Software Tools for Systems Biology

Herbert M. Sauro and Frank T. Bergmann

Department of Bioengineering,
University of Washington, Seattle, 98195-5061, WA

December 5, 2012

Contents

Abstract

Probably one of the most characteristic features of a living system is its continual propensity to change as it juggles the demands of survival with the need to replicate. Internally these changes are manifest by changes in metabolite, protein and gene activities. Such changes have become increasingly obvious to experimentalists with the advent of quantitative high-throughput technologies. Given the complexity of cellular networks it is no surprise that researchers have also turned to computer simulation and the development of more theory based approaches to augment and assist in the development of a fully quantitative understanding of cellular dynamics. In this chapter we highlight some of the software tools and emerging standards for representing, simulating and analyzing cellular networks. The chapter will not be concerned with tools for managing high throughput data or analyzing genome scale data using bioinformatic approaches.

Abbreviations:

SBML: Systems Biology Markup Language
CellML: Cellular Markup Language
SBW: Systems Biology Workbench
MCA: Metabolic Control Analysis
FBA: Flux Balance Analysis
VCell: Virtual Cell
API: Application Programming Interface
XML: Extensible Markup Language
SOSLib: SBML ODE Solver Library
ODE: Ordinary Differential Equation
PDE: Partial Differential Equation

1 Introduction

The use of simulation and theoretical studies in cellular and molecular biology has a long, if uneven, tradition spanning almost seventy years (Wright, 1934). With the recent advent of high-throughput technologies, interest in using more quantitative approaches to enhance our understanding of cellular dynamics has increased dramatically (Tyson et al., 2001; Neves and Iyengar, 2002; Geva-Zatorsky et al., 2006; Kholodenko, 2006). As part of this trend there has been a significant rise in the availability of computer software to help us model, simulate and analyze dynamic models of cellular processes. In this chapter we will briefly survey the available simulation software for systems biology. In addition we will also briefly discuss model databases, model exchange standards, ontologies and computational approaches that are relevant to the study of dynamics in biochemical networks. We will not be concerned with software tools for managing high throughput data or analyzing genome scale data using bioinformatic approaches.

Simulating biochemical networks has a long history dating back to at least the 1940s (Chance, 1943). The earliest simulations relied on building either mechanical or electrical analogs of biochemical networks. It was only in the late 1950s, with the advent of digital computers and the development of specialized software tools (Garfinkel, 1968) that the ability to simulate biochemical networks became more widely available. In the intervening years up to 1980, a handful of other software applications were developed (Burns, 1969, 1971; Park and Wright, 1973) to help the small community of modelers. In more recent years, particularly since the early 1990s, there has been a significant increase in interest in modeling biochemical processes and a wide range of tools is now available to the budding systems biologist. Many tools have been developed by practicing scientists and are therefore available for free (often in the form of open source) while others are commercial such as SimBiology from MathWorks (`http://www.mathworks.com/`, 2006).

Given the recent explosion in interest in quantitative biology (Klipp et al., 2005; Alon, 2006), the number and variety of tools and approaches has likewise risen. In order to make the coverage more manageable we will restrict our discussion to software that is aimed at modeling systems based on differential equations or stochastic chemical kinetics. We will not therefore be concerned software for modeling Boolean models, agent based models

or models that use process calculi. For a discussion of these models and related software the reader is referred to the recent review by Fisher and Henzinger (Henzinger, 2007) and Degenring et al. (2004).

Moreover we will largely confine ourselves to dynamic models which means that we will not discuss at any length software that is designed to study the structural properties of networks other than in a few notable cases. In particular software used to build and compute flux balance models (Fell and Small, 1986; Kauffman et al., 2003; Lee et al., 2006; Palsson, 2007) or software for building isotopomer models (**?**Schwender, 2008) will not be covered in detail even though such analyses are crucial to the metabolic engineering community.

Finally we will also not discuss tools that specialize in simulating spatial models or neurophysiology systems. For the former the interested reader should consult the following references for more information on this topic (Tomita et al., 1999; Ander et al., 2004; Broderick et al., 2005; Hattne et al., 2005; Coggan et al., 2005; Lemerle et al., 2005; Sanford et al., 2006). For simulating neural systems the reader can consult (Hines, 1993; Carnevale and Hines, 2006; Bower and Beeman, 1998; Bhalla, 2002). Although this might seem like a long list of exclusions, we are still left with over one hundred tools that focus on simulating non-spatial, deterministic or stochastic based models.

1.1 The Problem

The software discussed in this chapter is concerned with analyzing two particular types of problem. One is the classical problem of solving systems of differential equations that describe the deterministic evolution of molecular species concentrations in time. These systems are generally written in the form:

$$\frac{\boldsymbol{dx}}{\boldsymbol{dt}} = \boldsymbol{N}\ \boldsymbol{v}(\boldsymbol{x}(\boldsymbol{p}), \boldsymbol{p}) \tag{1}$$

where $\boldsymbol{x}$ is the vector of species concentrations, $\boldsymbol{N}$ the stoichiometry matrix, $\boldsymbol{v}$ the rate vector and $\boldsymbol{p}$ the vector of parameters. Often, the system of equations is overdetermined due to the presence of conserved moieties Reich and Selkov (1981) and the equation is then reexpressed in the form:

$$\boldsymbol{x_d} \ = \ \boldsymbol{L_0 x_i} + \boldsymbol{T}$$

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{N_R}\ \boldsymbol{v}(\boldsymbol{x}(\boldsymbol{p}), \boldsymbol{p}) \tag{2}$$

where $\boldsymbol{x_i}$ is the vector of independent species, $\boldsymbol{x_d}$ is the vector of dependent species, $\boldsymbol{T}$ is the vector of conserved mass totals, $\boldsymbol{N_R}$ is the reduced stoichiometry matrix and $\boldsymbol{L_0}$ is part of the Link matrix (Reder, 1988; Sauro and Ingalls, 2004). If the system (1) is overdetermined then it is highly advisable to eliminate the redundant equations. There are a number of reasons for this, the most obvious being the reduction in the size of the model that accompanies the elimination. A more important reason is that the Jacobian matrix, a fundamental quantity in dynamical theory, is singular (Vallabhajosyula et al., 2006) in an over-determined system. This in turn makes many important numerical methods unstable. Many simulators fail to carry out this important stage.

The second problem of interest is the realization of the master equation (Wilkinson, 2006). This equation describes the time evolution of the concentration probabilities. However, since we need one time evolution equation for every particle state in the system, the system of equations becomes unwieldy very rapidly as the system size increases. Instead modelers employ the exact stochastic simulation (SSA, Gillespie (1976)) algorithm (and it variants) developed by Gillespie. Equation (3) summarizes this approach where j indicates which reaction will fire and at what time in the future (τ).

$$p(\tau, j \mid \boldsymbol{x}, t) = a_j(\boldsymbol{x})\ e^{-a_o(\boldsymbol{x})\ \tau} \tag{3}$$

t is the current time, τ is the time to the next reaction, $\boldsymbol{x}$ is the species vector, a_j is the propensity function of reaction j and a_o the sum of all propensity functions. On its own, the SSA only generates a single trajectory or realization. In order to glean statistics on the model dynamics many repeated runs must be made using this equation. As with the deterministic case, the system is overdetermined and it is possible to reduce the work load by solving explicitly for the independent species only.

A third approach to modeling, which is intermediate between (1) and (3), is to employ the Langevin equation, that is a stochastic differential equation (Cyganowski et al., 2001). A stochastic differential equation can be constructed by appending a noise term to the deterministic formulation (1), however very few tools currently provide facilities to solve stochastic differen-

tial equation which is unfortunate (Adalsteinsson et al., 2004; Chickarmane et al., 2007).

Further information on modeling intracellular noise can be found in the excellent review by Rao et al. (2002).

