

Identifier CObject (1)

Message Circular dependencies detected for object 'OBJECT'.

Description INTERNAL

8.27 Lyapunov Exponent Calculation

These error messages can occur when calculating the Lyapunov Exponents of a system.

Message

Identifier CLyapMethod (1)

Message Problem is not a Lyapunov exponent problem.

Description INTERNAL

Message

Identifier CLyapMethod (2)

Message Number of exponents needs to be at least one.

Description This means that probably 0 was entered as the number of Lyapunov exponents to be calculated.

Message

Identifier CLyapMethod (3)

Message Only NUM_EXP exponents can be calculated for this model because the model has only NUM_IND independent variables.

Description The maximal number of exponents that can be calculated for a given model is the number of independent variables (that is the number of variables in the model minus the number of conservation relations).

Message

Identifier CLyapMethod (4)

Message Transient time is larger than overall time.

Description The Lyapunov exponent calculation starts after the transient time. The calculation stops after overall time. Therefore overall time should have a bigger value than transient time.

Message

Identifier CLyapMethod (5)

Message Orthonormalization interval is larger than overall time.

Description If the orthonormalization time step is larger than the complete calculation time not a single step can be performed.

8.28 ODE Export

These error messages can occur when exporting a model to a set of ODEs either as C code, Madonna file or XPPAUT file.

Message

Identifier CODEExporter (1)

Message Length of exporting line exeeds 1000 characters.

Description Lines in XPPAUT input files may not be larger than 1000 characters.

Message

Identifier CODEExporter (2)

Message The export was incomplete since the model depends on model quantities, which can currently not be exported.

Description !!!UNDOCUMENTED!!!

8.29 Commercial Version

These error messages can only occur in the commercial version and have to do with license issues.

Message

Identifier CRegistration (1)

Message Invalid registration code.

Description !!!UNDOCUMENTED!!!

Message

Identifier CRegistration (2)

Message Email and/or user name do not match registration code.

Description !!!UNDOCUMENTED!!!

Message

Identifier CRegistration (3)

Message Trial license expired on: 'DATE'.

Description !!!UNDOCUMENTED!!!

8.30 Time Scale Separation Problem

Message

Identifier CTSSAProblem (1)

Message Invalid step size = 'VALUE'.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTSSAProblem (2)

Message The step number 'VALUE' exceeds the limit. The step number and step size have been adjusted to avoid over flow.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTSSAProblem (3)

Message The step size 'VALUE' is too small for the machine accuracy. The step number and step size have been adjusted.

Description !!!UNDOCUMENTED!!!

8.31 Time Scale Separation Method.

Message

Identifier CTSSAMethod (1)

Message Deterministic integration failed. LSODA reported: LSODAERRORMESSAGE HERE Please see result for indications of numerical instability.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTSSAMethod (2)

Message Problem is not a time scale separation analysis problem.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTSSAMethod (3)

Message Internal step limit exceeded.

Description !!!UNDOCUMENTED!!!

Message

Identifier CTSSAMethod (4)

Message Numerical Error encountered.

Description !!!UNDOCUMENTED!!!

8.32 Eigen Value

Errors on Eigen value.

Message

Identifier CEigen (1)

Message Invalid argument 'ARGUMENTID' for dgees.

Description !!!UNDOCUMENTED!!!

Message

Identifier CEigen (2)

Message Failed to compute Eigen value with index 'INDEX'.

Description !!!UNDOCUMENTED!!!

Message

Identifier CEigen (3)

Message Unable to sort Eigen values.

Description !!!UNDOCUMENTED!!!

Message

Identifier CEigen (4)

Message Eigen values do not satify selection criteria after reordering.

Description !!!UNDOCUMENTED!!!

Chapter 9

Bibliography

9.1 References

[Baeck93] T. Bäck and H.-P. Schwefel. *An overview of evolutionary algorithms for parameter optimization.* Evolutionary Computation. 1. 1 - 23. 1997.

[Baeck97] T. Bäck, D.B. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation.* Oxford: IOP Publishing/Oxford University Press. 1997.

[Bell66] M. Bell and M.C. Pike. *Remark on algorithm 178 direct search.* Communications of the Association for Computing Machinery. 9. 684 - 685. 1966.

[Brent72] P.R. Brent. *A new algorithm for minimizing a function of several variables without calculating derivatives.* In Algorithms for minimization without derivatives, (Englewood Cliffs, NJ: Prentice-Hall, Inc.). 117 - 167. 1973.

[Burns85] J.A. Burns, A. Cornish-Bowden, A.K. Groen, R. Heinrich, H. Kacser, J.W. Porteous, S.M. Rapoport, T.A. Rapoport, J.W. Stucki, J.M. Tager, R.J.A Wanders, and H.V. Westerhoff. *Control of metabolic systems.* Trends Biochem. Sci.. 10. 16. 1985.

[Corana87] A. Corana, M. Marchesi, C. Martini, and S. Ridella. *Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm.* ACM Trans. Math. Softw.. 13. 262 - 280. 1987.

[Deuflhard96] P. Deuflhard an J. Heroth. *Dynamic dimension reduction in ODE models.* In Scientific Computing in Chemical Engineering, (F. Keil et al. Springer). 29-43. 1996.

[Fletcher87] R. Fletcher. *Practical methods of optimization.* 2nd Edition. Chichester: John Wiley & Sons. 1987.

[Fogel92] D.B. Fogel, L.J. Fogel, and J.W. Atmar. *Meta-evolutionary programming.* 25th Asiloma Conference on Signals, Systems and Computers. IEEE Computer Society, Asilomar . 540 - 545. 1992.

