Chapter 2

Computational Modeling of Biochemical Networks Using COPASI

Pedro Mendes, Stefan Hoops, Sven Sahle, Ralph Gauges, Joseph Dada, and Ursula Kummer

Summary

Computational modeling and simulation of biochemical networks is at the core of systems biology and this includes many types of analyses that can aid understanding of how these systems work. COPASI is a generic software package for modeling and simulation of biochemical networks which provides many of these analyses in convenient ways that do not require the user to program or to have deep knowledge of the numerical algorithms. Here we provide a description of how these modeling techniques can be applied to biochemical models using COPASI. The focus is both on practical aspects of software usage as well as on the utility of these analyses in aiding biological understanding. Practical examples are described for steady-state and time-course simulations, stoichiometric analyses, parameter scanning, sensitivity analysis (including metabolic control analysis), global optimization, parameter estimation, and stochastic simulation. The examples used are all published models that are available in the BioModels database in SBML format.

Key words: Simulation, Modeling, Systems biology, Optimization, Stochastic simulation, Sensitivity analysis, Parameter estimation, SBML, Stoichiometric analysis.

1. Introduction

Biochemical networks are intrinsically complex, not only because they encompass a large number of interacting components, but also because those interactions are nonlinear. Like many other nonlinear phenomena in nature, their behavior is often unintuitive and thus quantitative models are needed to describe and understand their function. While the concept of biochemical networks arose from the reductionist process of biochemistry, where the focus was on studying isolated enzymatic reactions, it is now better

Ivan V. Maly (ed.) *Methods in Molecular Biology, Systems Biology, vol. 500*

DOI: 10.1007/978-1-59745-525-1_2

understood in the framework of *systems biology*, where the focus is on the behavior of the whole system, or at least several reactions, and particularly on what results from the interactions of its parts. Computational modeling is thus a technique of systems biology as important as its experimental counterparts. This chapter covers the definition and analysis of computational models of biochemical networks using the popular software COPASI. It provides essentially a practical view of the utility of several computational analyses, using established models as examples. All software and models discussed here are freely available on the Internet.

From a modeling perspective, biochemical networks are a set of chemical species that can be converted into each other through chemical reactions. The focus of biochemical network models is usually on the levels of the chemical species and this usually requires explicit mathematical expressions for the velocity at which the reactions proceed. The most popular representation for these models uses ordinary differential equations (ODEs) to describe the change in the concentrations of the chemical species. Another representation that is gaining popularity in systems biology uses probability distribution functions to estimate when single reaction events happen and therefore track the number of particles of the chemical species. As a general rule, the latter approach, known as stochastic simulation, is preferred where the numbers of particles of a chemical species is small; the ODE approach is required when the number of particles is large because the stochastic approach would be computationally intractable.

1.1. ODE-Based Models

Each chemical species in the network is represented by an ODE that describes the rate of change of that species along time. The ODE is composed by an algebraic sum of terms that represent the rates of the reactions that affect the chemical species. For a chemical species X:

$$\frac{\mathrm{d}X}{\mathrm{d}t} = \sum_{\text{all reactions } i} s_i \cdot v_i, \qquad (1)$$

where s_i is a stoichiometry coefficient that is the number of molecules of X consumed or produced in one cycle of reaction i, with a positive sign if it is produced or negative if consumed, and v_i is the velocity of reaction i. Obviously, for reactions that do not produce or consume X the corresponding s_i is zero.

The velocity of each reaction is described by a *rate law* that depends on the concentrations of the reaction substrates, products, and modifiers (*see* **Note 1**). Rate laws are the subject of chemical and enzyme kinetics and are generally nonlinear (except the case of first-order mass action kinetics). Often these rate laws are saturable functions, i.e., have finite limits for high concentrations of substrates, products, and also for many modifiers (*see* **Note 2**).

An example of a rate law is depicted in **Eq. 2**, which represents a rate law of reaction with one substrate (S), one product (P), and a competitive inhibitor (I)

$$v = \frac{\frac{E \cdot k_{\text{cat}}}{K_{mS}}\left(S - \frac{P}{K_{\text{eq}}}\right)}{1 + \frac{S}{K_{mS}} + \frac{P}{K_{mP}} + \frac{I}{K_I}}. \qquad (2)$$

In **Eq. 2**, the limiting rate of reaction ("V_{max}") is directly represented as a product of the concentration of the enzyme and the turnover number ($E{\cdot}k_{\text{cat}}$). It is usually good practice to make this product explicit, since it is then possible to have the enzyme concentration be a variable of the model too. This is important if the model includes protein synthesis, degradation, or protein–protein interactions.

These ODE models can be used to simulate the dynamics of the concentrations of the chemical species along time given their initial values. This is achieved by numerical integration of the system of ODE which can be carried out with well-established algorithms (for example *(1, 2)* but *see* **Note 3**). It is also useful to find steady states of the system, which are conditions when the concentrations of the chemical species do not change. If the steady state is such that the fluxes are also zero, then the system is in chemical equilibrium, otherwise the fluxes are finite meaning that the concentrations do not change because the rates of synthesis balance with the rates of degradation for every chemical species. Steady states can be found using the Newton–Raphson method which finds the roots of the right-hand side of the ODE (which must be zero by the definition of steady state). Alternatively steady states can also be found by integration of the ODE. COPASI can use either one of these strategies or a combination of the two (*see* **Note 4**).

Other model analyses can be carried out but a description of their theory in any detail is beyond the scope of this article. Some of them are described at a high level in **Subheading 3**, whenever they are used.

1.2. Stochastic Models

When analyzing a biochemical system which contains small numbers of particles of each reactant, the assumption of continuous concentrations fails and consequently the underlying basis of the ODE representation also fails. Moreover, in such conditions, stochastic effects become more pronounced and may lead to dynamics that differ significantly from those that would result from the ODE approach. In the conditions described above, one should then use a stochastic discrete approach for the simulation of the system dynamics.

Stochastic models represent the number of particles of each chemical species and use a reaction probability density function (PDF) to describe the timing of reaction events. Gillespie developed a Monte Carlo simulation algorithm, known as the stochastic simulation algorithm (SSA) first reaction method, that simulates the stochastic dynamics of the system by sampling this PDF *(3, 4)*. The theoretical derivation of this method is too involved to be described here, and the reader is directed to the original publications *(3, 4)* or a recent review *(5)*. It is important to stress that one simulation run according to this approach is only one realization of a probabilistic representation, and thus provides limited amount of information on its own. In the stochastic formalism, it is very important that simulations are repeated for a sufficient number of times in order to reveal the entire range of behavior presented by such a system (i.e., to estimate a distribution for each chemical species and its dynamic evolution).

2. Materials

2.1. Copasi

The software COPASI *(6)* will be used throughout this chapter. COPASI is freely available for noncommercial use (*see* **Note 5**) and executable versions are provided for the most popular operating systems: Microsoft Windows, Apple Mac OS X, Linux, and Sun Solaris. The source code of COPASI is also available under an open source license and so it can be compiled for other architectures.

New versions of COPASI are released often and there is a distinction between *stable* and *development* versions. Development versions are those where new features are introduced; stable versions have no new features and differ from the previous development release only by having bug fixes. While testing is more intense in stable releases, the reader is encouraged to download the latest development release. Irrespective of being stable or development releases, COPASI releases are labeled with a *build number*, which is sequential (*see also* **Note 6**).

2.1.1. Installing COPASI

Instructions for installation of COPASI depend on the operating system version, but all start with downloading the appropriate binary from the project's Web page http://www.copasi.org. Choose the *download non-commercial* option from the site's menu and then select the appropriate version for your platform. Download will proceed after selecting the nearest server and accepting the license terms. Once the file has finished downloading, the installation instructions are different for each platform.

For Microsoft Windows, the downloaded file, Copasi-XX-WIN32.msi, is an installation program which you should run by

double clicking it. For Apple OS X, the downloaded file, Copasi-XX-Darwin.dmg, is a disk image containing a folder named "copasi." You can either start COPASI directly from the disk image or drop the folder into your applications folder and start it from there.

Finally, for Linux or Solaris, you need to unpack the archive where you want to install it (it can be in a system-wide location like /usr/local/copasi, or in a user home, such as ~/copasi). For optimal performance you should set the environment variable COPASIDIR to /usr/local/copasi (or wherever you have installed it).

2.2. BioModels Database

"BioModels" is a database that archives biochemical models that have previously been published in peer-reviewed journals *(7)*. Some examples in this chapter use models that are available there and so to avoid entering those models manually it is best to download them from BioModels.

Models in this database are primarily distinguished by their identifier, which is in the form BIOMDxxxxxxxxxx where the x's represent a number. To download a specific model, point your Web browser to http://www.ebi.ac.uk/biomodels/, select the *search* option, type the model ID in the search box, and then click on the link for that model's page (*see* **Note 7**). Once in the model's page you can examine the model's characteristics, including the citation of the original publication, who was the author of the model, etc. To download the model in a format that COPASI (and most other systems biology software) can read select the link entitled **SBML L2 V1** (*see* **Note 8**) at the very top of the page, and download it to your local computer (*see* **Note 9**). This will be a file entitled BIOMDxxxxxxxxxx.xml which COPASI can import.

3. Methods

3.1. Model Construction and Basic Simulation

3.1.1. Model Specification

The COPASI user interface is composed of two main areas: a hierarchical organization of functions on the left (a *tree*), and a larger area on the right which contains the controls related with the function selected on the tree. All features related with model specification reside on the first main entry of the tree, appropriately named *Model*. When this entry is selected on the left, the right displays the basic information about the model, such as its name, the units used, and a large field for comments (*see* **Note 10** and **11**). Expanding the *Model* subtree reveals two other entries: *Biochemical* and *Mathematical*, these are different views of the model. Specification of a new model is done in the *Biochemical* part as the other is only for examining equations and matrices.

The most practical way to enter a model is to start by adding its component reactions. Select *Reactions* and then double click the first empty row of the table on the right. This will change to display the detailed reaction window. Enter a name for the reaction and type its chemical equation, for example "NAD + ethanol = acetaldehyde + NADH" (*see* **Note 12**). The equals sign has quite a specific meaning, not only does it separate substrates from products, but it also means the reaction is considered kinetically reversible. If you want the reaction to be irreversible then you should use instead the combination of characters "->" (*dash* and *right angle bracket*).

After entering the reaction equation, you should select the appropriate rate law for this reaction. COPASI only allows selecting rate laws that match the characteristics of the reaction entered: same number of substrates and products and reversibility. You can chose a rate law from the menu; if the appropriate one is not available, then you can add one yourself by pressing the *New Rate Law* button. Type the rate law in the box, for example: $V/K*(A*B-P*Q)/(K + A + B + P + Q)$. Next select the appropriate type of each symbol in the equation (*see* **Note 13**). You should also mark the reaction as *reversible* or *irreversible* (*see* **Note 14**). **Figure 1** shows this window when entering the above rate law. When you finish press *Commit* and go back to *Model Reactions* where you can now select this rate law for the reaction (assuming it was reversible with two substrates and two

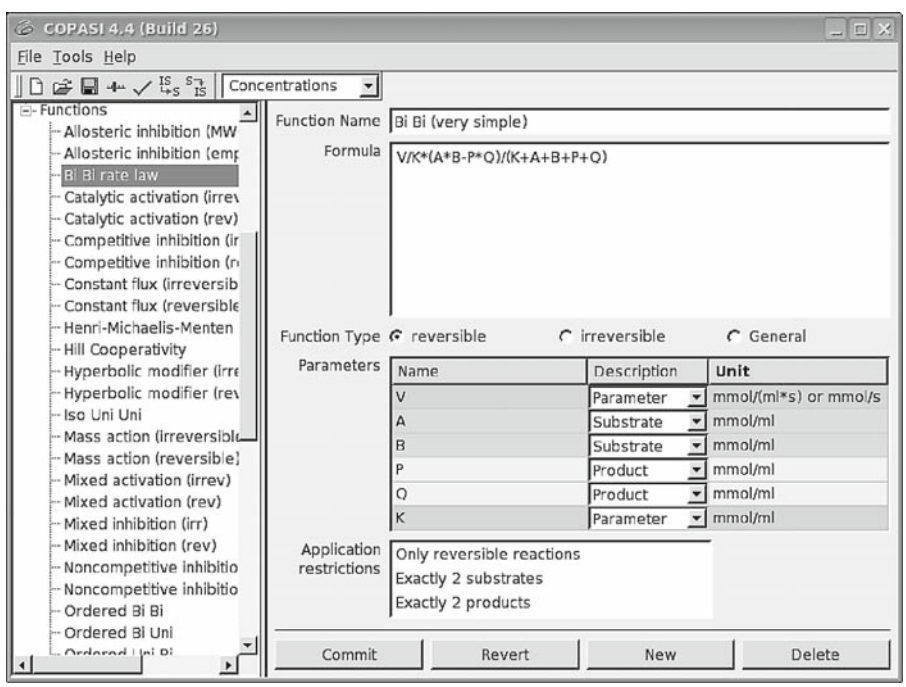


Fig. 1. Definition of a kinetic rate law.

products). At that point you will be able to enter the values of the parameters (in this case *V* and *K*).

You can define more compartments and change their sizes (volumes) on the entry named *Compartments*. By default there is always a compartment called simply "compartment" of unit volume. You can define any number of compartments of any positive size.

After entering all reactions you can examine the chemical species by selecting *Species* on the tree which will show a summary table with all species included in the reactions. You can change their initial concentrations, set their compartment and change their type. The species type determines how its concentration (actually its particle number) is calculated in the models: it can be set by *reactions*, which means that its concentration will be determined by the ODE generated from the reactions or by the SSA; it can be *fixed*, meaning that it becomes a parameter of the model; set by *assignments*, which are algebraic expressions (see below); or set by arbitrary *ODE* (i.e., entered directly by the user). You can add extra species directly at the end of the table. If you double click any row, you will then see a more detailed page for that species alone, which additionally lists all reactions where the species is involved in.