1.2 The Need for Software

The first question one might ask is why develop specialized software to model biochemical networks? Given the availability of both generic commercial and freely available tools for numerical analysis one might ask if there is such a need. There are probably at least two reasons why researchers develop their own specialized tools for modeling biochemical systems. The first reason is that specialized tools reduce the errors that occur when transcribing a reaction scheme (that is a biological representation) into the mathematical formalism ready for simulation. Deriving the math equation by hand is often a frequent source of error (especially in published papers) particularly in large models. The second important reason is that developing software offers an opportunity to codify and develop new numerical algorithms or new theoretical approaches that are specific to problems found in systems biology. Such examples include incorporating metabolic control analysis, MCA, (Kacser and Burns, 1973; Savageau, 1972) into software (which is the reason why one of us developed simulation software in the first place) or developing more efficient methods for network analysis (Vallabhajosyula et al., 2006). Although it is possible to carry many analyzes in tools such as Matlab, Mathematica or SciLab, the process tends to be more tedious and error prone and certain analyses are unavailable to the user without applying considerable effort.

1.3 Functionality

Given the long history of provision of software in modeling biochemical networks, we believe there is a minimum level of functionality that any newly developed or existing tool should have. Foremost is the ability to convert a biological description, that is a set of biochemical reactions and corresponding rate laws, into a set of differential equations. The input format for the model can be in the form of SBML (Systems Biology Markup Language, (Hucka

et al., 2003) or CellML (Hedley et al., 2001) - (Cell Markup Language), see section 4 for details - or it can be in the form of a human readable text format which can be more easily edited and, if necessary, converted later into SBML or CellML for export. Tools such as the Systems Biology Workbench (SBW, see section 2.7 for details) provide facilities for converting SBML to simple human readable text files which can be edited and loaded into the simulation tool.

Once a model has been converted into a set of appropriate differential equations, the solutions can be easily generated from widely available numerical integration routines. The US Department of Energy in particular has developed a very useful series of sophisticated libraries, including SUNDIALS (Hindmarsh et al., 2005) and ODEPACK (Hindmarsh, 1983). These libraries are quite straight forward to incorporate into software and can be used to solve even some of the most complex problems.

Together with generating a solution, some means to save simulation results is obviously needed and in a format that can be loaded by widely available tools such as Gnuplot (Williams et al., 1998) or Excel (`microsoft.com`). The final minimum requirement is some way to modify parameters and initial conditions to allow a user to repeat runs although such functionality could simply involve editing the model in a text file and rerunning the model.

What was just described is essentially the functionality of some of the earliest simulators dating back to the 1950s. Clearly, any simulator developed today should exceed these requirements.

Beyond the minimum requirements there are a whole host of other desirable capabilities, these include: model reduction by way of conservation laws, steady state determination, sensitivity analysis (via MCA is possible), frequency analysis, bifurcation discovery and analysis, structural analysis, and finally optimization and parameter fitting. In addition to these, other capabilities such as access to a model database (particularly the BioModels Database Le Novère (2006). see section 5), the availability of a variety of model editors (visual, text and GUI) and finally some way to generate camera ready copy of network diagrams for publication purposes would also be desirable.

2 A Brief Survey of Commonly Used Tools

In the last ten years or so, numerous software applications have been written to support modeling of biochemical processes. The variety of software is huge even within the restricted scope of this discussion. Almost every computer language has been used at one time or another to build simulation software. Some applications work on single operating platforms, others work across multiple platforms; some are trivial to install, while others require considerable persuasive efforts to make them work. There is also a wide variety of user interfaces such as graphical user interfaces incorporating menus, buttons, lists etc., network design interfaces that allow users to draw a pathway on the screen and text based scripts that enable users to control and define a computer model very precisely. Finally, application capabilities vary enormously, some attempt to be broad offering many different capabilities, others choose to specialize in one particular area. Some of the tools are extensible, that is new functionality can be added by a user either through scripting languages or by writing small plugin applications. Most have some form of reference documentation, including tutorials to help users get started.

A visit to the software guide at the SBML.org web site will reveal over one hundred different tools for modeling or working with biochemical networks. The majority of these have been developed in the last eight years or so. However, although there appears to be a large number of offerings, many are not actively developed and have become orphaned software. In this chapter we will only focus on those tools which are actively developed and have an excellent chance of being supported in the future.

We will focus initially one some tools we use ourselves. We sometimes use these tools because our software may lack certain features we need. More often we also use these tools to compare simulation results.

Before continuing we would also like to mention three legacy simulators which have influenced the development of most modern applications, these include Gepasi (Mendes, 1993), METAMOD (Hofmeyr and van der Merwe, 1986) and SCAMP (Sauro and Fell, 1991; Sauro, 1993). All three were developed in the 1980s and in some cases continue to be used.

All the tools we will describe can import SBML, and all, with the exception of JSim, can export SBML, however JSim can also import CellML. Both SBML

and CellML are established model exchange standards which we will discuss in more detail in a later section. All the tools we will describe are currently maintained and have an active user community. Most of the tools run on multiple platforms (Windows, Linux, Mac) and most, with the exception of one or two, are open source though under varying licenses.

Given that we are also developers of simulation tools it would be unfair for us to provide a detailed comparison of the different applications, instead we refer interested readers to recent independent reviews by Alves (Alves et al., 2006), Manninen (Manninen et al., 2006) and Vacheva (Vacheva and Eils, 2006) and two comparison web sites, one maintained by the VCell group at `http://ntcnp.org/twiki/bin/view/VCell/SBMLFeatureToolMatrix` and another by our own group at `http://www.sys-bio.org/sbwWiki/compare`. The feature matrix at the VCell site is a user contributed table that lists the capabilities of each tool. In contrast, the comparison site at sys-bio.org does not attempt to make statements about particular capabilities instead it runs each registered tool through every model in the BioModels Database (over 170 models) and compares the results. From such results, a developer or user can judge the capabilities of a particular software tool without any bias from ourselves.

Although not on the list, we should mention the use of Matlab as an excellent numerical and data analysis application in systems biology. Matlab is an application which we ourselves have frequently used for complex data analysis. MathWorks also supplies a specialized tool box called SimBiology which offers many useful capabilities.

All these tools that we will now describe are capable of basic simulation, that is solving differential equations or simulating a stochastic model, what makes them stand out is what they add in addition to this basic functionality.

2.1 COPASI

Pedro Mendes wrote one of the earliest PC simulators which he called Gepasi (Mendes, 1993). COPASI (Hoops et al., 2006) is essentially a rewrite of Gepasi, that comes in two versions: a graphical user interface and a command line version. The command line version is designed for batch jobs where a graphical user interface is unnecessary and where runs can be carried out

without human supervision. COPASI uses its own file format to store models, however like all the tools discussed here, it can import and export SBML. One of its undoubted strengths is optimization and parameter fitting which it inherited from its predecessor. It has an unique ability to optimize on a great variety of different criteria including metrics such as eigenvalues, transient times etc. This makes COPASI extremely flexible for optimization problems. Installation is very simple and entails using a one-click installer. Although the source code to COPASI is available and can be freely used for research purposes in academia, owing to the way in which the development of COPASI was funded there are restrictions on commercial use.

The graphical user interface is based on a menu/dialog approach, much like its immediate predecessor, Gepasi. COPASI has capabilities to simulate deterministic as well as stochastic models and includes a wide range of analyzes. It correctly takes into account conservation laws and has very good support for metabolic control analysis amongst other things. COPASI is without doubt one of the better simulators available. Although the user interface is graphical, it does, due to its particular design, require some effort to master but with the availability of the COPASI the source code, there is the opportunity to provide alternative user interfaces.. Finally there is a version that has an SBW interface (Systems Biology Workbench) which allows SBW enabled tools access to COPASI's functionality (currently available at `sys-bio.org`).