[Gibson00] M.A. Gibson and J. Bruck. *Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels.* J. Phys. Chem.. A104(9). 1876-1889. 2000.

[Giersch88] C. Giersch. *Control analysis of metabolic networks. 1. Homogeneous functions and the summation theorems for control coefficients.* Eur. J. Biochem. 174. 509 - 513. 1988.

[Gill81] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* London, Academic Press. 1981.

[Gillespie76] D.T. Gillespie. *A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions.* J. Comp. Phys.. 22. 403 - 434. 1976.

[Goldberg89] D.E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, Reading, Mass.. 1989.

[Goldfeld66] S. M. Goldfeld, R. E. Quant, and H. F. Trotter. *Maximisation by quadratic hill-climbing.* Econometrica . 34. 541 - 555. 1966.

[Heinrich74] R. Heinrich and T.A. Rapoport. *A linear steady-state treatment of enzymatic chains.* General properties, control and effector strength. Eur. J. Biochem.. 42. 89 - 95. 1974.

[Heinrich75] R. Heinrich and T.A. Rapoport. *Mathematical analysis of multienzyme systems.* II. Steady-state and transient control. BioSystems. 7. 130 - 136. 1975.

[Hindmarsh83] A.C. Hindmarsh. *ODEPACK, A Systematized Collection of ODE Solvers.* Scientific Computing, R. S. Stepleman et al. (eds.), North-Holland, Amsterdam, IMACS Transactions on Scientific Computation. 1. 55 - 64. 1983.

[Hooke61] R. Hooke and T. A. Jeeves. *"Direct search" solution of numerical and statistical problems.* Journal of the Association for Computing Machinery. 8. 212 - 229. 1961.

[Kacser73] H. Kacser and J.A. Burns. *The control of flux.* Symp. Soc. Exp. Biol.. 27. 65 - 104. 1973.

[Kaupe63] Kaupe. *Algorithm 178 direct search.* Communications of the Association of Computing Machinery. 6. 313 - 314. 1963.

[Kennedy95] J. Kennedy and R. Eberhart. *Particle Swarm Optimization.* Proceedings of the Fourth IEEE International Conference on Neural Networks, Perth, Australia. 1942 - 1948. 1995.

[Kirkpatrick83] S. Kirkpatrick, J., C.D. Gelatt, and M. P. Vecchi. *Optimization by simulated annealing.* Science. 220. 671 - 680. 1983.

[Levenberg44] K. Levenberg. *A method for the solution of certain nonlinear problems in least squares.* Quart. Appl. Math.. 2. 164 - 168. 1944.

[Maier91] W.L. Maier. *A Fast Pseudo Random Number Generator.* Dr. Dobb's Journal. May. 152 - 157. 1991.

[Marquardt63] D.W. Marquardt. *An algorithm for least squares estimation of nonlinear parameters.* SIAM Journal. 11. 431 - 441. 1963.

[Matsumoto98] M. Matsumoto and T. Nishimura. *Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator.* ACM Trans. on Modeling and Computer Simulations . 8. 3 - 30. 1998.

[Michalewicz94] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs.* 3rd Edition. Springer-Verlag, Berlin.. 1994.

[Mitchell95] M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Boston. . 1995.

[Nash84] S. G. Nash. *Newton-type minimization via the Lanczos method.* SIAM Journal of Numerical Analysis. 21. 770 - 788. 1984.

[Nelder65] J. A. Nelder and R. Mead. *A simplex method for function minimization.* Computer Journal. 7. 308 - 313. 1965.

[Petzold83] L. Petzold. *Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations.* SIAM J. Sci. Stat. Comput.. 4. 136 - 148. 1983.

[Powel64] M.J.D. Powell. *An efficient method for finding the minimum of a function of several variables without calculating derivatives.* Comput. J.. 7. 155 - 162. 1964.

[Reder88] C. Reder. *Metabolic control theory: a structural approach.* J. Theor. Biol.. 135. 175 - 201. 1988.

[Runarsson00] T. Runarsson and X. Yao. *Stochastic ranking for constrained evolutionary optimization.* EEE Transactions on Evolutionary Computation. 4. 284 - 294. 2000.

[Surovtsova09] I. Surovtsova, N. Simus, Th. Lorenz, A. König, S. Sahle and U. Kummer. *Accessible Methods for the Dynamic Time-scale Decomposition of Biochemical Systems.* Accepted for Bioinformatics. 2009.

[Swann72] W.H. Swann. *Direct search methods.* Numerical methods for unconstrained optimization., W. Murray, ed. (London & New York: Academic Press). 13 - 28. 1972.

[Vallabhajosyula06] R. R. Vallabhajosyula, V. Chickarmane, and H. M. Sauro. *Conservation analysis of large biochemical networks.* Bioinformatics. 22(3). 346 - 353. 2006.

[Westerhoff84] H.V. Westerhoff and Y.-D. Chen. *How do enzyme activities control species concentrations? An additional theorem in the theory of metabolic control.* Eur. J. Biochem.. 142. 425 - 430. 1984.

[Wolf85] A. Wolf, J. B. Swift, H. Swinney, and J. A. Vastano. *Determining Lyapunov exponents from a time series.* Physica. 16D. 285 - 317. 1985.

[Zobeley05] J. Zobeley et al. *A new time-dependent complexity reduction method for biochemical systems.* In Transactions on Computational Systems (C. Prami et al., Springer). 90-110. 2005.