The entry marked *Global Quantities* in the tree is to add explicit mathematical expressions that are to be calculated in the model (unlike the ODE that are defined implicitly from the reaction stoichiometry and rate laws). There are three types of global quantity (1) *fixed*, which are arbitrary constants; (2) *assignment*, which are new variables that have their value calculated by algebraic expressions; or (3) *ODEs*, which are new variables that are determined by an explicit ODE. These global quantities are useful to expand the model to include features that are not directly linked with the biochemistry. As an example, suppose you would want to calculate the ratio of NADH/(NAD + NADH) at all times in your model, either because you just want to monitor it, or maybe you want to make it affect something else. Then you should double click the list of global quantities to enter a detailed form. There you should enter its name, select the type as *assignment*, then enter the expression in the larger box. Note, however, that you are not allowed to type "NADH" or "NAD," since these are variables of the model (species) and you will instead have to select them from a dialog box: press the small button that has the COPASI logo (*see* **Note 15**), then select *Species*, *Transient Concentrations*, and select "[NADH](t)"; the division sign and the brackets are typed directly. ODEs are set the same way, except that the expression is now the right-hand side of the differential equation that will be integrated in the model simulations.

3.1.2. Importing and Exporting SBML

The popular SBML format is used as a means of sharing models between systems biology software, so an important feature of

COPASI is that it can indeed read and write models in this format. However, it is important to realize that COPASI can represent a small number of features that are not possible to be specified in SBML and those will be lost on export to SBML. On the other hand there are features of SBML that are not yet implemented in COPASI – when loading files with such features a warning is produced such that you are aware of this fact. When importing SBML there are also often warnings about issues with the files that are either not like the specifications require, or because they follow bad practices. In any case, even with warnings, COPASI almost always succeeds in importing the model if not totally, at least partially.

3.1.3. Time Course

Once a new model has been entered or loaded it is ready to be used for simulation. There are two basic types of simulation: *Time Course* and *Steady State* which are entries under the *Tasks* branch. Let us use model number 10 of BioModels, a MAP kinase model *(8)*, to illustrate these basic tasks.

To run time course, select the appropriate entry on the tree on the left and the time-course control window will appear on the right. You have to decide for how long you want to run the time course (in model time, not real time) and enter that value in the box labeled *Duration* (enter 1,000 for this example); you also need to decide how many *Intervals* in the time course you will want to sample, or alternatively the *Interval size* (when you set one, the other one updates automatically). Below are several control variables of the numerical methods, which are outside the scope of this article. Simply press the *Run* button to carry out the simulation, which will take place very fast. Expand the *Time Course* entry on the tree to reveal *Results* and select it. This displays a table with the numerical values of the time series, which can be saved to a file (button *Save data to file*).

It is also very useful to visualize the results of a time series simulation in a plot. To create it press the button *Output Assistant* (located at the level of *Time Course*) and select the first line entitled "Concentrations, Volumes, and Global Quantity Values," then press the button *Create!*. This creates a plot definition that will plot all variables, however, the plot is constructed while the simulation runs, and thus you must run it again to make the plot appear. The legend of the plot is composed of buttons, one for each curve, and by pressing them you select/deselect that curve from being displayed.

3.1.4. Steady-State Simulations

Another important task is the calculation of a steady state of the model. Select the *Steady State* entry under *Tasks*. The control variables of the steady state deserve some attention, mainly the *Steady-state resolution*, which is the smallest value of a change in species concentration that is the smallest distinguishable from

zero. This value will be used to decide when to stop iterations, but also to recognize a steady state. Smaller values of this parameter lead to more accurate solutions. Another important set of control variables are named *Use Newton, Use Integration, Use Back Integration*, which are related to the strategy used to find the steady state. These variables take the values 1 or 0 meaning to use them or not, respectively. The Newton method is a solver for nonlinear algebraic equations which is very fast. However, the Newton method is not guaranteed to converge, therefore the integration method can be used to help. Integration is the method used to calculate a time course – this method attempts to find a steady state as it goes along a time course. The back-integration method is a fail safe device that is used when the other two cannot converge and may be able to find an *unstable* steady state. Pressing the *Run* button will trigger all calculations. The results are also in a branch of the tree, below the *Steady state* and can also be saved to a file like the time course results.

3.1.5. Scanning and Sampling Parameters

One of the most frequent aims of using models to study biochemical networks is to find out how certain parameters affect several aspects of the system. Thus it is likely that one needs to carry out several steady-state and/or time-course simulations at different values of the parameters of interest. COPASI supports this activity by providing a flexible scheme for changing parameter values with associated simulations, which is termed *Parameter Scan* and is under the tree branch *Multiple Task* (*see* **Note 16**). The *Parameter scan* window (**Fig. 2**) is an interface that allows us to specify a series of hierarchical changes in model parameters which culminate with the execution of a task (e.g., time-course or a steady-state simulation).

We will use here the model of the branch point of threonine/methionine biosynthesis of Curien et al. *(9)*, which is number 68 in BioModels. This is a very simple model of the branch point with only one variable chemical species that has one input and two output fluxes. Curien et al. study the effect of the Cysteine (Cys) and S-Adenosylmethionine (AdoMet) on the partition of the output fluxes. One issue that you may wonder about is that while AdoMet is a chemical species, the authors of the SBML file decided to represent it as a constant in the kinetic rate law of the enzyme threonine synthase (TS). While this is not incorrect, it would have been clearer to define it as a chemical species with fixed concentration.

We will first investigate the effect of AdoMet on the partition of fluxes. In order to be able to visualize the results, we must first define a plot where the flux of the TS and CGS are plotted as a function of AdoMet. Plots are defined under the main hierarchy *Output* and then under *Plots*. A list of plots is displayed (currently empty) and there you should double click the last empty

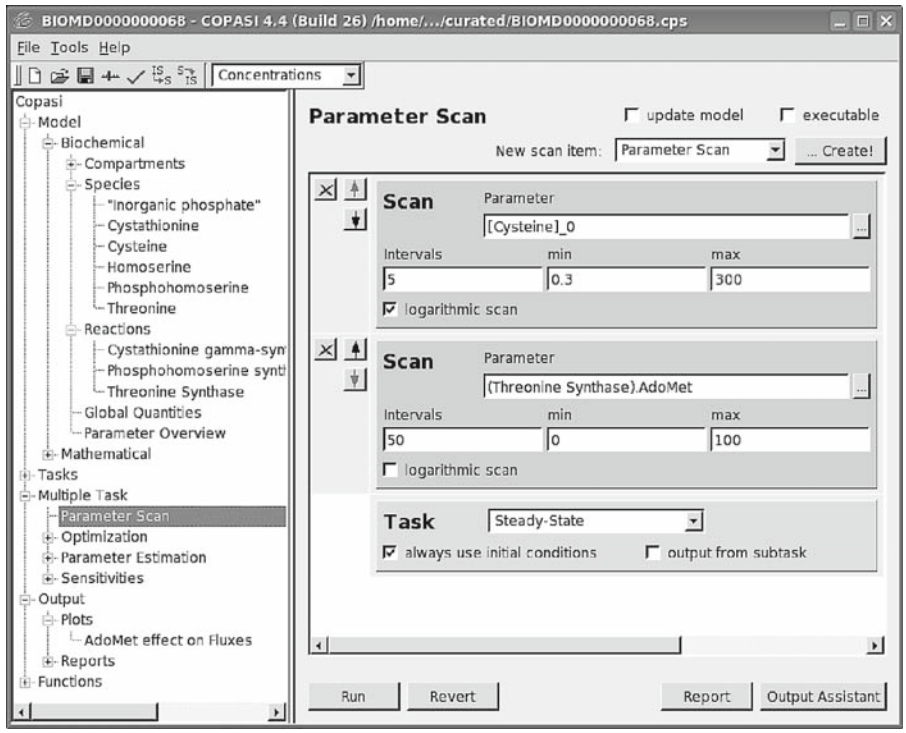


Fig. 2. Parameter scan window. The structure on the center right is a stack of operations that are carried out in order. Thus the example shown is for changing the initial concentration of Cysteine between 0.3 and 300 in five intervals spaced logarithmically, then for each of those change the parameter AdoMet between 0 and 100 in 50 equally spaced intervals, and finally to run a steady-state calculation for each value of the above.

row which will generate the plot definition window. There you should define the name of the plot ("AdoMet effect on Fluxes" is suggested) and then create a new curve (press *New curve*) where you select the parameter AdoMet in the *X*-axis (under *Reactions-Reaction Parameters-Threonine synthase-AdoMet*) and flux(Cystathionine gamma-synthase) for Y-axis (under *Reactions-Concentration fluxes-flux*(*Cystathionine gamma-synthase*)). Because you want to plot both fluxes in the same plot, you should then create a new curve again and select the same item for the *X*-axis, but then flux (*Threonine Synthase*) for the Y-axis. At this point each curve in the plot is represented under a different tab which have long titles; it is advisable to make the titles of each curve smaller strings for esthetic reasons, rename the first one to J(CGS) and the second to J(TS). The plot definition is ready; you can go back to *Parameter Scan*.

The task to be carried out in this case is *Steady state* and we want to scan the parameter AdoMet, so in *New scan item* at the top select *Parameter scan* and then press…*Create!* A new entry will appear in the stack above the steady-state task. There you need to select the parameter to change, press the button

marked "…" and select *Reactions, Reaction Parameters, Threonine synthase, AdoMet*. The minimum and maximum values that this parameter will be changed also have to be entered, for example 0 and 100 (as in **Fig. 2** of **ref.** *9)*, and finally the number of intervals desired, a value of 50 will produce a smooth curve. At the bottom of the stack, in the Tasks slot (*see* **Fig. 2**), you should disable the check *output from subtask* since we only want the final estimate of the steady-state calculation (*see* **Note 17**). At this point you can press *Run* and see the result appear as a plot in a new window (**Fig. 3**). The results plotted can be saved by selecting the menu entry *Save data*. You can also switch on or off each of the curves, simply by pressing its entry in the legend. This plot shows that increasing values of AdoMet push the flux toward the CGS reaction, as shown in the original *(9)*.

Let us now ask whether the behavior changes with different values of Cys. To do this we simply add this model entity to the scan and therefore perform a two-dimensional scan. In the *Parameter Scan* window, add another scan item by pressing …*Create!* once again. A new slot appears where you need to select *Species, Initial concentrations, [Cysteine](t = 0)*. Set it to vary between 0.3 and 300 with five intervals and tick *logarithmic scan* (*see* **Note 18**). You can also move this slot to the top by pressing the up arrow on the left (*see* **Note 19**). Press *Run* again, and now you will see several curves for each of the fluxes plotted. Each of them is for different values of Cys. As you can see, Cys acts by affecting the initial flux partition and also by moving the value of AdoMet where both fluxes are equal.

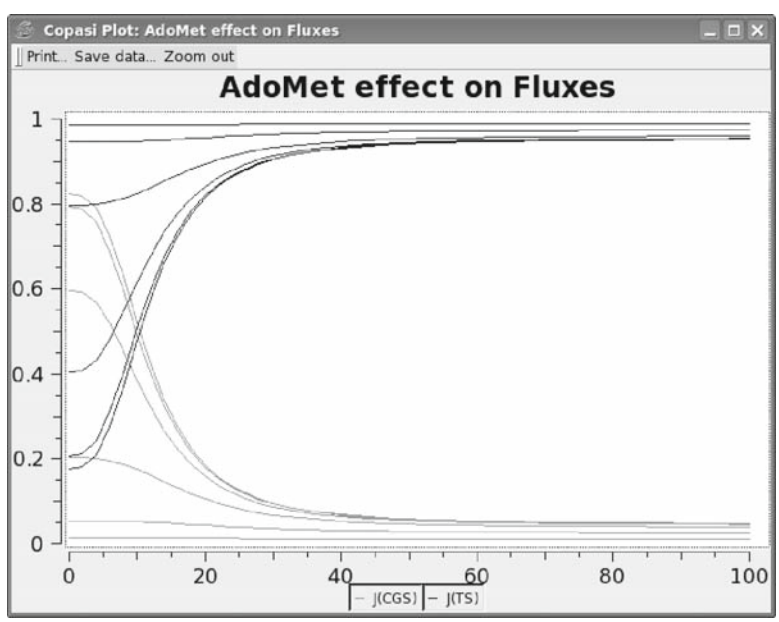


Fig. 3. Results of a two-dimensional parameter scan.

Finally, we will see how to do a random sample, rather than a scan. We shall probe 10,000 random values of the two parameters in the same range. For this remove the two slots of AdoMet and Cys by pressing the button marked X. Now select *Repeat* in the *New scan item* at the top and create a new slot; set the number of iterations to 10,000. Next select *Random distribution* in the *New scan item* and select the parameter AdoMet as above, and set it to the same limits as above. Repeat the same for Cys, also with the same limits. You can have both sampled from a uniform distribution. Note that the stack of operations should be read from the top and it means: repeat 10,000 times a random value of AdoMet and a random value of Cys and calculate the steady state. To visualize these results it is best to just plot symbols and not connect them with lines, so you have to go back to the plot definition and change *Type* to *symbols* for each of the curves (it was *lines*). Go back to *Parameter Scan* and press *Run*. The run now takes some more time (there are 10,000 simulations, after all) and finally you should obtain a plot as in **Fig. 4**. Each point plotted represents the steady-state flux for a pair of values of AdoMet and Cys. It is very easy to add more parameters to this sampling, by adding more slots associated with those parameters and moving it below the *Repeat* slot. It is even possible to combine a sequence of repeats below scans below other repeats, etc. This feature of COPASI is a very powerful means to program complex simulations.