2.2 Jarnac

Jarnac (Sauro, 2000) is a rapid prototyping script based tool that was developed as a successor to SCAMP (Sauro and Fell, 1991). It is distributed as part of the Systems Biology Workbench which makes installation a on-click affair. Jarnac was developed in the late 1990s before the advent of portable GUI toolkits which explains why it only runs under Windows although it runs well under Wine (Windows emulator) thus permitting it to run under Linux. Visually, Jarnac has two main windows, a console where commands can be issued and results returned and an editor where control scripts and models can be developed. The application also has a plotting window which is used when graphing commands are issued.

Jarnac implements two languages, a biochemical descriptive language which

allows users to enter models as reaction schemes (similar to a SCAMP script) and a second language, the model control language which is a full featured scripting language that can be used to manipulate and analyze a model. The main advantage of Jarnac over other tools is that models can be very rapidly built and modeled. From our experience with using many simulation tools over the years, Jarnac probably offers the fastest development time for model building of any tool. Models can also be imported or exported as SBML. Like COPASI and PySCeS, Jarnac offers many analysis capabilities including extensive support for metabolic control analysis, structural analysis of networks and stochastic simulation. It has no explicit support for parameter fitting but this is easily remedied by transferring a model directly to a tool such as COPASI via SBW.

2.3 JSim

JSim (Raymond et al., 2003) is a Java-based simulation system for building quantitative numeric models and analyzing them with respect to experimental reference data. JSim can either be used from a web browser or downloaded to a desktop machine. JSim's primary focus is physiology and biomedicine, however its computational abilities are quite general and applicable to a wide range of scientific domains. JSim models may intermix ODEs, PDEs, implicit equations, integrals, summations, discrete events and procedural code as appropriate. One of JSim's strengths which makes it standout from other simulators is its careful attention to dimensional analysis. Users can thus specify units for all terms in a model and JSim will then carry out an automatic check to ensure that the specified units are consistent with the model. In addition JSim can automatically insert conversion factors for compatible physical units. JSim is one of the few modeling programs that can import SBML as well as CellML formats.

JSim's modeling language is mathematically orientated and thus may not be suited to all users, however the authors also define a more biologically orientated language called BCL which allows users to specify models in a more biological orientated manner.

2.4 MathSBML

MathSBML (Shapiro et al., 2004) is an open-source package for working with SBML models in Mathematica. It provides facilities for reading SBML models, converting them to systems of ordinary differential equations for simulation and plotting in Mathematica, and translating the models to other formats. MathSBML is one of the few simulators to support differential/algebraic equations that may present themselves in models. For Mathematica users, MathSBML is a good choice.

2.5 PySCeS

PySCeS (Olivier et al., 2005) is probably the best Python based simulator currently available. Although users interact with PySCeS via Python scripts, many of the underlying numerical capabilities are provided by well established C/C++ or FORTRAN based numerical libraries. It was written by a team well seasoned in biochemical modeling and as a result the software is reliable and comprehensive. As a portable free scripting tool for systems biology, PySCeS is currently one of the best. The only major element missing is parameter optimization but such functionality can be added through additional Python scripting. As with COPASI and Jarnac, PySCeS correctly handles conservation laws and has very extensive support for metabolic control analysis including extensive structural analysis capabilities. PySCeS is also one of the few tools to have some limited support for bifurcation analysis. Installation on Windows is very straight forward with a single click install. Although the current version of PySCeS is mainly concerned with deterministic simulations, there are plans to add stochastic capabilities in collaboration with the University of Newcastle, UK. There are a number of other Python based simulators including sloppyCell (Myers et al., 2007), Scrumpy (Poolman, 2006), ByoDyn (`http://cbbl.imim.es:8080/ByoDyn`) and PyLESS (`http://sysbio.molgen.mpg.de/pyless/`) which specializes in model reduction. PySCeS however is the most mature and comprehensive of these.

2.6 SBToolbox^2

This is a very extensive Matlab tool box developed by Henning Schmidt (Schmidt and Jirstrand, 2006). The tool box has a wide range of capabilities. One of it unique features is the ability to interact with XPPAUT (Ermentrout, 2002) by generating the XPP files and launching XPPAUT. For a Matlab user the SBToolbox^2 is a particularly useful tool to have available.

2.7 Systems Biology Workbench

The Systems Biology Workbench (Sauro et al., 2003; Bergmann et al., 2006), unlike other tools described in this chapter is not a simulation resource itself, it is instead an open source framework for connecting heterogeneous software applications. SBW is made up of two kinds of components:

- Modules: These are the applications that a user would use. There is available a broad collection of model editing, model simulation and model analysis tools.
- Framework: The software framework that allows developers to cross programming language boundaries and connect application modules to form new applications.

Modules in SBW can be written in a wide variety of languages including C/C++, Java, .NET, Delphi, Matlab, Python and Perl. Each module will expose application programming interfaces that allow other application modules access to their functionality.

One of the core modules in SBW is the simulation module, roadRunner. roadRunner is implemented using the C# programming language and exploits the ability of .NET to generate model code on the fly and subsequent compilation and optimization by the .NET framework. This approach leads to the generation of very fast simulation times. Numerical analysis is provided by the CVODE (Hindmarsh et al., 2005) integrator and the NLEQ library (`http://www.zib.de/Numerik/numsoft/NewtonLib/`) for steady state computations. RoadRunner is available on the Windows platforms through the .NET framework and on POSIX systems through the

mono project (`http://www.mono-project.com`). roadRunner relies on several SBW modules for the reading of SBML and for conservation analysis, resulting in a smaller set of ODEs to solve but also the ability to generate non-singular Jacobian matrices which is required for steady state evaluation and other analyses. In addition to simulation and steady state evaluation, roadRunner also incorporates a full set of routines to compute sensitivities (i.e. Metabolic Control Analysis), frequency analysis, various structural metrics from the stoichiometry matrix and a basic continuation algorithm based on arc length (Kubicek, 1976; Kubicek and Marek, 1983). RoadRunner is also the backend simulator for the online simulation tools at `sys-bio.org`.

2.8 VCell

The Virtual Cell (VCell, Schaff et al., 1997; Moraru et al., 2002; Slepchenko et al., 2003) is a client/server based tool that specializes in three-dimensional cell simulations. It is unique in that it provides a framework for not only modeling biochemical networks but also electrophysiological, and transport phenomena while at the same time considering the subcellular localization of the molecules that take part in them. This localization can take the form of a three-dimensional arbitrarily shaped cell, where the molecular species might be heterogeneously distributed. In addition, the geometry of the cell, including the locations and shapes of subcellular organelles, can be imported directly from microscope images. VCell is written in Java but has numerical analysis carried out by C/C++ and FORTRAN coded software to improve performance. Currently, modeling must be carried out using the client/server model which necessitates a connection to the internet. In addition models are generally stored on the VCell remote server rather than the clients desktop. This operating model is not always agreeable to users and as a result the VCell team are reorganizing the software so that it can also be run as a standalone application on a researchers machine. Recently the VCell team has incorporated the BioNetGen (Blinov et al., 2004) network generator which allows models to be specified in a rule based manner. VCell is also one of the few tools that can both import and export SBML and CellML. This feature could in principle be used to translate between SBML and CellML models.

2.9 Visual Editors

Tools that allow users to draw pathways on a screen and turn them into simulatable models seem to be fairly rare. We confine our attention here to tools that are specifically designed to assist in simulation rather than pathway annotation. Examples of the later include the Edinburgh Pathway Editor (Sorokin et al., 2006), cytoscape (Shannon et al., 2003), BioUML (Kolpakov, 2004) and BioTapestry (**?**) but many others exist. We mention the later tools specifically because they have some simulation capability but their strengths lie elsewhere. A very complete review of non-simulation based network viewers can be found in Suderman and Hallett (2007).

Probably one of the first visual editors to be written for simulation was a Mac based tool called KineCyte (Cook and Atkins, 1997; Cook et al., 2001) by Dan Cook and around the same time JDesigner (Bergmann et al., 2006) was developed by Sauro for the Windows platform. Other tools include applications such as SimWiz (Rost and Kummer, 2004), CADLive (Kurata et al., 2007) and Cellware (Dhar et al., 2004), although unfortunately Cellware does not appear to be under development any longer. The three main visual design tools still supported and geared towards simulation are JDesigner, ProMoT (Ginkel et al., 2003) and CellDesigner (Kitano et al., 2005).

2.9.1 JDesigner

JDesigner (Sauro et al., 2003; Bergmann et al., 2006) is open source (BSD licence) and runs under Windows. It requires SBW to enable simulation capabilities. With SBW, models can be constructed using JDesigner and seamlessly transferred to other tools such as COPASI or any other SBW enabled tools. Unlike CellDesigner, JDesigner takes a minimal approach to representing networks. CellDesigner has twelve node types (plus variants) and six different transition types. JDesigner in contrast has one node type, one generic reaction type and two regulatory types. All networks can be constructed from these four basic types. This minimal approach reflects the fact that the underlying models are the same regardless of the molecules or reaction types. Thus protein network models and metabolic models are indistinguishable at the mathematical level. Although JDesigner has only a limited number of types, nodes, reactions and membranes can be modified

visually to change colors, shapes etc. Moreover, nodes can be decorated with covalent sites and multimeric structures. JDesigner uses fully adjustable multi-bezier arcs to generate reactions and regulatory arcs and has a variety of export formats that allow camera-ready copy to be generated for publications. Models are stored in native SBML with specific open access annotations to store the visual information.

2.9.2 CellDesigner

CellDesigner (Kitano et al., 2005), developed by Akira Funahashi in the Kitano group, is a popular close-source but cross-platform Java based tool. Like JDesigner it also needs external support to enable simulation, which can include SBW or SOSLib ((Machné et al., 2006), see section 2.13.2). One of the main strengths of CellDesigner is its visually appealing graphical representation which has made CellDesigner a popular tool. Its other strength, though still under development, is the availability of a plug-in architecture which in principle will allow other third-party programmers to develop add-ons to the tool (Andreas et al., 2008). One potential drawback of CellDesigner is that the format that the tool uses to store the models is undocumented so that only CellDesigner is capable of reading and writing the models. Only one other tool, the visual layout tool in SBW can read CellDesigner models and that was accomplished by reverse engineering the CellDesigner format. With the advent of the Systems Biology Graphical Notation it is hoped that tool writers will move towards a community agreed and open format for representing biochemical networks (See later section for more details).

2.9.3 Commercial Visual Design Tools

There are also some commercial pathway editors including Cell Illustrator (`http://www.cellillustrator.com`), SimBiology from MathWorks and an interesting contribution from Gene Network Sciences (`http://www.gnsbiotech.com/`) which utilized a visual language called DCL (Diagrammatic cell language) which unfortunately was patented and thereby sealed its future and in any case seems no longer to be available.

2.10 CellML based Applications

Although SBML is by and large one of the more popular exchange formats, CellML is also an important standard particularly among cell physiologists, where models combine biochemical network models with transport and electrophysiological behavior. For this reason most applications developed with CellML import and export capabilities tend to be geared towards physiological modeling and often do no to support SBML. Hybrid applications such as VCell and JSim try to support both formats with varying degrees of success since the formats are not generally compatible.

We have already mentioned JSim as an application that can import CellML models. In addition to JSim there are a number of prominent simulators in this category. The most important of these include, Andre's CellML Tools, COR (Garny et al., 2003), PCEnv and VCell (Slepchenko et al., 2003). All these tools can import *and* export CellML.

Andre's CellML tools include a series of utilities to browse CellML models and carry out basic simulations. COR, another important CellML based tool, is a Windows application developed by Alan Garny at Oxford University in the UK and was probably the first simulator to support CellML. COR uses an interesting scripting language to describe a model which means users do not need to know the intricacies of CellML. The language incorporates facilities to allow a user to specify units, much like the facilities found in JSim. COR also supports high speed simulation capabilities through runtime compilation. Installation is very simple and uses a one-click installer.

PCEnv is a tool that is being development by the CellML team at Auckland. It incorporates a basic simulator permitting users to generate time course simulations, change parameters and generate new models. Model generation is GUI based where a user builds a tree representing the various components of the model. In addition to the GUI interface there is also a JavaScript interface which offers some control over the output and simulation. Installation is very simple and uses a one-click installer. Tools such as COR and PCEnv are under very active development and it is hoped that in the future they will offer many of the facilities that are currently available in other systems biology simulation tools.

2.11 Other Tools of Interest

We list here some other tools that we have not mentioned but are notable for one reason or another. Not all of these tools appear to be actively developed but may experience continued developed in the future. Most if not all of these tools support SBML import and export. All these tools are worth looking at, some implement particular functions very well. In the descriptions only a brief hint is given concerning their capabilities and the reader is strongly urged to investigate them further if some particular functionality is of interest.

BioNetGen: BioNetGen (Blinov et al., 2004) is a tool for automatically generating mathematical models of biological systems from user-specified rules for biomolecular interactions, particularly signaling pathways. Rules are specified in a script based language from which the reaction scheme is generated and subsequently the mathematical model. There are not many tools that operate on a rule based approach (*cf.* little b, `http://www.littleb.org/`) but all of them take a particular and fairly radical view of biology that assumes that all interactions are potentially significant of which there may be many thousands in a given model depending on the specified rules.

Cellerator/xCellerator: This is part of the SIGMOID project (Cheng et al., 2005) at Caltech and Irvine although its origins are earlier. It is a Mathematica based tool designed to facilitate biological modeling via automated equation generation. Two notable features of this tool include the ability to embed arrow-based chemical notation directly into Mathematica scripts and support for multicellular models. Model analysis is fairly basic, confined largely to generating time course simulations. The other Mathematica tool, MathSBML (Shapiro et al., 2004) has much more extensive analysis capabilities.

E-Cell: The E-Cell (Tomita et al., 1999) simulation environment is an object-oriented software suite for modeling, simulation, and analysis of large scale complex systems such as biological cells. It has been in development since 1996 and can, in the latest version, be accessed via Python. For some reason E-Cell has never gained the popularity that perhaps is should have. Early versions suffered from a difficult to use user interface

which most likely slowed its acceptance by the community and model interchange support has been weak which also hampered its acceptance by the wider community.

JigCell: JigCell (Vass et al., 2004) is a modeling tool developed by the Tyson group who are well known for their cell cycle models (Tyson et al., 2002). Many tools are deigned by programmers rather than modelers but JigCell is one of those few tools that was written specifically by a well established modeling group to fit models to data and test the models against numerous experimental results. The system works in a similar way to regression testing in software design where the computational model is repeatedly tested as the model evolves or new experimental data is made available. This enables the generation of robust and more reliable models. As part of this goal, JigCell also supports parameter estimation and the ability to maintain an ensemble of models. The unique functionality provided by JigCell reflects the fact that the authors are modelers rather than just software developers.

ProMoT: ProMoT (Ginkel et al., 2003) stands for process modeling tool and originated from the process engineering community but has since evolved to encompass biochemical modeling. Currently it only runs well on Linux but plans have been made to make a Windows version. It is notable for its ability to model differential-algebraic equations and is one of the few tools to provide visual support for dealing with model libraries, modules and hierarchical models.