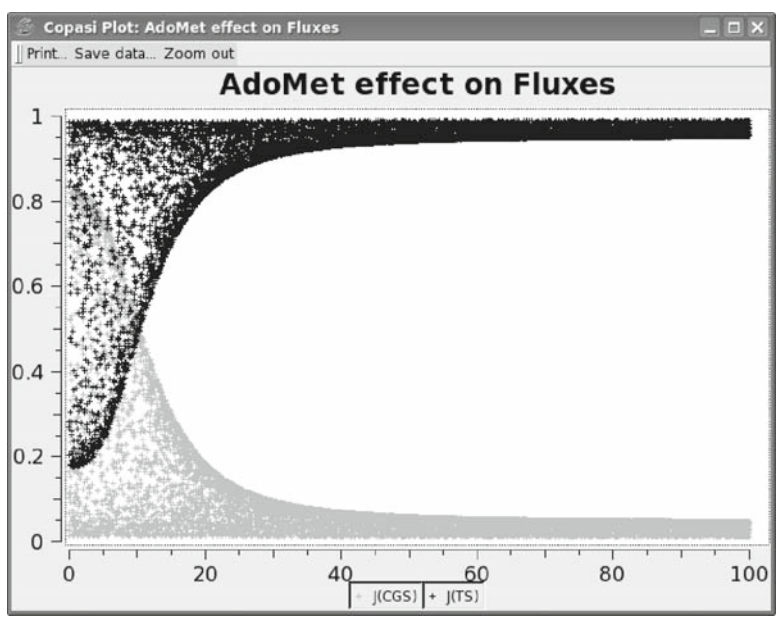


Fig. 4. Results of a two-dimensional parameter random sampling.

3.2. Stoichiometric Analyses

While the analysis of the dynamics of biochemical models is seen as the ultimate goal of these models, some properties of the model reveal themselves even without considering the kinetics of the reactions involved. These properties are sometimes known as *structural properties* because they depend only on the structure of the network, or as *stoichiometric properties* because they depend only on the stoichiometric coefficients of **Eq. 1**. COPASI provides two stoichiometric analyses (1) identification of elementary flux modes *(10, 11)* and (2) identification of mass conservation relations *(12)*.

3.2.1. Elementary Flux Modes

Elementary flux modes are the minimal subsets of reactions that would still be able to maintain a steady state if isolated from the rest of the network *(10)*. They are the basic components of flux and any observable flux is a linear combination of these. They can also be seen as "functions" that the network fulfills because they represent parts of the network that could still operate even when the rest of the network had been removed. They can be useful to identify what functions would be lost by removing a specific reaction from the network (e.g., by a gene knockout) and also to calculate maximal yields of a certain end product that can be obtained from some substrate *(11)*.

Let us use a model of erythrocyte metabolism by Holzhütter *(13)*, which is model 70 in the BioModels database. Download the corresponding SBML file and import it into COPASI as described before. You can examine the model by inspecting the various categories under the *Model* section, where you will find that it is composed of 38 reactions. The *Elementary Modes* task is under *Tasks-Stoichiometry*. To run this task simply press *Run* button on the bottom left of the right pane as there are no other choices to make. The table on the right pane should now be filled and at the top there is an indication that the model can be decomposed into 105 elementary modes. The table, depicted in **Fig. 5**, lists each of the modes in detail. The first column indicates whether the mode is reversible or irreversible (*see* **Note 20**); the second column lists the reactions that compose the mode and the third column lists the actual reaction equations. Note that in the second column, the name of the reaction is preceded by a number which is a multiplier for that flux, and if the number is negative then the flux of that reaction goes in the reverse direction in this elementary mode. In the mode depicted in **Fig. 5**, Glucose transport operates in the reverse direction (glucose is exported) while Bisphosphoglycerate mutase operates in the forward direction threefold faster than Glucose transport.

Elementary flux modes can either link a source to a sink (external substrates and products of the model) or be cyclic. The overall chemical reaction of the mode depicted in **Fig. 5** is 2*External Lactate + PRPP = 2*External Pyruvate + 3 External Phosphate + Glucose outside (*see* **Note 21**). This erythrocyte

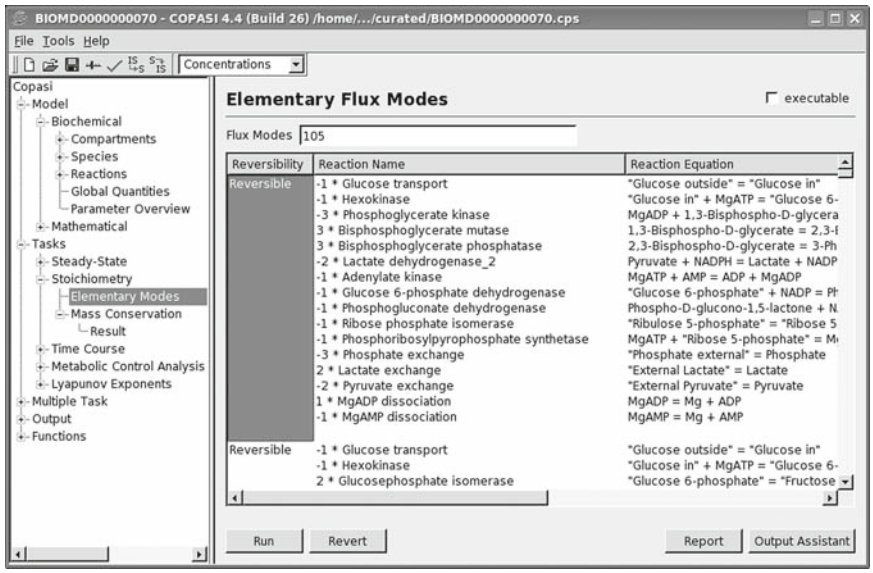


Fig. 5. Elementary flux modes.

model also contains cyclic flux modes. In the list of 105 there is, for example, a reversible mode composed of Phosphoglycerate kinase, Bisphosphoglycerate mutase, Bisphosphoglycerate phosphatase, and ATPase. Cyclic modes have no net production or consumption of any metabolite (thus they are sometimes called *futile cycles*).

To save the results of the elementary mode analysis you first need to set a report file: press the button labeled *Report* on the bottom right, then press *Browse* and enter a filename for your report in the desired folder. You will then have to press the *Run* button again in order to create the report. The report is a tab-delimited text file that contains a table with the same information as displayed in the front-end. You can read this file with a plain ASCII text editor, such as "wordpad" in Windows; you can also import this file into a spreadsheet program like "Excel."

3.2.2. Mass Conservation Relations

Mass conservation relations are algebraic sums of amounts of chemical species that are constant in any state of the model. These algebraic sums imply that the amounts of some chemical species are constrained, such that one of them can be directly calculated from the others using the algebraic expression. A special case of mass conservation relations is when there is conservation of a chemical moiety (*see* **Note 22**).

Let us continue with the erythrocyte model, and examine the mass conservation relations that it contains. The *Mass Conservation* task is also under *Tasks-Stoichiometry* and is also run by pressing the *Run* button on the bottom left of the right pane.

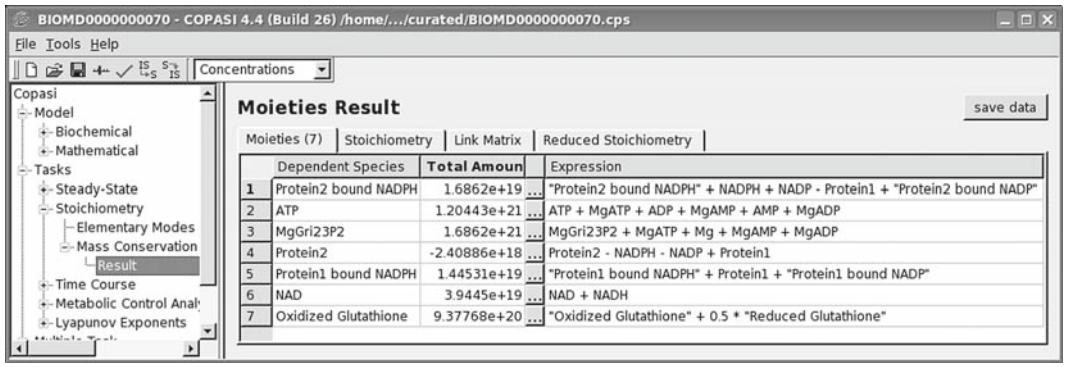


Fig. 6. Mass conservation relations.

The results of this analysis will appear in a new entry marked *Results* that appears below *Mass Conservation* (in the tree on the left), which you have to select to inspect the results.

The erythrocyte model has seven mass conservation relations as shown in **Fig. 6**. The results are listed in a table where the first column identifies the chemical species that COPASI will calculate from the mass conservation (the *dependent* species, *see also* **Note 23**). The second column lists the total number of particles of this conservation relation. The third column contains a button labeled "…" which creates a new global quantity that mirrors the total number of particles. Finally, the fourth column contains the actual expression which is constant. In the erythrocyte model, the first of these relations reads: "Protein2 bound NADPH" + NADPH + NADP – Protein1 + "Protein2 bound NADP" = 1.6862×10^{19}. That means that adding the number of particles of all the species with a positive sign and subtracting those with a negative sign adds up to 1.6862×10^{19} particles. This algebraic expression is constant throughout any condition of this system, except when the initial amounts of any of the chemical species involved change (in which case the total would be different). In particular, this expression is always true during any time course and thus does not depend on the dynamics of the system. The reader may recognize the second relation in **Fig. 6** to be the conservation of the Adenine moiety, the third is conservation of Mg, the fifth is conservation of Protein 1, the sixth is conservation of the NAD moiety, and the seventh conservation of the glutathione moiety. Together, relations 1 and 4 represent the conservation of the Protein 2 moiety (when summing the two, NADP, NADPH, and Protein 1 cancel out, leaving just the Protein 2 forms). Together, relations 1, 4, and 5 represent the conservation of the NADP moiety and also that its total is inversely related with the total of the free protein forms (expressed by the result of computing relation 1 – relation 4 + relation 5). This last complex

relation appears due to the fact that whenever the free forms of the proteins react they always do it with NADPH or NADP – the three moieties (NAD, Protein 1, and Protein 2) are intertwined.

Note that there are other results of this task, which can be inspected by selecting the tabs *Stiochiometry*, *Link Matrix*, and *Reduced Stoichiometry*. These are the matrices that are used to calculate these conservation relations and are described in the theoretical derivations of Reder *(12)*. To save all of the results of this task, just press the *Save data* button on the top right corner, which creates a tab-delimited ASCII file.

3.3. Sensitivity Analyses

As discussed in the context of parameter scans, it is frequently desirable to investigate the behavior of a model systematically. In addition, every model contains a number of parameters (kinetic constants, initial concentrations, and so on) whose values are not all known exactly. Changing the values of the parameters will of course change the behavior of the model, so it is interesting to find how much the model depends on parameters. Sensitivity analysis describes how much does a specific parameter change the behavior of the model. This is useful for several reasons:

- In many cases the value of a parameter is unknown. For example, while K_m values of enzymes can be measured relatively easily in vitro, often the enzyme concentrations in vivo are not well known. In this situation, sensitivity analysis can tell us if it is important to know a specific parameter value. If a parameter is found not to affect the system very much, a rough guess for its value may be sufficient. If, on the other hand, a parameter influences the behavior of the model significantly, steps must be taken to find out its value more accurately, either by executing more experiments or by literature searches.
- Sometimes the aim of research is to change the behavior of the system. Perhaps we want to increase the yield of some biotechnological production process, or to find a drug that inhibits a metabolic pathway. Sensitivity analysis can give hints about which parameters should be changed to achieve a specific effect.
- Robustness with respect to external influences is an important property of biological systems. Living organisms need to be able to function under a wide range of environmental conditions. This means some biological processes need to be rather insensitive to parameter changes. On the other hand an organism needs to react to its environment, so other processes need to be very sensitive to external influences. Therefore robustness (or the lack of robustness) is an interesting property of biological systems, and sensitivity analysis is a way to determine this.

One should note that sensitivities as they will be described below are only able to provide a local description of robustness. This means that its results are only valid for a given parameter set (set of environmental conditions). If several parameters were to change at the same time then the individual sensitivity coefficients would also be expected to change. COPASI contains two frameworks for doing sensitivity analysis: metabolic control analysis (MCA) and generic sensitivities.

3.3.1. Metabolic Control Analysis

MCA is a concept developed by Kacser and Burns *(14)* and Heinrich and Rapoport *(15)*. Its most practical formulation deals only with steady states (*see* **Note 24**) and provides means to quantify how much the rates of the various reactions of a network affect the concentrations and fluxes at the steady state. A deeper description of the theory does not fit this text and the reader is directed to specialized reviews and books *(16–19)*.

As an example we use a model of sucrose accumulation in sugar cane *(20)*, model 23 in BioModels. After importing the SBML file into COPASI select *Tasks* and *Metabolic Control Analysis* in the tree on the left (*see* **Note 25**). Then simply click the *Run* button on the right. The *Results* window then presents a screen with three tabs labeled *Elasticities* (**Fig. 7**), *Flux Control Coefficients* (**Fig. 8**), and *Concentration Control Coefficients*.