Oscill8: Oscill8 (`http://oscill8.sf.net/`) is a bifurcation analysis tool developed by the Emery Conrad at the Tyson group. It runs well on the Windows platform and takes XPP scripts as input. Oscill8 represents one of the few bifurcation packages with a modern GUI interface. SBW supplies a special SBML to XPP translator that is designed to work with Oscill8. Although the XPPAUT site also provides an SBML to XPP translator this is unsuitable for many biochemical models because it doesn't take into consideration conservation laws in the models which renders the Jacobian singular and thereby unsuitable for the AUTO numerical algorithms.

PottersWheel: This is a very comprehensive parameter fitting tool (Mai-

wald and Timmer, 2008) that works well with the SBToolBox^2 (Schmidt and Jirstrand, 2006) but can also be used alone. In a number of cases it is better than COPASI's capabilities particularly in the area of generating nonlinear confidence limits on parameter fits and analyzing the resulting fit. The experimental data input formats are also very flexible. The tool provides out of the box a number of optimization algorithms including Genetic and simulated annealing approaches. For a Matlab user interested in doing parameter fitting properly, this tool is a serious contender especially since the built-in optimization methods in Matlab itself are not sufficiently robust for fitting biochemical models. A further advantage of this tool is that is comes with copious documentation. PottersWheel combined with the SBToolbox^2 is a powerful mix.

R Packages: There is currently limited support for dynamic modeling in R, but this is changing. Given the growing interest in using R in biological research it would not be surprising if further work is done in this area. Currently the two tools that are of interest include SBMLR package by Tomas Radivoyevitch and a more modern version by Michael Lawrence called R-SBML. Modeling is however in all cases limited to simple time course simulations. Both packages use the ODEPACK library for integrating the ODEs which arguably is superior to CVODE from SUNDIALS due to ODEPACK's ability to switch integration modes during the integration.

SBML-PET: SBML-PET (Zi and Klipp, 2006) is a Systems Biology Markup Language (SBML) based Parameter Estimation Tool. Of interest is that the tool supports not only event handling in SBML but also uses the stochastic ranking evolution strategy (SRES, Runarsson and Yao (2005)) for fitting, which is arguably one of the best fitting strategies to use (Moles et al., 2003).

SBMLToolbox: This is a Matlab tool box developed by the Caltech SBML team (Keating et al., 2006). Its primary function is to load and save SBML in to Matlab data structures. It uses the built-in Matlab functions to carry out simulations. Its importance lies in it's careful adherence to the SBML specifications. Other Matlab related tools add functionality to this toolbox.

SloppyCell: This is a Python based tool developed at Cornell (Myers et al., 2007), it has some significant capabilities in parameter fitting related to estimating confidence limits in fitted models. For a Python based tool that supports parameter fitting this is probably the most mature although the supplied optimization methods are limited in scope (Levenberg-Marquardt (Levenberg, 1944; Marquardt, 1963) and Nelder and Mead (Nelder and Mead, 1965)). It does support ensemble model fitting which is very useful and is probably it's main Raison d'etre. It does not however compare with PySCeS in other areas. If additional fitting methods, for example, a stochastic ranking evolution strategy (SRES, Runarsson and Yao (2005)), were included, a combination of sloppyCell and PySCeS would be extremely powerful.

XPPAUT: Although XPP (Ermentrout, 2002) is not a tool that was written specifically for the systems biology community, its great utility is its extensive support for bifurcation analysis, a largely neglected field in the software development community particulary in systems biology. XPP gains its computational capabilities from a well established software library called AUTO (Doedel, 1981). Although it is possible to use AUTO on its own, XPP adds a user interface to make it easier to use. Although very capable, XPP's user interface is however not very modern by current standards and does take a little getting used to. Running XPP on Windows is awkward and XPP is really meant to be run on the Linux/Unix platform. For an easier tool, but perhaps with more limited capability, Oscill8 is a good choice for the Windows user. For Python users there is an interesting tool called PyDSTool (Myers et al., 2007) (inspired by the original DSTool (Back et al., 1992)) although users must derive the differential equations themselves and installation is by no means trivial. However it may be possible to modify the tool for the systems biology community.

2.12 Commercial Tools

There is a small but growing list of commercial titles available in the market. The most prominent of these include Berkeley Madonna, Jacobian, SimBiology from MathWorks and Terranode. We should also mention SimPheny from Genomatica even though it is not a dynamic simulation package but

because it is a well established tool for flux balance analysis. For a non-commercial flux balance tool see (Klamt et al., 2007) - note that this tool requires a commercial copy of Matlab to operate. All these tools can export SBML with the exception of Berkeley Madonna although COPASI can export Berkeley Madonna files. The one real issue with commercial tools is the lack of transparency with regard to the numerical algorithms they employ and the impossibility of changing the methods when necessary. MathWorks to its credit has documentation on the methods they use but the code itself is not visible to the researcher. This is a significant problem for a research based community, less so perhaps for a cooperate funded project. It is of course a difficult issue to reconcile for commercial vendors since the underlying numerical methods are often the source of value in the product.

2.13 Dedicated Libraries

Up to now we have focused on describing enduser applications where a user would download an installer and in a very short time be ready to start defining a model and running a simulation. As mentioned previously there are a huge number of such tools available, though sadly many are started but left unfinished. In addition many tools reinvent basic functionality over and over again wasting considerable human intellect and time. In this section we would like to describe a small but growing list of software libraries dedicated to dynamic simulation in systems biology and targeted at developers of new software. These libraries are cross-platform and most significantly cross-language solutions to common computational and data management problems in systems biology. They enable developers to create new applications from existing well tested and documented solutions and enable developers to focus on novel functionality rather than spending most of their effort on reinventing well understood themes. Furthermore, from a historical perspective, good, well tested software libraries tend to survive the turmoils of funding cycles and researcher turnover whereas enduser applications tend not to.

2.13.1 libSBML

LibSBML (Bornstein et al., 2008) is a very useful software library provided by the SBML team that can be used to read and write SBML. The success of SBML over competing standards can be ascribed in part to libSBML. The software library is based around a C/C++ core, with wrappers provided for many programming languages. Furthermore the library is available for Windows and POSIX operating systems where a C based API is exposed and thus can be linked to virtually any other computer language with relative ease. With an abundance of documentation and examples, software developers can readily use libSBML for their SBML support.

By using libSBML a developer can focus on how to interpret computational models rather than concerning themselves with the mechanics of reading and writing SBML. At of the time of writing, libSBML has released version 3. This version has new features including the ability to validate the model, such as unit consistency checking, or checks on whether the model assignment expressions are over determined. LibSBML also provides support for MIRIAM compatible annotations (Le Novère et al., 2005). LibSBML is very well developed library that sets a high standard for the development of other software libraries in the systems biology community.

2.13.2 SOSLib

The SBML ODE Solver Library (Machné et al., 2006) or SOSlib, is a simulation library that can be linked against any application using the C programming language. At the core, SOSlib uses the SUNDIALS suite (Hindmarsh et al., 2005) and libSBML. Like most SBML capable simulators it does not deal with models using delayed differential equations and algebraic equations. On the other hand SOSlib is one of the few simulators that can handle events reasonably well. Supporting SBML events has proved to be a difficult undertaking for many developers because of the numerical difficulties that they generate. Very few ODE integrators support event detection and as a result only very few simulators (such as SOSLib and roadRunner) are capable of correctly handling them.

SOSlib can be run in two modes, in the interpreted mode, SOSlib will use the abstract syntax trees of libSBML to evaluate the kinetic laws and

other equations. The use of the abstract syntax tree also permits SOSLib to invoke symbolic evaluation when computing the elements of the Jacobian matrix (a techniques also employed by SCAMP (Sauro and Fell, 1991; Sauro, 1993)). The second mode is where the right hand side of every ODE is compiled, using either GCC on POSIX environments or tcc (`http://fabrice.bellard.free.fr/tcc/`) on Windows. This allows SOSLib to generate fast simulation times. SOSlib however has one significant drawback which is its inability to remove dependent variables from the model. If this deficiency could be corrected, SOSLib would be a much more useful library to the community.

2.13.3 libStructural

The libStructural library is a software library for conservation and structural analysis of stoichiometric networks and is written in C/C++ for cross platform portability (`http://sys-bio.org/libStructural`).

One of the initial steps in a simulation (deterministic or stochastic) is to consider the stoichiometric structure of a model. This is carried out for a number of reasons, first it allows model reduction to take place and permits faster simulation times, secondly it allows users to study the structural characteristics of the model, such as flux and moiety conservation. The flux constraints can be further used to establish linear programming models for flux balance analysis (Fell and Small, 1986; Palsson, 2007) and also as an aid to constructing the isotopomers dynamic models used in isotopic measurements of fluxes (**?**Schwender, 2008). The library provides over sixty API calls that gives information on various stoichiometric properties such as the kernel, link, dependent and independent flux and species matrices. In addition the library exposes some useful functionality from the LAPACK library (Anderson et al., 1999). The library can also be built against libSBML so that models in standard SBML can be analyzed otherwise the library accepts raw stoichiometry matrices. The library is fully documented and tested against the very large models from the Palsson group (Price et al., 2003) and over twenty smaller test models.

3 Dedicated Tools for Stochastic Simulation

Stochastic simulation has received much attention in the last ten to twenty years owing to the realization that continuous models are not always an appropriate description of what is, at the molecular level, a fundamentally discrete process. The development of practical algorithms for stochastic simulated owes its origin to a series of classical papers published by Gillespie in the 1970s. Gillespie's approach yielded an algorithm that was extremely straight forward to implement in software but was largely unknown until the early 1990s in biology. Two of the first papers to utilize Gillespie's approach in biology are Kraus (Kraus et al., 1992) and Moniz-Barreto (Moniz-Barreto and Fell, 1993). However it was probably the paper by Arkin et al. (Arkin et al., 1998) that convinced many of the importance of stochastic dynamics in biology. In the intervening years there has been an explosion in interest in stochastic dynamics, attracting researchers from many diverse fields (Curti et al., 2003; Wilkinson, 2006; Ullah and Wolkenhauer, 2007).

Together with modeling efforts there have been developments in software particulary in relation to making the basic Gillespie method more efficient. An excellent and highly readable review of the current state of numerical methods is given by (Gillespie, 2007). Here we will briefly review some of the software available for stochastic simulation, we will not consider hybrid methods which combine discrete with continuous modeling since the methods are still under development (see (Salis et al., 2006) for details), however a brief review is given in (Ullah and Wolkenhauer, 2007). We will also not consider here spatial-stochastic modeling tools such as MCell (Stiles and Bartol, 2001).

Unlike deterministic models where a single run provides all the information on the trajectory, the Gillespie method only generates one of a very large number of possible realizations. Ideally stochastic simulators should support ensemble simulations so that statistics can be collected on the behavior of the model. Many simulators do not provide for this facility and presumably expect the user to manually make the necessary repeats (which ideally should run into tens of thousands of runs). Only a few simulators support ensemble runs (e.g. COPASI) but only one software tool (Vallabhajosyula and Sauro, 2007) computes a wide range of statistics for analyzing stochastic models.

As with differential equation based models, there is a test suite available that allows developers of stochastic simulators to evaluate their efforts (See section 6). The test suite was developed and is maintained by the Wilkinson group at Newcastle, England (Evans et al., 2008). An example of how the test suite can be used is given in Vallabhajosyula and Sauro (2007).

As for software, there are a number of tools that support stochastic simulation, many of these include the tools we have described in previous sections, for example, COPASI and Jarnac. SBW also incorporates a number of stochastic solvers, including a high speed solver written using .NET. Other more specific tools that focus exclusively on stochastic simulation include BioNetS (Adalsteinsson et al., 2004) which is a Mac based tool (a version with a Java frontend also exists for Windows and Linux) that incorporates a number of stochastic related solvers including Gillespie and a capability to solve Langevin equations. This later ability is rare and makes BioNetS a very useful tool for this kind of work.

Two other well known applications, include Dizzy and StochSim. Dizzy (Ramsey et al., 2005) is a Java based application that was developed by Stephen Ramsey at the Institute of Systems Biology in Seattle and incorporates a number of stochastic methods. It is capable of reading SBML level 1 and has an SBW programmatic interface. StochSim (Le Novère and Shimizu, 2001) was developed in the 1990's at Cambridge University, UK by Carl Firth in Bray's group and subsequently developed further by Le Novère and Shimizu. It uses a slightly different method to the Gillespie approach which according to the authors scales better when a system contains molecules that can exist in multiple states. In addition later versions have provision for spatial simulation in two dimensions. One significant difference is that StochSim uses a fixed time step whose size will determine the accuracy of the algorithm. In contrast, the Gillespie SSA method uses an exact approach where the step size of computed as part of the algorithm itself. Subsequent variants of SSA such as tau-leaping are also however approximate and for detailed statistical analysis of stochastic run, such approximate methods should probably be avoided.

Other examples of stochastic simulators include Cyto-Sim (Sedwards and Mazza, 2007) from the Microsoft Trento group written in Java and ESS from the University of Tennessee (Cox et al., 2003). Cyto-Sim incorporates

an interesting and easy to learn script language for describing biochemical networks. ESS and its descendants are notable for implementing probably the worlds fastest stochastic simulation software.

Finally, there currently exists one software library, StochKit, that is dedicated to implementing a wide variety of Gillespie based approaches (Li et al., 2007) and is being being developed by the Petzold group in Santa Barbara, USA. StochKit is a C++ based library and therefore can in principle only be linked easily to other C++ applications. A mundane but more useful C based API (Application Programming Interface) would make the library much more portable to other software languages. In addition StochKit currently lacks many of the advanced analysis methods that a stochastic solver would need for practical use. Finally, support for Windows appears to be very limited which makes the library essentially a Linux based tool. StochKit is a promising start but has someway to go before it reaches the usability and portability of libraries such as SUNDIALS (Hindmarsh et al., 2005) or ODEPACK Hindmarsh (1983). The Tennessee group that developed ESS (Cox et al., 2003) is developing a very high performance C based solution that will incorporate many advanced features such as support for discrete events and ensemble simulations. In addition there are dedicated hardware solutions to stochastic simulation which involve the use of field programmable gate arrays and other technology to build what could potentially be very high speed simulation engines (Peterson and Lancaster, 2002; McCollum et al., 2003; Yoshimi et al., 2004).

4 Standards

With the surge in the number of incompatible simulation tools since the year 2000, it was realized by at least two communities that some form of standardization for model exchange was necessary. The two standards that emerged, include CellML and SBML. CellML is primarily a notation for representing biochemical models in a strict mathematical form, as a result it is in principle completely general. SBML on the other hand uses a biologically inspired notation to represent networks from which a mathematical model can be generated. Each has its strengths and weaknesses, SBML has a simpler structure compared to CellML, as a result there is more software support for

SBML. Most software tools at the present time support import and export of SBML. Both standards have very active communities with intra cellular models being primarily the domain of SBML and physiological models for CellML.

4.1 CellML

CellML (Hedley et al., 2001; Lloyd et al., 2004) represents cellular models using a mathematical description. In addition, CellML represents entities using a component based approach where relationships between components are represented by connections. The literal translation of the mathematics however goes much further, in fact the representation that CellML uses is very reminiscent of the way an engineer might wire up an analog computer to solve the equations (though without specifying the integrators). As a result CellML is very general and in principal could probably represent any system that has a mathematical description. CellML is also very precise in that every item in a model is defined explicitly. However, the generality and explicit nature of CellML also results in increased complexity especially for software developers.