The *elasticity coefficients* (or simply elasticities) quantify the amount of change of a reaction rate with the change in concentration

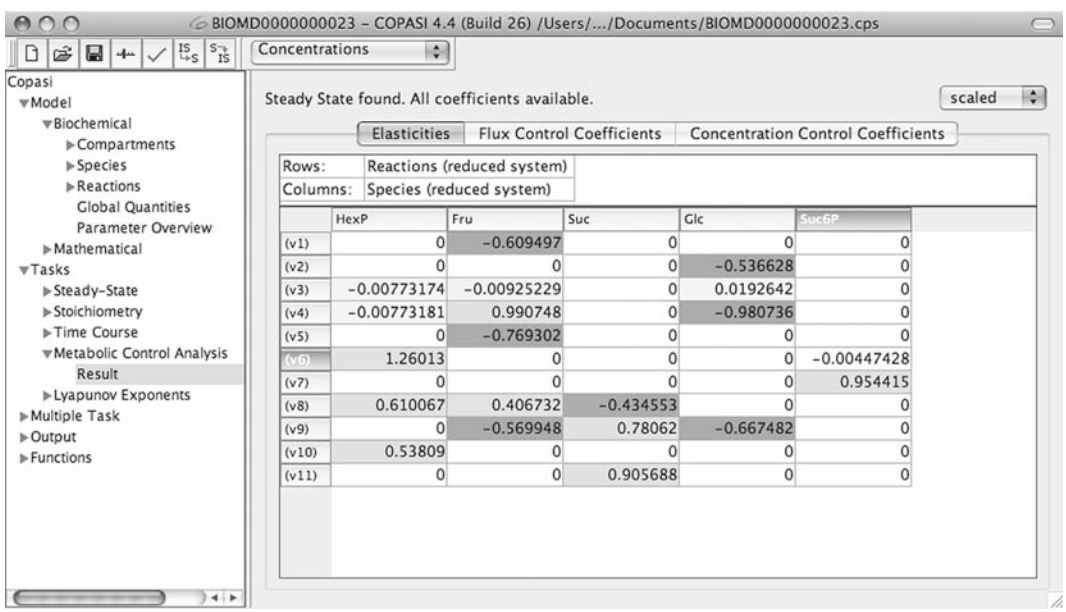
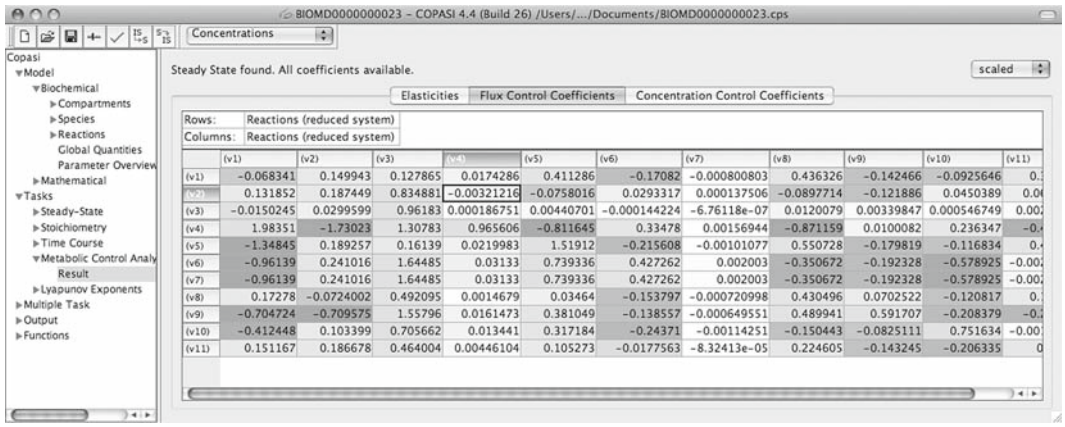


Fig. 7. Display of elasticity coefficients. Note that the cells of the matrix are colored according to the magnitude of the values, green for positive values and red for negative (colors not shown in this figure).



	(v1)	(v2)	(v3)	(v4)	(v5)	(v6)	(v7)	(v8)	(v9)	(v10)	(v11)
(v1)	-0.068341	0.149943	0.127865	0.0174286	0.411286	-0.17082	-0.000800803	0.436326	-0.142466	-0.0925646	0.1
(v2)	0.131852	0.187449	0.834881	-0.00321216	-0.0758016	0.0293317	0.000137506	-0.0897714	-0.121886	0.0450389	0.0
(v3)	-0.0150245	0.0299599	0.96183	0.000186751	0.00440701	-0.000144224	-6.76118e-07	0.0120079	0.00339847	0.000546749	0.00
(v4)	1.98351	-1.73023	1.30783	0.965606	-0.811645	0.33478	0.00156944	-0.871159	0.0100082	0.236347	-0.
(v5)	-1.34845	0.189257	0.16139	0.0219983	1.51912	-0.215608	-0.00101077	0.550728	-0.179819	-0.116834	0.
(v6)	-0.96139	0.241016	1.64485	0.03133	0.739336	0.427262	0.002003	-0.350672	-0.192328	-0.578925	-0.00
(v7)	-0.96139	0.241016	1.64485	0.03133	0.739336	0.427262	0.002003	-0.350672	-0.192328	-0.578925	-0.00
(v8)	0.17278	-0.0724002	0.492095	0.0014679	0.03464	-0.153797	-0.000720998	0.430496	0.0702522	-0.120817	0.1
(v9)	-0.704724	-0.709575	1.55796	0.0161473	0.381049	-0.138557	-0.000649551	0.489941	0.591707	-0.208379	-0.
(v10)	-0.412448	0.103399	0.705662	0.013441	0.317184	-0.24371	-0.00114251	-0.150443	-0.0825111	0.751634	-0.00
(v11)	0.151167	0.186678	0.464004	0.00446104	0.105273	-0.0177563	-8.32413e-05	0.224605	-0.143245	-0.206335	0

Fig. 8. Display of flux control coefficients.

of a certain chemical species. The elasticities of all the reactions with respect to all the species in the model are calculated by COPASI and displayed in a table where the columns correspond to the species and the rows to the reactions. Consider the line labeled "(v8)" (**Fig. 7**): the numbers in this line describe how the flux of reaction *v8* (HexP + Fru = Suc + UDP) changes with changes of the concentrations of the different species. Notice positive values for "HexP" and "Fru," which are the substrates of the reaction. This means an increase of 1% in Fructose concentration would increase the speed of the reaction by 0.61% (*see* **Note 26**). Correspondingly the elasticity with respect to the product (*Suc*) is negative – an increase in product concentration would lead to a lower flux. Another case, in line "(v4)" the negative value for *Glc* indicates that glucose is an inhibitor for this reaction. An elasticity equal to zero means that the metabolite concentration has no influence on the reaction rate (*see* **Note 27**).

The elasticities are properties strictly of a single reaction and are independent of the rest of the system (the elasticity of reaction A toward species B does not depend on reactions C, D, etc.). The calculation of elasticities is carried out only from the kinetic rate law of the respective reaction. Likewise, in an experiment the elasticity could be measured in vitro using the purified enzyme, so long as the concentrations of its substrates and products are set to their physiological value (and the enzyme properties remain the same after purification).

Note that in COPASI all sensitivities (i.e., MCA and generic sensitivities) can be displayed with either scaled or unscaled values. The scaled values describe relative changes, e.g., a scaled sensitivity of 0.5 means that if the parameter is increased by 10% the target value will increase by 5% (0.5 times 10%). The unscaled sensitivities describe absolute changes, e.g., an unscaled elasticity of 0.5 could mean that increasing the substrate concentration by

1 μM will result in an increase of the reaction flux by 0.5 μM/s (if those are the units that are used in the model). The scaled sensitivities are the ones most discussed in literature, particularly for MCA (but *see* **ref.***12)*.

The next tab shows the *Flux Control Coefficients* (**Fig. 8**). Unlike the elasticities, control coefficients are global properties that depend on the whole system. They quantify the extent of change of the steady-state flux of one reaction when another reaction is made slower or faster. For the MCA formalism it does not matter *how* the reaction is made faster or slower, but in practice changing the enzyme concentration is the most practical solution. Imagine a system in a steady state in which at some point we increase the concentration of one of the enzymes by 1%. After some time a new steady state will be reached, potentially all the concentrations and fluxes in the system will have changed slightly. The relative change of one of the reaction fluxes is the flux control coefficient of this reaction with respect to the reaction with the changed enzyme concentration. Like in the case of the elasticities, all combinations of flux control coefficients are calculated by COPASI and displayed in a table where the column indicate the rate of reaction that is changed and the row indicates the flux of the reaction that has been affected. The fact that the table contains almost no zeros already indicates that these are global properties of system: a change in one reaction changes the steady-state fluxes of all reactions.

In the example of the sucrose accumulation model, one thing that stands out immediately is that two lines ("(v6)" and "(v7)") are identical. This is very common in flux control coefficients and comes from the fact that the two reactions (HexP→UDP + Suc6P and Suc6P→Suc + P) form a chain without any branches in between, so that their steady-state flux is always the same. The original publication of the model discusses the accumulation of sucrose in sugar cane (reaction *v11*) vs. the hydrolysis of sucrose (reaction *v9*), arguing that the sucrose accumulation is most effective when the flux of *v11* is large and that of *v9* is small. Inspection of the last row of the table calculated in COPASI indicates the control that each reaction has over the flux of *v11* and it is interesting that the largest coefficient (0.464) corresponds to *v3*. This means that with an overexpression of hexokinase (the enzyme that catalyzes *v3*) by 10% we expect an increase in the rate of sucrose accumulation of about 4.6%. However, reaction *v3* also has a high control over the flux of *v9*, in fact much larger: 1.558; thus *v3* is not a good candidate for manipulation because while it would stimulate sucrose accumulation it would lead to a much larger increase in the hydrolysis of sucrose actually decreasing the overall efficiency. Rohwer et al. argue that the fructose/glucose transporters (*v1* and *v2*, the first two columns) are better candidates for this purpose since increasing their rates causes a

simultaneous increase in sucrose accumulation and decrease in hydrolysis (as indicated by a negative control coefficient).

From this example of a relatively simple model with only five variables, it becomes evident that there is no intuitive way to reason about the response of the system to perturbations from the network structure alone. In even moderately complicated models it is impossible to predict which enzymes control the fluxes without performing actual sensitivity analysis calculations.

Another issue that may be obvious to some readers from **Fig. 8** is that the values of each row in the table sum up to 1. This is a reflection of the flux control summation theorem *(14)*, which allows us to reason about the system. For example, if the value of a flux control coefficient is known to be 0.3 then one can be sure that also other reactions will control that specific flux (since the coefficients have to add up to 1).

The third tab of the results window contains the *concentration control coefficients*. These are similar to their flux counterparts, and describe how the steady-state concentrations change depending on the changes in specific reaction rates. The main difference between these and the flux control coefficients is that they add up to 0 rather than 1.

An important thing to keep in mind about both the *elasticities* and the *control coefficients* is that they provide information only about small changes to the model. So while you can in many cases reliably predict from the control coefficients what the effect of a 5% increase in the expression of one enzyme will be, it is not possible to predict the effect of a tenfold increase or decrease. This type of information, however, could be obtained from parameter scans, i.e., by direct numerical simulation, however, that would be for a single parameter at a time (*see* **Note 28**).

MCA is a powerful concept, and the way it is implemented in COPASI is numerically robust *(12)*. Basically whenever COPASI is able to find a steady state, the MCA calculations will also provide reliable results.

3.3.2. Generic Sensitivities

Control coefficients are concepts geared toward an interpretation that is dominated by changes in enzyme concentrations (derived from gene expression), as they only measure the effects of changing the overall rate of reactions. It is also interesting to study how other parameters, such as K_m, affect the model behavior. In MCA, these generic sensitivities are known as *response coefficients* and measure the change in a system property effected by any system parameter (*see* **Note 29**), however, these have no known special summation theorems. COPASI can also calculate these generic sensitivities and to access this feature we select *Multiple Task and Sensitivities* in the tree on the left; the corresponding sensitivities window is depicted in **Fig. 9**. Basically these generic sensitivities (response coefficients in the vocabulary of MCA) are for arbitrary

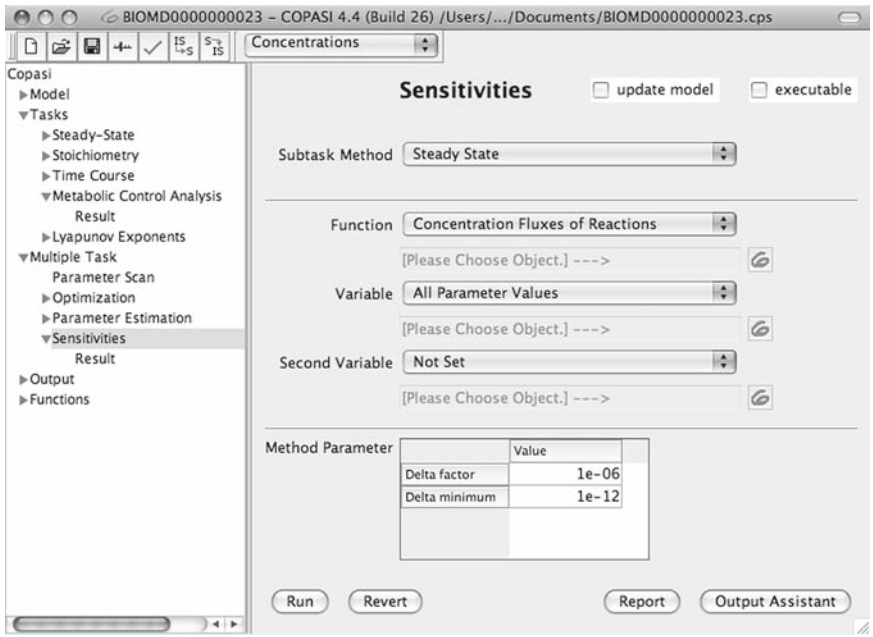


Fig. 9. Generic sensitivities window.

values in the model (*Functions* in **Fig. 9**) with respect to arbitrary parameters (*Variables* in **Fig. 9**) and are calculated numerically using finite differences (i.e., not using a matrix method from elasticities, *see* **Note 30**).

Generic sensitivities can also be calculated for time courses, but we will start with a steady-state example. Make sure that *Subtask method* is set to Steady State. In the *Function* select *Concentration Fluxes of Reactions*, meaning that we want to calculate how the steady-state reactions fluxes (measured in concentration units) are affected by parameter changes. Next the parameters of interest need to be selected in *Variables*. For this example, select *All Parameter Values* that will calculate the sensitivities with respect to all kinetic parameters in the model. After pressing the *Run* button results will appear in its window, and we shall discuss the *Scaled* tab (**Fig. 10**). Once again, the rows correspond to the reactions (as in the flux control coefficients table) and the columns correspond to the kinetic parameters of the model. Since there are usually several parameters for each reaction, this table does not fit entirely on the screen and the scroll bar needs to be used.

A comparison of this table with that of **Fig. 8** reveals that columns 3 and 6 here are identical to columns 1 and 2 of **Fig. 8**. This is expected because the sensitivities of fluxes toward v_{max} parameters can be shown to be the same as flux the control coefficients (unless there are enzyme–enzyme interactions). However, we can see from the sensitivities that some of the inhibition constants (e.g., column 1) also strongly affect the fluxes.