Another key aspect of CellML is its provision for metadata support. The metadata can be used to provide a context for a model, such as the author name, when it was created and what additional documents are available for it's description. CellML uses standard XML based metadata containers such as RDF and within RDF the Dublin Core. CellML metadata, such as BioPAX (`http://www.biopax.org`, Luciano and Stevens (2007)), is how biological information can be introduced into a CellML model.

The CellML team has amassed a very large suite (hundreds) of models(`www.cellml.org/models/`, Lloyd et al. (2008)) which provides many real examples of CellML syntax. This is an extremely useful resource for the community.

4.2 SBML

Whereas CellML attempts to be highly comprehensive, SBML was designed to meet the immediate needs of the modeling community and is therefore more focused on a particular problem set. One result of this is that the stan-

dard is simpler compared to CellML although more recent revisions add new functionality so that the difference in complexity between CellML and SBML is becoming less significant. Like CellML, SBML is based on XML, however unlike CellML, it takes a different approach to representing cellular models. The way SBML represents models, closely maps the way existing modeling packages represent models. Whereas CellML represents models mathematically, SBML represent models as a list of chemical transformations. Since every process in a biological cell can ultimately be broken down into one or more chemical transformations this was a natural representation to use. However SBML does not have generalized elements such as components and connections, SBML employs specific elements to represent spatial compartments, molecular species and chemical transformations. In addition to these, SBML also has provision for rules which can be used to represent constraints, derived values and general math which for one reason or another cannot be transformed into a chemical scheme.

SBML, like any standard, evolves with time (Finney and Hucka, 2003). Major revisions of the standard are captured in levels, while minor modifications and clarifications are captured in versions. An example of a major change within the standard would be the use of MathML in level two of SBML, whereas level one encoded infix strings to denote reaction rates and rules. A minor change on the other hand would for example be the introduction of semantic annotations that can be added to SBML level two version three, whereas this was not possible in a supported fashion in earlier versions (see section 4.3.3). As of this writing, SBML level three is still in development. With level three the standard will develop in an extensible manner. This means there will be a set of core features that must be supported around which additional features, such as spatial modeling, can be included.

There has been little effort to develop methods to inter-convert between SBML and CellML, however Schilstra from the UK developed a tool called CellML2SBML (Schilstra et al., 2006) that allows users to convert CellML based models into SBML. It is also possible to use tools such as VCell to inter-convert between these two standards.

4.3 Other Related Standards

4.3.1 Graphical Layout

Graphical modeling applications (Bergmann et al., 2006) routinely enhance computational models by layout annotations. Recently the SBML community has decided on a common standard on how to embed the layout information within SBML. The layout extension (Gauges et al., 2006) allows a model to store the size and dimension of all model elements, along with textual annotations and reactions. Originally the intention was to embed the layout extension in a model annotation for level 2 versions of SBML but with the upcoming level 3 the layout extension will be added to the SBML as a first-class construct. LibSBML has been modified to provide access to all elements of the Layout Extension. Also several reference implementations exist (Bergmann et al., 2006; Deckard et al., 2006).

Whereas the layout extension is concerned with representing simple elements, the Systems Biology Graphical Notation (SBGN) (`http://sbgn.org`) aims to standardize the visual language of computational models unambiguously. While this standard is still in development and strictly speaking independent of the SBML effort, experience in other fields such as electrical engineering has demonstrated the essential need for standardizing the visual notation for representing models in diagrammatic form.

4.3.2 MIRIAM

Model Definition Languages such as SBML and CellML target the exchange of models. They aim to pass on the quantitative computational models from one software tool to another. However these description formats do not concern themselves with semantic annotations. Both SBML and CellML have launched efforts to remedy this problem. Both communities agreed on the Minimum Information Requested In the Annotation of biochemical Models (MIRIAM, Le Novère et al. (2005)). These annotations aim to further the confidence in quantitative biochemical models, making it easier and more precise to search for particular biochemical models, enabling researchers to identify biological phenomena captured by a biochemical model and perhaps most importantly to facilitate model reuse and model composition.

In order to call a model MIRIAM compliant, the model has to be encoded in a standard format, such as SBML. Furthermore it needs to be tied to a reference description, describing the properties and results that can be obtained from the model. Parameters of the computational model have to be provided so that the model can be loaded into a simulation environment where the results can be reproduced. Other information that has to be provided is a name for the model, the creator of the model the date and time of the last modification as well as a statement about the terms of distribution.

4.3.3 SBO – Systems Biology Ontology

In order to assign meaning to model constituents an ontology specific to Systems Biology has been developed: The Systems Biology Ontology (SBO, `http://www.ebi.ac.uk/sbo/`). The ontology consists of five controlled vocabularies and two relationships: is-part-of and is-a. Qualifying model participants, say as enzyme, macromolecule, metabolite or small species such as an ion will make it easier to generate meaning from the model. It will make the generation of standard visual notations such as SBGN possible. Moreover it presents a solution on how to interpret the model computationally, as the SBO allows tagging a model as continuous, discrete or logical model. One could even go a step further, making kinetic interactions in a model obsolete, by just referencing that the rate law is one specified by an ontology identifier (e.g.: tagging a reaction as following Henri-Michaelis Menten enzyme kinetics and specifying the parameters). The SBO is community driven and new terms or modifications to the existing onotology can be requested by the community.

4.4 Other Ontologies

The most recent developments in CellML and particularly the SBML communities revolve around the creation of ontologies and refining the exchange semantics. Apart from classifying model constituents with an appropriate ontology, one of the current areas of interest is describing the dynamical behavior of a model. The "Terminology for the Description of Dynamics" (TEDDY, `http://www.ebi.ac.uk/compneur-srv/teddy/`) provides a rich ontology to describe and quantify what kind of behavior a computational model is able

to exhibit (e.g.: the characteristics of a model could describe bifurcation behavior where the functionality of a model could be described as featuring oscillations or switch behavior). However knowing that a model exhibits interesting behavior is not enough: more information is needed in order to recreate that behavior. The "Minimum Information About a Simulation Experiment" (MIASE, `http://www.ebi.ac.uk/compneur-srv/miase/`) project focuses in this problem. MIASE will help to describe the simulation algorithms and the simulation tool used along with all needed parameter settings. In order to do so it will use the Kinetic Algorithm Ontology (KiSAO) that relates simulation algorithms and methods to each other. As these ontologies are still currently under devlopment, it will be interesting to see how they progress and are taken up by the community.

Lastly we should mention BioPAX (Luciano and Stevens, 2007), Biological Pathway Exchange. BioPAX is an XML based format that will act as a bridge between different pathway databases and data. In relation to modeling software, BioPAX may offer a means to embed rich annotation data into an SBML or CellML model. Some of this capability is being addressed to a limited extend by the new ontologies being developed at EBI in Cambridge, UK. However, BioPAX, given it role to allow a common exchange of biological data between pathway database, may offer a useful complementary way to bind data to computational models.

4.5 Human Readable Formats

SBML and CellML are examples of formats that use XML to represent information. One advantage to using XML is that there is much software available to assist in reading and manipulating XML based data. However, XML is not suited for human consumption but is designed strictly to be read by computer software. In order for humans to build and read models, human readable formats are required, often these are text based but sometimes they are graphical. In relation to text based formats, there has been a long tradition to using human readable formats for representing biochemical models, starting with BIOSSIM (Garfinkel et al., 1970). Other examples of early human readable formats include work by Park (Park and Wright, 1973) and Burns (Burns, 1971) to cite but a few. In more recent years simulators such as SCAMP (Sauro and Fell, 1991) and METAMOD (Hofmeyr

and van der Merwe, 1986) also introduced human readable formats to define models. Both software tools were developed in later years into Jarnac and PySCeS respectively.