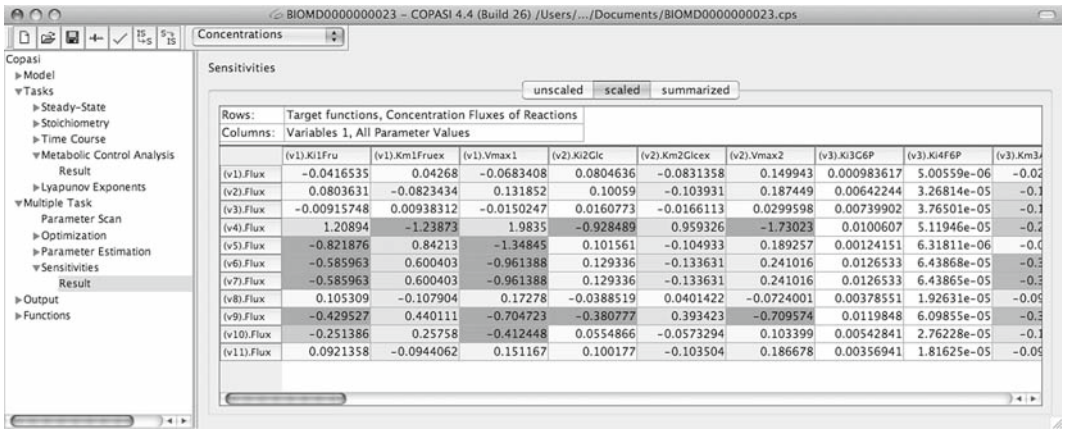


Fig. 10. Results of generic sensitivity analysis.

The generic sensitivities feature allows the calculation of many other kinds of sensitivities as well. For example, the sensitivity of a simulation result with respect to the initial concentrations could also be calculated. It is also possible to calculate second-order sensitivities (*sensitivities of sensitivities, see* **ref.** *21)* which can help determining whether sensitivity analysis results are valid over a larger parameter range.

3.4. Tuning Models with Optimization Methods

Optimization is the search for maximum or minimum values of some function (the *objective function; see* **Note 31**). In biochemical modeling, optimization can be used to find conditions in which the model behaves in some desired way *(13, 22)*. Because biochemical models are composed of nonlinear functions, their variables may have several minima or maxima, thus the problem is usually of *global optimization* where one wants to find the largest of all maxima or the smallest of all minima. Global optimization problems are hard to solve and it is well known that no single algorithm is best for all problems *(23)*. Thus COPASI is equipped with a diversity of optimization algorithms that follow very different strategies (*see* **Table 1**), and in general one should search the best solution with more than one algorithm (and at least one should be a global optimizer).

To demonstrate an application of optimization we will continue analyzing the model of sucrose accumulation in sugar cane *(20)*, which is model 23 in BioModels. Remember that accumulation of sucrose is measured by the steady-state flux of reaction *v11* but there is also a certain amount of sucrose hydrolysis, reaction *v9*, that decreases the efficiency of accumulation. So one important question is what conditions lead to a low proportion of sucrose hydrolysis relative to accumulation. This can be seen as a typical optimization problem, where we are interested in minimizing the ratio of fluxes J_{v9}/J_{v11} – our objective function. In all

Table 1
Optimization algorithms available in COPASI Version 4.4 (Build 26)

Algorithm	Strategy	Type	References
Evolutionary programming	Evolutionary algorithm with adaptive mutation rate without recombination	Global	(*24*)
Evolution strategy (SRES)	Evolutionary algorithm with numerical recombination, selection by stochastic ranking	Global	(*25*)
Genetic algorithm	Evolutionary algorithm with floating-point encoding and tournament selection	Global	(*26*)
Genetic algorithm SR	Variant of Genetic algorithm where selection is by stochastic ranking	Global	(*25, 26*)
Hooke and Jeeves	Direct search algorithm based on pattern search	Local	(*27*)
Levenberg–Marquardt	Gradient-based, adaptive combination of steepest descent and Newton method	Local	(*28–30*)
Nelder–Mead	Direct search method based on geometric heuristics	Local	(*31*)
Particle swarm	Inspired on social insect search strategies; works with population of candidate solutions like evolutionary algorithms	Global	(*32*)
Praxis	Direct search method based on the alternate direction (minimize one dimension at each time)	Local	(*33*)
Random search	Random search with uniform distribution (a shotgun approach)	Global	
Simulated annealing	Monte Carlo method that mimics the process of crystal formation (biased random search with Boltzmann distribution)	Global	(*34*)
Steepest descent	Gradient method based on first derivatives (estimated by finite differences)	Local	
Truncated Newton	Based on Newton method (uses second derivatives)	Local	(*35*)

optimization problems, it must also be specified which parameters of the model are allowed to change in order to meet the objective. In this particular example, let us imagine that we could manipulate the steady-state level of the enzymes of reactions *v1*, *v2*, *v3*, *v4*, and *v5* (e.g., by overexpression or by interfering with the upstream regulatory sequences of their genes). The question then becomes what would be the best combination of the levels of these enzymes to achieve the lowest possible ratio J_{v9}/J_{v11}. The parameters that are allowed to change are then the *Vmax* of the five reactions.

In COPASI, the optimization task is found under *Multiple Tasks* and then *Optimization* in the tree on the left. The application of

optimization to biochemical modeling consists typically of three parts (1) the objective function, (2) the adjustable parameters, and (3) the search algorithm. This is mirrored in COPASI's interface as seen on **Fig. 11**. First, the objective function must be set by entering the mathematical expression J_{v9}/J_{v11}, this is done by selecting the required model entities from a menu that is activated by pressing the small button with the COPASI icon (at the right) (*see* **Note 15**). J_{v9} appears as <(v9).Flux>, then you have to enter the division sign from the keyboard, and finally select J_{v9} which appears as <(v11).Flux> (*see* **Note 32**). If you wanted to instead maximize this expression you should precede it by a minus sign (so that you minimize its symmetric).

Next you need to select the adjustable parameters, i.e., those that are allowed to change. To add one parameter to the list press the *New* button (the one with a blank page) and then the button with the COPASI icon to select the actual parameter. COPASI provides a shortcut to add all parameters: select the first

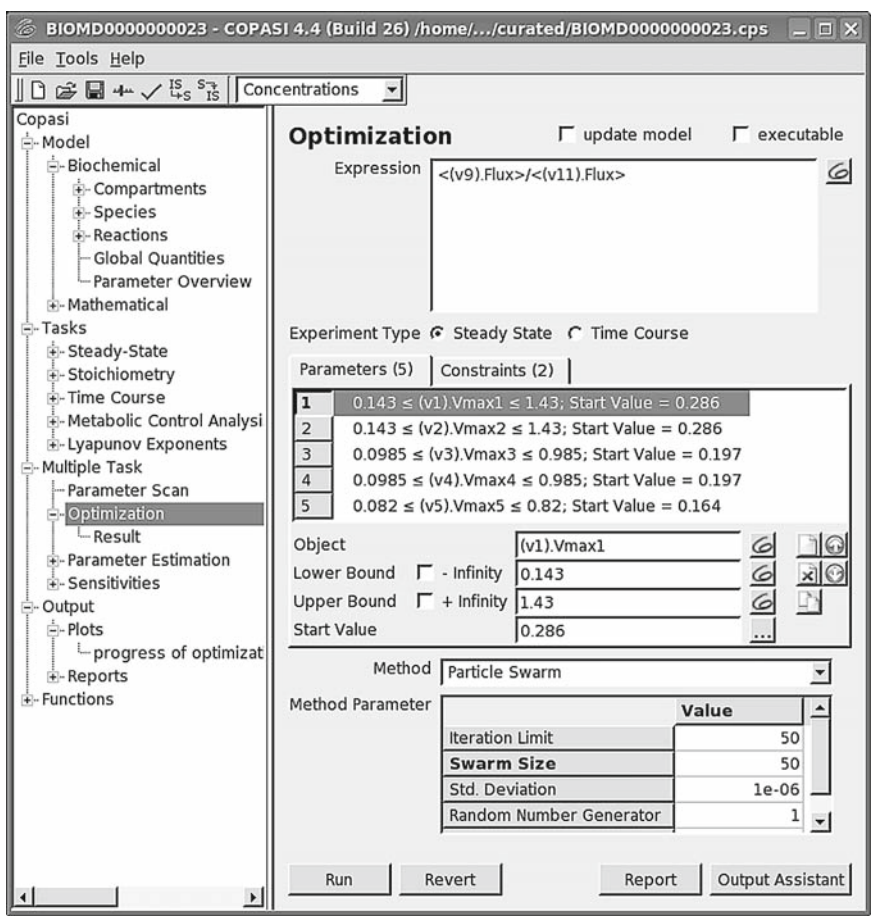


Fig. 11. Optimization window, with an objective function definition at the top, a list of adjustable parameters at the center, and an optimization algorithm at the bottom.

one by expanding *Reactions*, then *Reaction parameters* and then *v1* where you select *Vmax1*; but then rather than just accepting that, expand the other reactions of interest one by one, and while pressing the CTRL key (or the APPLE key on Macs) also select also *Vmax2, Vmax3*, *Vmax4*, and *Vmax5*. When you finish, all five parameters will be listed. You will realize that they are listed inside the interval between –infinity and +infinity, which is quite large indeed, but in general we want tighter limits. Let us say that it is feasible to downregulate the enzyme concentrations to 50% and to overexpress it by 400% in this example. To change all of the boundaries together select all five rows, then remove the check on –inf and type –50% on the box; similarly, remove the check on +inf and type +400% on the box below, after changing the cursor to another field, the limits of each parameter will have changed to the appropriate values (*see* **Note 33**). The start values are by default those that are specified in the model section but they could be changed; yet we shall leave them as they are now (*see* **Note 34**). Please note that if the start value of a parameter is outside the boundaries specified, COPASI will force it to the nearest boundary during the optimization.

Finally, one needs to select the method of optimization desired. For our first attempt let us use the *Truncated Newton* method and press *Run* which will quickly finish. Then move on to the *Results* section (on the left tree, below Optimization). This will show that the objective function value obtained was 0.000593843 (*see* **Note 35**), and below you will see listed the values of *Vmax* for each of the reactions. You will realize that *Vmax1* and *Vmax2* are close to the upper limit specified (indeed as argued in **ref.***20)*, and *Vmax3*, *Vmax4*, and *Vmax5* are near the minimum specified. This means that we would need to overexpress the first two enzymes and downregulate the remaining three. The reader may wonder about this solution, particularly if compared with the network diagram in **Fig. 1** of **ref.** *20*, one clue is given by the concentrations achieved in this solution, which you can inspect if you switch to the tab named *Species* (at the top of the right pane). Both Fructose and Glucose are very highly concentrated (almost 1 molar for Fructose) – it seems that the best way to minimize hydrolysis of sucrose and maximize its storage is to maintain a very high concentration of the products of the hydrolysis. However, this solution may not be achievable in practice due to the high concentrations of the intermediates.

After considering the results of the previous analysis, it becomes interesting to ask the same question but now not allowing the concentrations of Glucose and Fructose go above 100 mM. This is a new set of requirements of the method named *constraints* as they attempt to force the solution to a more restricted domain. To enter constraints, return to the *Optimization* page, and select the tab named *Constraints* in the center of the page.

Then let us add the constraints like we added the adjustable parameters, pressing *New* and then the COPASI icon, and then expand *Species* and *Transient concentrations* and select *Fru* and *Glc*. Set the lower limit to 0 and the upper to 100. Now press *Run* again and inspect the result, which is now a ratio of fluxes of 0.0679853 (about 100× higher than the previous solution), and the concentration of Fru is 89 and Glc is 99.

Now select a different method of optimization, for example *Particle swarm* and set the *Iteration Limit* to 50 (the default of 2,000 is way too long for this problem) and run again. This method takes longer, and you will see a window appear with a progress dialog, which shows the number of function evaluations and the current value of the objective function. At the end it is possible that a window appear with several warnings, if so please *see* **Note 36**. In the end, it will show an objective function value of 0.0584978 or somewhere close to that. Run this a few times and note that the result differs each time; this is because the algorithm is stochastic and it does not always necessarily converge to the same value (*see* **Note 37**). Note that now the concentrations of Fru and Glc are within 0.1% of the upper limit of 100. This shows the great utility of optimization methods in biochemical modeling, and the MCA/sensitivity approach would never be able to answer this constrained problem. With optimization we can solve practical problems with realistic constraints (not just calculations based on infinitesimal changes). It is also very reassuring to realize that the modeler is entirely driving the process *by the definition of objective functions and constrains*, which are a means of directing the computations to solve specific problems. Optimization is an excellent way to explore the space of behavior of complex multidimensional models, such as those of biological systems.

3.5. Parameter Estimation

Biochemical models depend on many parameters, but quite frequently the values of these parameters are unknown and have to be estimated from some data. Parameter estimation is a special case of an optimization problem, in which one attempts to find values for a set of model parameters that minimize the distance between the model behavior (simulation results) and the data. COPASI provides specific parameter estimation functionality that is based on the optimization methods described in **Subheading 3.4**.

COPASI measures the distance between model and data using an expression that is derived from a least-squares approach *(36)*. The objective function used is:

$$O(p) = \sum_i \sum_j \sum_k \omega_{k,i} \left(X_{k,i,j} - \mathrm{Y}_{k,i,j}(\boldsymbol{p}) \right)^2, \qquad (3)$$

where $X_{i,j,k}$ is the experimental value of variable i at measurement j within experiment k and the corresponding simulated data point is given by $\mathrm{Y}_{k,i,j}(\boldsymbol{p})$ where $\boldsymbol{p}$ is the vector of parameter values

used for the simulation. It is important that the data for the different variables be of comparable magnitudes so each group of values for each variable in each experiment is multiplied by a weight $\omega_{k,i}$ (*see* **Note 38**).

3.5.1. Example

To illustrate parameter estimation we shall use the MAP kinase cascade model of Kholodenko *(8)* which is model 10 in BioModels. You will also need some experimental data, and a file (MAPK-data.txt) is provided at http://www.comp-sys-bio.org/tiki-index.php?page=CopasiModels. You must download this file and store it in the same folder where you have put the SBML file with the model that was downloaded from BioModels. The data contained in this file are for "measurements" of the single-phosphorylated form of MAPK and of the phosphorylated MAPKK at various time points (*see* **Note 39**). The problem then consists of adjusting the $V_{\max}$ parameters of a few reactions in order for the model to be as close to the data as possible.

3.5.2. Experimental Data

As implied in the objective function above (**Eq. 3**), COPASI allows fitting the model to multiple experiments simultaneously. The software also allows using steady-state and time-course data, which can even be used together (i.e., some experiments be time courses while others are steady-state observations). The experimental data must be provided in ASCII data files with columns of data delimited by tabs or commas; each column will be mapped to a model entity. Since COPASI knows nothing about your data files, there is a necessary step of creating a mapping between the data columns and model entities. To make this mapping easier we suggest that the data file should include a row of column headings. Additionally, if there are several experiments in a single file, these experiments should be separated by an empty line (allowing COPASI to detect the beginning and end of each experiment's data automatically). Each column of an experiment data file must be classified as one of the types listed in **Table 2**. Even if some columns are not needed, they must be classified as *ignored*. It is important that all columns of type *independent* and *dependent* are actually mapped to the actual model entities they correspond to.

At this point it is best to proceed with the MAPK example and you should examine the structure of the data file with a plain text editor, for example Notepad on Windows (a spreadsheet will also work, as long as you do not overwrite the file). Then import the SBML file in COPASI and select *Multiple Tasks* and *Parameter Estimation*. The complete specification of the data file format is done in a dialog box (**Fig. 12**) that is invoked with the *Experimental Data* button. To add the data file press the *New* button (blank page) that is above the box named *File*. Select the MAPKdata.txt file that you have previously downloaded, and then COPASI will automatically recognize that there is one experiment in this file

Table 2
Classification of data types for mapping experimental data to the model entities

Data type	Meaning
Independent	Independent model items are those which need to be set before the experiment takes place. Possible model elements are initial concentrations but could also be kinetic parameters. Note that in time-course experiments only the first row of independent data columns is used (since it refers to the initial state of the system). Columns of this type must be associated with elements of the model
Time	This column type is only available for time-course experiments and is a special case of an independent model item. Obviously one and only one column of this type may exist in each time course experiment. COPASI will attempt to automatically identify this column if there are column headers but it may fail and in such a case you must set this type for the appropriate column
Dependent	The dependent data are those that were measured in the experiment and are entities in the model that are variables (i.e., determined from the solution of equations rather than set by the modeler). These are the target data that COPASI attempts to match, and are the data specified in the objective function (**Eq. 3**). Columns of this type must be associated with the actual model elements that they correspond to
Ignored	These are columns of data that the user does not want to include in the problem. Columns marked in this way are not taken into account in the parameter fitting task. This is useful to ignore potential irrelevant columns of data files. This setting is also useful to "switch-off" using one data column when desired

that it names *Experiment* (you can change this if you like) and that it goes from line 1 to line 11, including the header in line 1. You must indicate that these data are from a time course, so select the appropriate check box on the *Experiment Type*. The table at the bottom of this dialog box indicates the columns found in the data file for the current experiment, and under the heading *Column Name* it reproduces the titles in the header (line 1 of the file). The first column has been identified as type *Time* because of its title, the remaining two are set to the default type *ignored*. Since these columns contain the measurements of the concentrations of MAPKKK-P and MAPK-P you have to set their type to *dependent*. When you do that, a new dialog appears for you to point to the actual model entity that this column represents, select *Species* and then *Transient Concentrations* and chose the appropriate one (*see* **Note 40**). Repeat the process with the other column; when you finish the dialog box should look the same as **Fig. 12**.

Note that COPASI has already determined values for the weights ($\omega_{k,i}$ in **Eq.**); the brackets indicate that they were calculated rather than set by you. However, you are free to change any weight by editing them and removing the brackets (but note that they should always be positive numbers smaller or equal to 1).

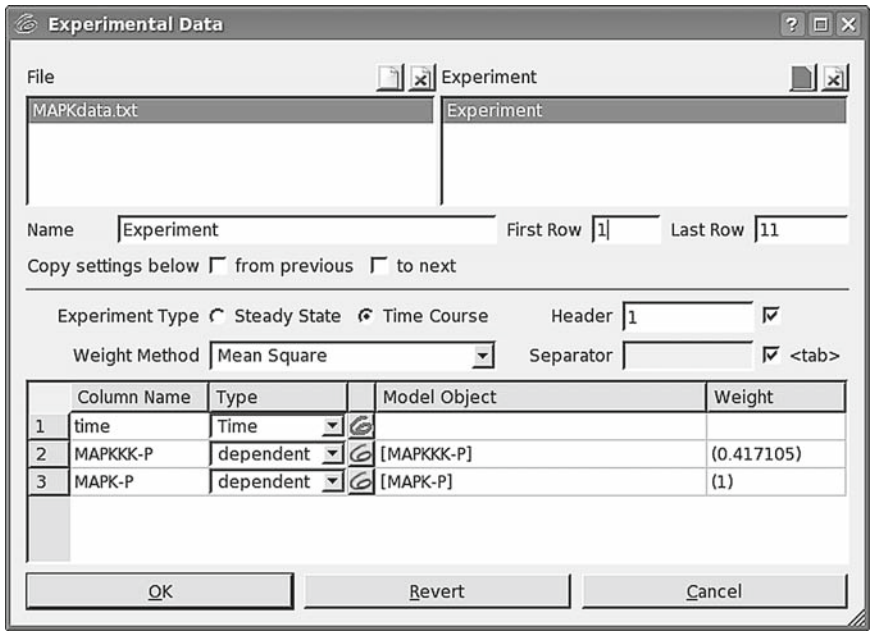


Fig. 12. Experimental data definition window.

It is important to realize that changing the weights affects the ability of the software to perform the fit, and particularly bad choices might entirely prevent success of the fit. COPASI contains three different methods to calculate these weights (*mean*, *mean square*, and *standard deviation*, respectively), as depicted in **Eqs. 4–6**:

$$\omega_{j,k} = 1 / \left| \left\langle X_{j,k} \right\rangle \right|, \tag{4}$$

$$\omega_{j,k} = 1 / \sqrt{\left\langle X_{j,k}^2 \right\rangle}, \tag{5}$$

$$\omega_{j,k} = 1 / \left(\left\langle X_{j,k}^2 \right\rangle - \left\langle X_{j,k} \right\rangle \left\langle X_{j,k} \right\rangle \right). \tag{6}$$

The *mean* and *mean square* methods (**Eqs. 4** and **5**) assure that data columns with small values contribute in the same order of magnitude to the objective function as columns containing large values. The *standard deviation* method (**Eq. 6**) sets larger weight to columns that have little fluctuations.

3.5.3. Estimated Parameters and Constraints

Obviously the exercise of parameter estimation requires one to select the parameters that are to be estimated. Typically these are initial values (concentrations, volumes, etc.) or parameters of the kinetic functions of the reactions (or arbitrary ODE if there are any in the model). The selection of these parameters and their boundaries is specified in exactly the same way as for optimization (*see* **Subheading 3.5.2**).

Sometimes it is necessary to estimate a parameter differently for each experiment, meaning that the software should estimate one value per experiment rather than a single value that best fits *all* experiments (which is the default). For example this is needed when one has executed replicate experiments but where one is not confident that the initial concentration of a chemical species is the same in all experiments. COPASI is able to deal with this, allowing the user to restrict the effect of a parameter to a subset of the experiments listed (obviously this only matters when there are several experiments, but this not the case in the present example). The button labeled *Duplicate for each experiment* is there for this purpose and will multiply the parameters selected when it is pressed to as many new parameters as there are experiments.

It is also possible to define constraints, just like in the optimization task. But beware that adding any arbitrary constraints may well render a problem unsolvable if the constraints cannot be fulfilled. Remember that the main constraints you want for the model is that it fits the data, so the use of constraints in parameter estimation should be taken with care or avoided if possible.

For the present example of the MAPK model, select the reaction limiting rates *V1*, *V2*, *V5*, *V6*, *V9*, and *V10* and set their limits to be –90% and +90% of their original values, in the same way as in the optimization example above.

3.5.4. Fitting the Data

At this point the parameter estimation problem has been completely specified and the actual fitting task can proceed using any of the optimization methods available (*see* **Table 1**). Before running the task it is advisable to define a plot to monitor the progress of the fit and another one to examine results. It is also important to save the file in COPASI format (in case you want to come back to it later, since the SBML file does not contain instructions for the parameter estimation). To create the plots mentioned press the button *Output Assistant* which lists a series of plots and reports that are commonly useful. You can select the plot named *Progress of Fit* and press *Create*, which will generate a plot of the values of **Eq. 3** vs. the number of function evaluations (*see* **Note 41**). The second plot of interest is called *Parameter Estimation Results per Experiment* and it consists of the experimental values of the variables (i.e., contained in the data file) plotted against the values of their corresponding simulated value. The plot also contains the weighted residuals of each data point (i.e., the terms calculated inside the summation in **Eq. 3**).

After defining the plots, save the file, select an optimization method, and press *Run*. For this example select the Levenberg–Marquardt method and run it with the default values. After a short while the method will have finished and you will have plots like those of **Fig. 13**. You should also examine the results by selecting the *Results* page (below *Parameter Estimation* on the left). There you will see the statistics for the sum of squares, though be aware

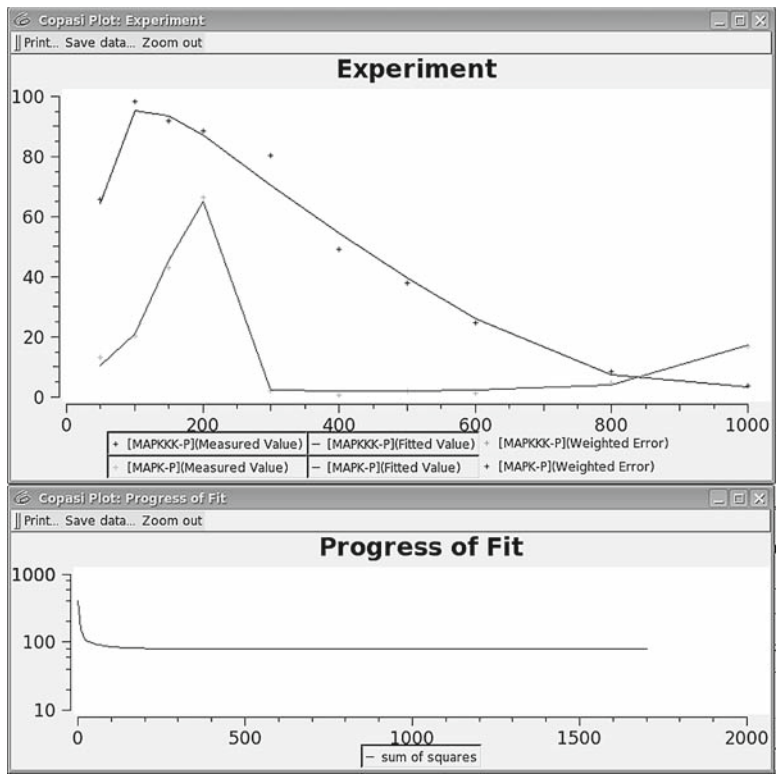


Fig. 13. Results of fitting model parameters to a data set. The plot on the top overlays the experimental data (*crosses*) over the model behavior after fitting (*lines*). The plot at the bottom displays the progress of the sum of squares (**Eq. 3**) as the optimization algorithm progressed (note the logarithmic scale of the *Y*-axis).

that these are problem dependent and you should not compare sums of squares between different problems (not even the same problem with different data sets). More useful are the statistics for the estimated parameters on the second tab, where you will likely see that the coefficients of variation of the estimated parameter values are smaller than 35%, which is very good given the presence of noise in the data. You can also examine the parameter correlation matrix that provides information about dependencies between parameter estimates.

3.6. Stochastic Simulation

Along with the traditional ODE approach, COPASI is also equipped to carry out stochastic simulations based on the theoretical framework derived by Gillespie *(4)*. The *Time Course* task can easily be executed with the algorithm of Gibson and Bruck *(37)* (*see* **Note 42**) and this is as simple as selecting the Gibson–Bruck method from a pull-down menu (**Fig. 14**). This is particularly appealing to those who normally carry out simulations with the ODE approach but sometimes have a need to switch to the stochastic approach. Of course, this also means that COPASI is equally useful for modelers who mostly use the stochastic approach.

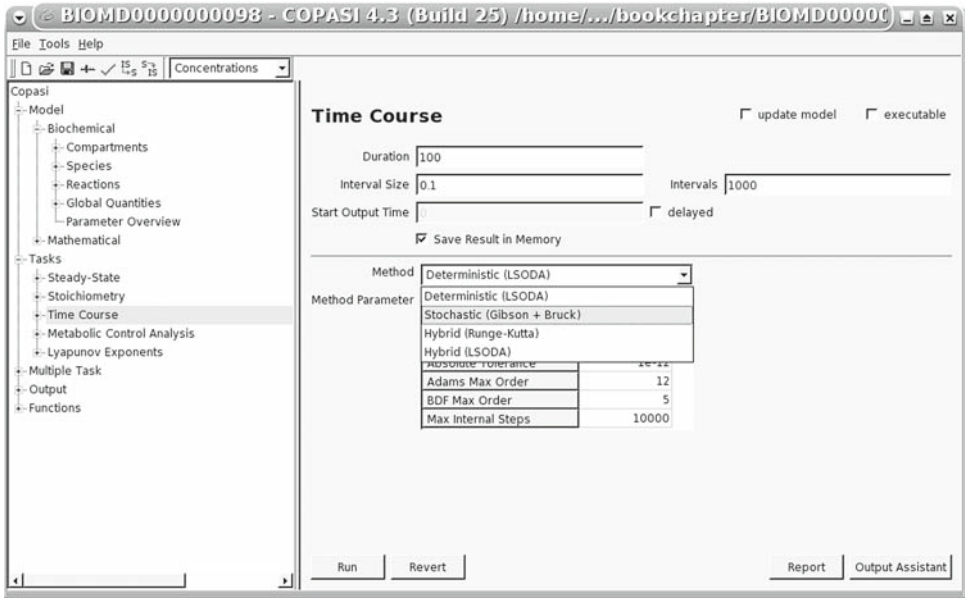


Fig. 14. Switching to a stochastic simulation approach in the *Time Course* window.

Let us consider an example using a model of calcium oscillations by Goldbeter *(38)*, which is model 98 in BioModels. After importing the SBML, go to the *Time Course* task. It is useful to define a trajectory plot of the number of particles against time, which can be done via the *Output Assistant*: chose either the second option (*Particle Numbers, Volumes, and Global Quantity Values*) which will have a scale of numbers of particles, or the first option which will output the corresponding concentrations to the computed particle numbers in the course of the simulation. **Figure 15** shows the outcome of a stochastic simulation for the calcium model.

There are several issues that have to be considered to carry out successful stochastic simulations. The first consideration is that in this approach reversible reactions must be handled as two separate irreversible reactions (the forward and reverse directions). In ODE-based simulations, the forward and backward reaction rates are usually aggregated and thus can cancel each other out (resulting in a null rate); in stochastic simulations each single reaction event has to be considered separately and even if there is no net rate, the actual cycling rate will be explicitly represented. In order to facilitate the conversion of ODE-based models to the stochastic representation, COPASI provides a feature that, at the modeler's request, converts all reversible reactions to the corresponding individual forward and backward reactions (**Fig. 16**). This useful tool adjusts the model automatically – the reaction scheme and the kinetics – and is able to work for a wide range of kinetic rate laws, such as mass action and standard enzymatic kinetics. Nevertheless, there are certain cases when it is not able to dissect rate laws into two separate irreversible kinetic functions. These cases can be very

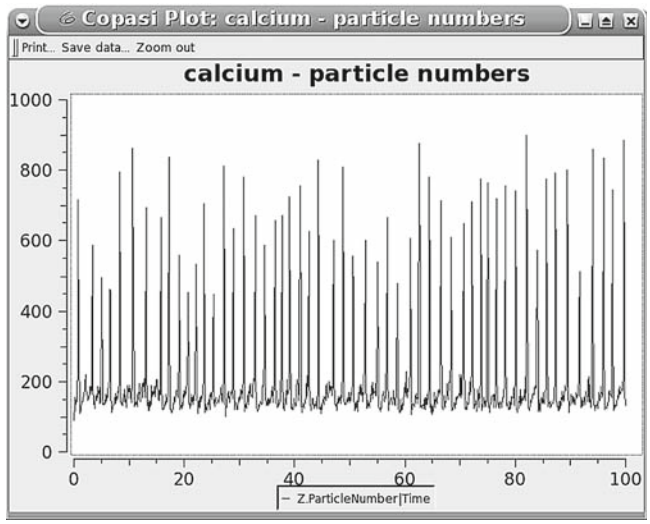


Fig. 15. Trajectory of calcium oscillations using the stochastic simulation algorithm.

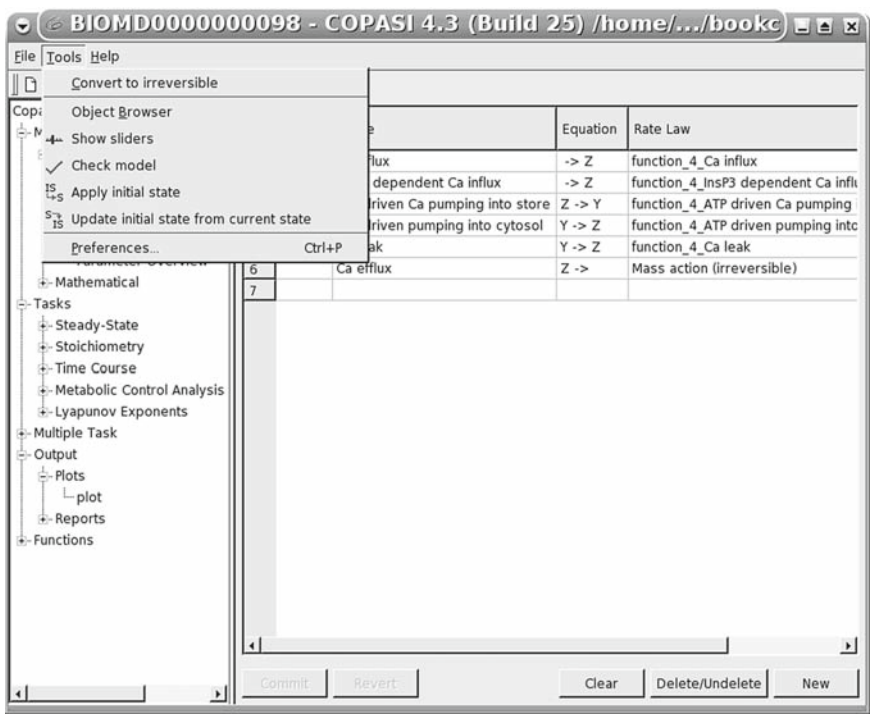


Fig. 16. Menu option to convert a model to be composed only of irreversible reactions.

complex rate laws or rate laws that are actually not appropriate (e.g., an expression that is never negative, thus that is not really reversible). When COPASI cannot automatically convert all reactions, the user will have to adjust the model her/himself.

Often, models are specified without considering the specific volume of the compartment. But for stochastic simulations the volume of the systems is crucial: the volume should not be too big so that the computed particle numbers are not too high and within numerical possibilities of a computer (*see* **Note 43**). This should pose no problem, since it is the purpose of these stochastic simulations to deal with systems that have relatively low particle number. Thus, it is important that the volume of the system be defined in the compartment description in such a way that the particle numbers are not too high.

Another consideration is whether or not the assumptions implied in the rate law of a specific reaction still holds in the presence of low particle numbers. Thus, when stochastically simulating a reaction network which has been described by a set of ODEs all reaction rates have to be converted to a corresponding reaction probability. This is rather simple and straightforward in the case of mass action kinetics *(3)*. However, enzyme kinetic rate laws represent the overall rate of a series of elementary mass action reactions (binding of substrate to enzyme, isomerizations, etc.). An important question is then whether it is justifiable to use such a rate expression in stochastic simulations. Several authors *(39, 40)* have shown that as long as the initial assumptions for the assumed kinetics hold (e.g., excess substrate, fast reversible enzyme–substrate complex formation, etc.), it is indeed justifiable to assume the enzymatic reaction to constitute one single step with a corresponding rate law. The modeler must then ensure that the initial assumptions still hold.

Stochastic simulations are computationally expensive. If a large system is considered which contains some species with high particle numbers and some others with low particle numbers then the use of a hybrid method should be taken into consideration. In COPASI there are currently (version 4.4 Build 26) two hybrid methods implemented. These methods dynamically divide the system into two subsystems: one of them contains reactions with participants that occur in large quantities and is simulated by numeric integrations of ODE; the other one contains reactions that have no participants in large quantities and is stochastically simulated (*see* **Note 44**). In many cases, this approach will speed up the simulation. The two hybrid methods differ only in their numerical integration algorithm – one uses Runge–Kutta, the other uses LSODA.

Since repeated runs of the stochastic simulation will differ considerably, as long as the stochastic influence is noticeable, it is advisable to execute many runs in order to sample a distribution. This can be easily done by using the *Repeat* function of the *Parameter scan* task in COPASI already discussed in a previous section (**Fig. 17**). If a plot of particle numbers over time has been defined, this repeated run will result in multiple time courses being overlaid in a single plot. However, this is not very useful when the dynamics is complex as in the example of calcium oscillations. In these cases, it is best to define a histogram (**Fig. 18**)

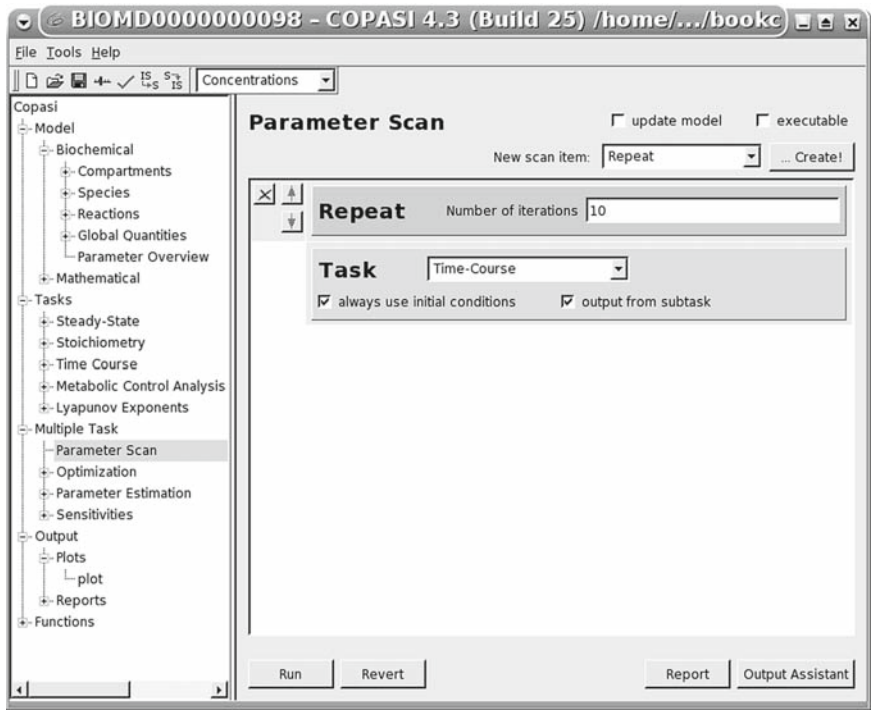


Fig. 17. Using the *Parameter scan* window to repeat the same stochastic trajectory several times.

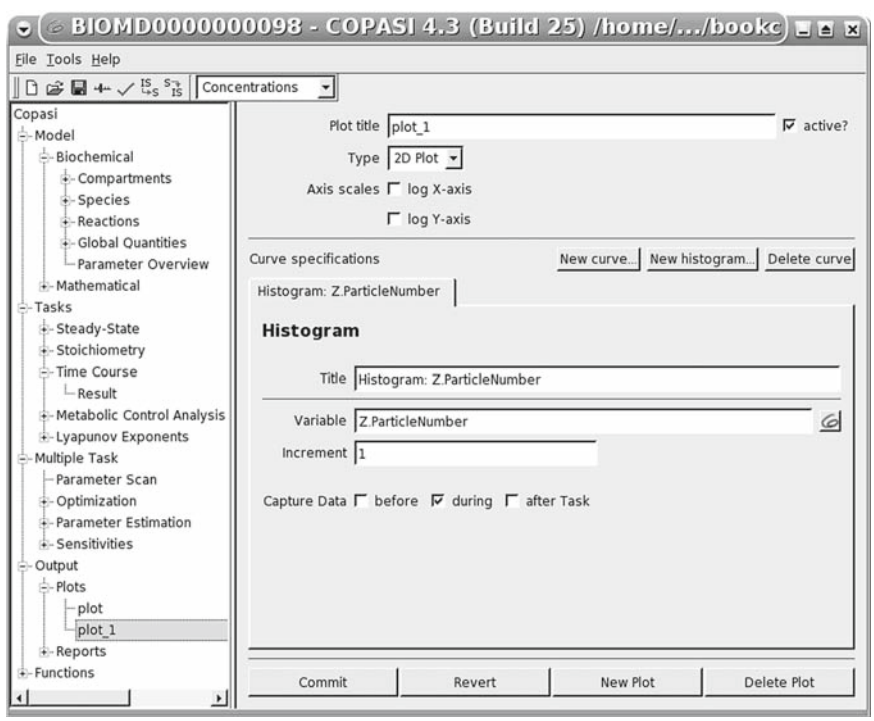


Fig. 18. Defining a histogram plot of a species concentration.

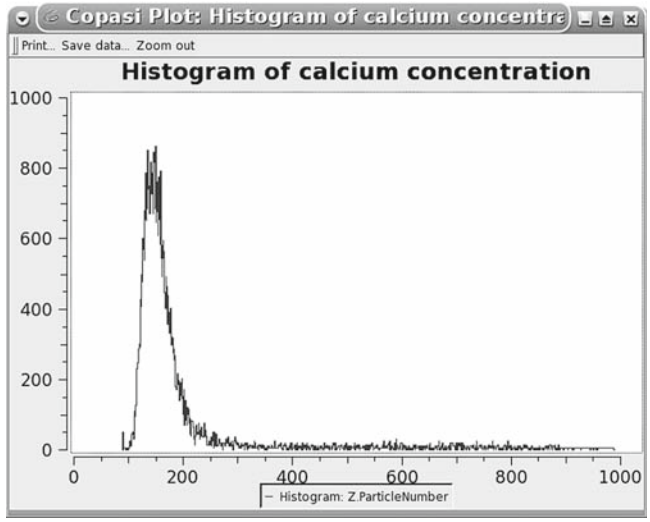


Fig. 19. Histogram of calcium concentration for ten runs of the stochastic simulation algorithm.

to display the cumulative concentration distribution, which is a better way to summarize stochastic simulations (**Fig. 19**).

4. Notes

1. A modifier is a chemical species that affects the rate of reaction but which, unlike substrates or products, is not transformed by the reaction itself. A special case of modifier is the enzyme that catalyzes the reaction, but this class also includes inhibitors and activators.
2. But often not for enzymes, for which the rates usually depend linearly on their concentration – the exception is when there are enzyme–enzyme interactions.
3. It is well known that these equations are often stiff, meaning that they contain very fast and very slow components and this poses a significant numerical problem. Beware of software that does not include ODE integrators (or solvers) that are able to cope with stiff ODEs. Methods such as forward Euler or Runge–Kutta are *not* appropriate when stiffness is present in the equations and can lead to completely spurious solutions because they accumulate truncation error. COPASI uses the LSODA method which is adaptive and is stable under stiff conditions.

4. COPASI can use any of the following three strategies: Newton method, ODE integration forward in time or integration backward in time. If all three are chosen, it first tries the Newton method and if this does not converge, it then integrates in time for a while and then tries the Newton method again – this is repeated ten times, each time integrating even further ahead (10× what was done earlier). If at the end a steady state is not found it will then go back to the original starting point and apply the same strategy but now integrating backward in time. Backward integration, if successful, will find an unstable steady state. The user has control over this strategy by selecting the parameters "Use Newton," "Use Integration," and "Use Back Integration."
5. A commercial license is also available for purchase allowing use of COPASI for applications that are for commercial profit. Go to http://www.copasi.org/commercial for further details.
6. Distribution filenames are in the format: Copasi-XX-YYYYYY.ZZZ where XX is the build number, YYYYYY is a reference to the operating system (*WIN32* for Windows, *Darwin* for OS X, *Linux*, *SunOS* for Solaris, and src for the source code), and ZZZ is the appropriate extension for the type of file, which depends on the operating system.
7. Alternatively you can *browse* the database and find the model that way. However, such a method will become essentially unworkable as the database grows.
8. This means the model is in SBML level 2 version 1; in the future the BioModels database may supply the model in another level/version of SBML so the title of this link may become something like SBML Lx Vy for level x and version y.
9. To download this file you should right click the link and then select an option that allows saving the link to disk (like *save link as…* in Firefox). If you simply click the link your browser will likely show a blank page with some sentences and then a (long) list of parameter names. This is actually part of the model and appears because the browser is trying to interpret the XML encoding as if it was HTML.
10. Additionally there is also an initial value for time, this is only important in the case when some rate equations reference time explicitly (nonautonomous models). In that case, the value of time at the start of the simulation is important and the modeler may need it to be some value different from zero.
11. There is also a selection for the interpretation of rate equations, as there are differences between the ODE and the stochastic approaches. Note that this selection only indicates whether the kinetics used are *meant* for one or the other

approach, not that the approach will be used. In fact, this feature exists so that COPASI can automatically adapt the rate equations to the required approach.

12. There must be one space between each chemical species name, otherwise COPASI will interpret the whole string as one species name. This is because the character "+" is allowed in species names, thus the space is needed to delimit species names from symbols that are not part of the species name.
13. *Substrate* and *product* are obvious; *modifier* is any chemical species that is not transformed by the reaction (inhibitors, activators, and the enzyme if represented explicitly); *volume* is the volume of any compartment; *time* is obvious; and *parameter* is anything else that does not fit any of the other categories.
14. You should not mark as reversible a rate law that can only produce positive values; to be reversible a rate law must be able to take negative values (i.e., flux in the opposite direction). Conversely, an irreversible rate law should not be able to produce negative values.
15. In COPASI, the buttons that are marked with the program's icon are always used to select model entities.
16. *Multiple task* groups a set of computational analyses that require running multiple simulations at each time.
17. This *output from subtask* button would need to be checked if the task was a time course and we wanted the whole time course to be plotted rather than just the final value (although there are also circumstances where that could be desirable thus the choice given to the user).
18. It is best to scan in logarithmic space when the parameter varies by more than one order of magnitude, otherwise most of the samples will lie in the upper order of magnitude.
19. The order of the scan items in the stack is important for the way in which the plot is constructed, but otherwise produces the same results since it generates a regular grid and executes the task at each grid position. The order of the stacks only affects the order in which the grid positions are visited.
20. Obviously a flux mode can only be reversible if *all* reactions that compose it are also reversible.
21. An astute biochemist will realize that there is a carbon and two oxygens missing on the substrate side of this equation, and obviously there should be a CO_2 in that side of the equation. This is missing because the modeler made the decision of not including CO_2 in the model (it should be in the Phosphogluconate dehydrogenase reaction). Since the erythrocyte is not known for fixating CO_2 then the mode must

operate in the reverse direction, i.e., production of PRPP from glucose. In this case, since the mode is reversible it means that one would not know this fact from the stoichiometry alone.

22. A chemical moiety is a set of atoms bound in a fixed structure which are part of molecules, which in a chemical context are referred to as "chemical groups."
23. This means that the dependent species is not calculated from a differential equation, but rather from this mass conservation relation. Thus each mass conservation relation reduces the number of ODE by one.
24. Formalisms of MCA have also been derived for time-dependent states *(41, 42)* but they are rather complicated and some of the coefficients therein are hard to conceptualize, so it is not usually applied.
25. Since COPASI only uses the MCA steady-state formalism, the software first needs to find a steady state before doing the MCA calculations. It is a good idea to investigate the steady state(s) of a model before running MCA, especially regarding the stability of a steady state. While it is technically possible to calculate the MCA for an unstable steady state it is of little practical value.
26. The value of 1% change is here used only for illustration as a "small" change, the coefficients are actually defined only for infinitesimal changes and all the theory is based on that.
27. Since the framework of MCA is based on linearizations and reaction kinetics are generally nonlinear, the values of the elasticities depend on the actual concentrations of the chemical species, so they have to be calculated for specific cases.
28. While it is possible in theory to carry out a large multidimensional scan, the computational time of that exercise would be prohibitive and is beyond simple improvements in computer efficiency (it is an *NP*-complete problem) and thus is essentially impossible for models larger than four or five variables.
29. Control coefficients are actually a special case of response coefficients that have unit elasticity.
30. This in practice consists of finding the steady state, then changing one of the parameter values slightly, and then calculating the new steady state and using ratios to estimate the differentials (the change applied is very small).
31. Maximizing a function is the same as minimizing the symmetric function.
32. There are two types of fluxes in COPASI which only differ by scale: "concentration flux" is expressed in concentration per unit time, while "particle flux" is expressed in numbers

of particles per unit time. In this case you should select "concentration flux." However, what is important is that both be of the same type, since this is a ratio.

33. It is also possible to chose another parameter for the upper or lower bounds, in which case we just need to specify which one with the usual button with the COPASI icon (to the left of the text field). In fact, it is even possible to choose another *estimated* parameter (i.e., one on the list to adjust) as long as that parameter appears in the list before it is used as a boundary value.
34. You may manually override the initial value by highlighting the parameter and then entering a number in the box labeled Start Value or use the tool button labeled as "…" to chose other options, such as random values within the interval.
35. Your numbers may be slightly different due to different precision of different computer architectures, but it should be a number in this range.
36. Possibly there were several warnings of the type "CTrajectoryMethod (12): Internal step limit exceeded," which mean that for some parameter values COPASI could have failed to find a steady state through integration of the ODEs (due to the equations being too stiff). This is not a problem since it may have solved the steady state with the Newton–Raphson method. Even if it indeed failed completely to find a steady state for some parameter combinations, the method will have still converged, as you can judge by the final result. This is one advantage of population-based algorithms: they still work even when the objective function is not continuous (which is what it would look like if the numerical solution could not be obtained).
37. The likelihood that it gets to the same result increases with the length that the algorithm is left running; if you run it with the default 2,000 iterations, it will likely always converge to the same value, but it will run for a much longer time of course.
38. These weights are scaling factors; they are not dependent on the quality of the experimental measurement like a standard deviation.
39. These data were actually created with a slightly modified version of the model where some parameters were changed, a time course was simulated and then noise was added to the values of MAPK-P and MAPKKK-P.
40. This is why it is useful to have column headers because COPASI displays them and you can remember what this column is. This is particularly important if you have a data file with many columns.

41. A function evaluation is the complete calculation needed to simulate the data that needs to match the experimental data. Therefore it consists of calculating all time courses and steady states corresponding to each experiment.
42. Gibson and Bruck's next reaction method *(37)* is a more efficient version of the original Gillespie first reaction method. It achieves better performance by an intelligent use of data structures. For example it stores dependencies between the reactions in dependency graphs and this avoids redundant recalculations of the reaction propensities.
43. Stochastic simulations determine the time interval between reactions, and this time is dependent on the number of particles. If there are too many particles the interval between any two reactions is extremely small, meaning that it would just take too long to simulate any time interval of interest (i.e., at least milliseconds).
44. The division between the subsystems is done with respect to the participating particle numbers and there is a control variable that corresponds to this threshold value which can be adjusted by the user.

Acknowledgments

We thank the users of COPASI whose feedback on the software is crucial for its continued improvement. We also thank *all* developers who have actively contributed to COPASI. COPASI development is supported financially by generous funding from the US National Institute for General Medical Sciences (GM080219), the Virginia Bioinformatics Institute, the Klaus Tschira Foundation, the German Ministry of Education and Research (BMBF), and the UK BBSRC/EPSRC through The Manchester Centre for Integrative Systems Biology.

References

1. Garfinkel, D., Marbach, C. B., and Shapiro, N. Z. (1977) Stiff differential equations. *Ann. Rev. Biophys. Bioeng.* 6, 525–542.
2. Petzold, L. (1983) Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM J. Sci. Stat. Comput.* 4, 136–148.
3. Gillespie, D. T. (1976) A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.* 22, 403–434.
4. Gillespie, D. T. (1977) Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* 81, 2340–2361.

5. Gillespie, D. T. (2007) Stochastic simulation of chemical kinetics. *Ann. Rev. Phys. Chem.* 58, 35–55.
6. Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., and Kummer, U. (2006) COPASI – a COmplex PAthway SImulator. *Bioinformatics* 22, 3067–3074.
7. Le Novere, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J. L., and Hucka, M. (2006) BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res.* 34, D689–D691.
8. Kholodenko, B. N. (2000) Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur. J. Biochem.* 267, 1583–1588.
9. Curien, G., Ravanel, S., and Dumas, R. (2003) A kinetic model of the branch-point between the methionine and threonine biosynthesis pathways in Arabidopsis thaliana. *Eur. J. Biochem.* 270, 4615–4627.
10. Schuster, S. and Hilgetag, C. (1994) On elementary flux modes in biochemical reaction systems at steady state. *J. Biol. Syst.* 2, 165–182.
11. Schuster, S., Fell, D. A., and Dandekar, T. (2000) A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nat. Biotechnol.* 18, 326–332.
12. Reder, C. (1988) Metabolic control theory. A structural approach. *J. Theor. Biol.* 135, 175–201.
13. Holzhütter, H. G. (2004) The principle of flux minimization and its application to estimate stationary fluxes in metabolic networks. *Eur. J. Biochem.* 271, 2905–2922.
14. Kacser, H. and Burns, J. A. (1973) The control of flux. *Symp. Soc. Exp. Biol.* 27, 65–104.
15. Heinrich, R. and Rapoport, T. A. (1974) A linear steady-state treatment of enzymatic chains. General properties, control and effector strength. *Eur. J. Biochem.* 42, 89–95.
16. Fell, D. A. (1992) Metabolic control analysis – a survey of its theoretical and experimental development. *Biochem. J.* 286, 313–330.
17. Heinrich, R. and Schuster, S. (1996) *The Regulation of Cellular Systems.* Chapman & Hall, New York, NY.
18. Fell, D. A. (1996) *Understanding the Control of Metabolism.* Portland Press, London.
19. Cascante, M., Boros, L. G., Comin-Anduix, B., de Atauri, P., Centelles, J. J., and Lee, P. W. (2002) Metabolic control analysis in drug discovery and disease. *Nat. Biotechnol.* 20, 243–249.
20. Rohwer, J. M. and Botha, F. C. (2001) Analysis of sucrose accumulation in the sugar cane culm on the basis of in vitro kinetic data. *Biochem. J.* 358, 437–445.
21. Höfer, T. and Heinrich, R. (1993) A second-order approach to metabolic control analysis. *J. Theor. Biol.* 164, 85–102.
22. Mendes, P. and Kell, D. B. (1998) Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics* 14, 869–883.
23. Wolpert, D. H. and Macready, W. G. (1997) No free lunch theorems for optimization. *IEEE Trans. Evolut. Comput.* 1, 67–82.
24. Fogel, D. B., Fogel, L. J., and Atmar, J. W. (1992) Meta-evolutionary programming, in *25th Asilomar Conference on Signals, Systems & Computers* (Chen, R. R., ed.). IEEE Computer Society, Asilomar, CA, pp. 540–545.
25. Runarsson, T. and Yao, X. (2000) Stochastic ranking for constrained evolutionary optimization. *IEEE Trans. Evolut. Comput.* 4, 284–294.
26. Michalewicz, Z. (1994) *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, Berlin.
27. Hooke, R. and Jeeves, T. A. (1961) "Direct search" solution of numerical and statistical problems. *J. ACM* 8, 212–229.
28. Levenberg, K. (1944) A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math.* 2, 164–168.
29. Goldfeld, S. M., Quant, R. E., and Trotter, H. F. (1966) Maximisation by quadratic hill-climbing. *Econometrica* 34, 541–555.
30. Marquardt, D. W. (1963) An algorithm for least squares estimation of nonlinear parameters. *SIAM J.* 11, 431–441.
31. Nelder, J. A. and Mead, R. (1965) A simplex method for function minimization. *Comput. J.* 7, 308–313.
32. Kennedy, J. and Eberhart, R. (1995) Particle swarm optimization. *Proc. IEEE Int. Conf. Neural Netw.* 4, 1942–1948.
33. Brent, P. R. (1973) A new algorithm for minimizing a function of several variables without calculating derivatives, in *Algorithms for Minimization Without Derivatives* (Brent, P. R., ed.). Prentice-Hall, Englewood Cliffs, NJ, pp. 117–167.
34. Corana, A., Marchesi, M., Martini, C., and Ridella, S. (1987) Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Trans. Math. Softw.* 13, 262–280.

35. Nash, S. G. (1984) Newton-type minimization via the Lanczos method. *SIAM J. Numer. Anal.* 21, 770–788.
36. Johnson, M. L. and Faunt, L. M. (1992) Parameter estimation by least-squares methods. *Methods Enzymol.* 210, 1–37.
37. Gibson, M. A. and Bruck, J. (2000) Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A* 104, 1876–1889.
38. Goldbeter, A., Dupont, G., and Berridge, M. J. (1990) Minimal model for signal-induced Ca2+ oscillations and for their frequency encoding through protein phosphorylation. *Proc. Natl Acad. Sci. USA* 87, 1461–1465.
39. Rao, C. V. and Arkin, A. P. (2003) Stochastic chemical kinetics and the quasi-steady-state assumption: application to the Gillespie algorithm. *J. Chem. Phys.* 118, 4999–5010.
40. Cao, Y., Gillespie, D., and Petzold, L. (2005) Multiscale stochastic simulation algorithm with stochastic partial equilibrium assumption for chemically reacting systems. *J. Comput. Phys.* 206, 395–411.
41. Acerenza, L., Sauro, H. M., and Kacser, H. (1989) Control analysis of time dependent metabolic systems. *J. Theor. Biol.* 137, 423–444.
42. Ingalls, B. P. and Sauro, H. M. (2003) Sensitivity analysis of stoichiometric networks: an extension of metabolic control analysis to non-steady state trajectories. *J. Theor. Biol.* 222, 23–36.