Other formats of interest include composable languages developed by James McCollum at the University of Miami, Sauro and Bergmann (Bergmann and Sauro, 2006) at the University of Washington and Michael Pederson at the University of Edinburgh (Pedersen and Plotkin, 2008). Blinov, Faeder, Goldstein and Hlavacek developed BioNetGen (Blinov et al., 2004) which is a rule based format for representing systems with multiple states, CytoSim, which we have mentioned previously, incorporates an interesting human readable language for representing biochemical systems. The SBML community (Wilkinson, 2007) has also developed a human readable script called SBML-shorthand. This notation maps directly on to SBML but is much easier to hand write compared to SBML (as are all these human readable formats). The shorthand is also much less verbose and uses infix to represent expressions rather than MathML. Finally we should mention a lisp based language called little b (`http://www.littleb.org/`) being developed at Harvard University. The aim of little b is to allow biologists to build models quickly and easily from shared parts.

5 Databases

Along with the standardization of model representation there has been an obvious desire to create model repositories where models published in journals can be stored and retrieved. There are at the present time, five repositories with varying degrees of quality and usability. Probably the most promising is the UK based searchable , BioModels Database, which at the current time (July 2008) holds over one hundred and seventy fully curated and working models that can be downloaded in standard SBML as well as other formats. BioModels also has the great benefit of providing programmatic access to its database via web services which allows any software program to access the database seamlessly across the internet. Models stored in the BioModels Database are curated, meaning that models will reproduce the original's authors intention. In addition, the models are liberally annotated so that model components can be referenced from other database sources.

Another large database has been assembled by the CellML community Lloyd et al. (2008) which has over three hundred models stored in CellML format. From their site it is possible in principle to convert the CellML into standard C code for compilation in to a working model.

The JSim group at the University of Washington has a large database of physiological models `http://nsr.bioeng.washington.edu/Models/` stored in the mathematical language used by the JSim simulation application. These models can only be read by JSim and currently there is no simply way to translate these to any of the common exchange formats though this is likely to change in the future.

Another small but very useful database is the JWS online database developed by Brett Olivier and Jacky Snoep (Olivier and Snoep, 2004) which has almost eighty fully working models. JWS allows export in both SBML and the script format PySCeS which can be easily translated to other formats such as Jarnac script. JWS is arguably one of the first databases of models although physiological models such as those supported by JSim have been available for longer. Many if not all the models on JWS have also been ported to the BioModels Database and vice versa.

Another database, DOQCS `http://doqcs.ncbs.res.in/`focuses on signaling networks which contains over two hundred models. Models in DOQCS can only be downloaded however in Genesis format (Bhalla, 2002) which limits the portability to other frameworks. Recently the DOQCS database has been merged with the BioModels Database.

Finally there is a major NIH sponsored database called Sigmoid (Cheng et al., 2005) which currently has about twenty models. The focus of the Sigmoid project however appears to be infrastructure rather than curation which explains the limited number of available models. Access to the database is limited to the model explorer tool SME (`www.sigmoid.org`), thus access from other applications may not be possible. In addition models can only be accessed from the web interface using Cellerator format which limits portability to other frameworks. Work on Sigmoid is ongoing and no doubt changes will occur in the future to make it more open.

6 Test Suites

An important need in the simulation community is some means to compare and test simulation codes. Currently there are available a number of test suites expressed in SBML. One is provided by Andrew Finney (`http://sbml.org/wiki/Semantic_Test_Suite`) which provides various tests for deterministic models and another by Evans Evans, Gillespie and Wilkinson., (Evans et al., 2008) which represents a stochastic test suite. Currently the deterministic test suite is undergoing a redesign because it has significant issues related to numerical stability. The stochastic test suite has recently gone through a second iteration and is an extremely useful resource. Another suite of models that can be used for testing is the BioModels Database of models. Although test results are not available, it is possible to use the database models for comparison purposes as has been done by Bergmann (`http://www.sys-bio.org/sbwWiki/compare`,(Bergmann and Sauro, 2008)). The comparison site gives detailed side-by-side information on how different simulators solve a given SBML model. Overall agreement of the simulation packages appears to be high.

The CellML team in Auckland are in the process of developing a test suite for CellML which should aid considerably in testing CellML compliant applications.

7 Future Prospects

In surveying the development of software in systems biology we see a vibrant and sometimes innovative community with a very wide range of tools to satisfy all manner of users. There are still some areas that are lacking, most notably bifurcation analysis and model composition. There are some very notable tools for bifurcation analysis but they have been written more for general use than specifically for systems biology (Oscill8 being a notable exception). User interface design is still somewhat primitive in these bifurcation tools and there is much opportunity for innovation in this area particularly with respect to interactive bifurcation analysis. The second area that is lacking is model composition (*cf.* ProMoT). As models become more common place, there is a growing desire to be able to take models and combine them

easily. Currently this is not possible without considerable effort. Model composition is not a simple problem to solve however as there are many issues to consider including unit consistency and interface protocols. The benefits would be significant however, and with the rise of synthetic biology there is even more reason to develop the idea.

7.1 Reusable Software Libraries

One of the issues that has plagued software development in systems biology is chronic reinvention. Many tools, particularly the tools listed at SBML.org carry out similar functions at their core (solving ODEs, computing steady states, sensitivities etc). The fact that each tool reimplements the same functionality is arguably a waste of resources and it would be of great benefit to the community if some of the core functionality could be released in the form of reusable software libraries. There are many benefits to such an approach including, improved testing, documentation, maintainability, and extensibility. Unless there are strong reasons to do so, very few developers now write their own stiff ODE solver or SBML reader. There are already some useful libraries in the community such as libSBML, SUNDIALS, ODEPACK, LAPACK etc. These libraries are successful for a number of reasons, they are well documented, open, and most importantly, they are written in languages (C and FORTRAN) that make it easy to interface them to other programming languages. Libraries that are written in other languages, such as Java or C++, or where a C interface is not exposed, tend to be less successful. We have seen some efforts to develop additional libraries, most notably SOSLib and StochKit, but these are still under development and lack some critical functionality (SOSLib lacks the ability to remove dependent species) or ease of integration in the case of StochKit. These issues will no doubt be resolved in the future and will then allow developers to focus on other areas of interest such as user interface design and the development of new analysis approaches.

Acknowledgements

We would like to acknowledge the generous support from a number of funding agencies including the US Department of Energy GTL program, and the NIH (1R01GM081070-01). We would also like to acknowledge the many useful discussions we have had over the years with our colleagues in the software and computational systems biology community, in particular we would like to acknowledge Athel Cornish-Bowden, David Fell, Jannie Hofmeyr, Mike Hucka and Pedro Mendes,

References

Adalsteinsson, D., McMillen, D., and Elston, T. C. (2004). Biochemical Network Stochastic Simulator (BioNetS): software for stochastic modeling of biochemical networks. *BMC Bioinformatics*, 5:24–24.

Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits (Chapman & Hall/Crc Mathematical and Computational Biology Series)*. Chapman & Hall/CRC.

Alves, R., Antunes, F., and Salvador, A. (2006). Tools for kinetic modeling of biochemical networks. *Nat Biotechnol*, 24(6):667–672.

Ander, M., Beltrao, P., Di Ventura, B., Ferkinghoff-Borg, J., Foglierini, M., Kaplan, A., Lemerle, C., Tomás-Oliveira, I., and Serrano, L. (2004). SmartCell, a framework to simulate cellular processes that combines stochastic approximation with diffusion and localisation: analysis of simple networks. *Syst Biol (Stevenage)*, 1(1):129–138.

Anderson, E., McKenney, A., Sorensen, D., Bai, Z., Bischof, C., Blackford, L., Demmel, J., Dongarra, J., Du Croz, J., Hammarling, S., et al. (1999). *LAPACK Users' guide*. Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

Andreas, D., Nadine, H., Jochen, S., Adrian, S., and Andreas, Z. (2008). SBMLsqueezer: A CellDesigner plug-in to generate kinetic rate equations for biochemical networks. *BMC Systems Biology*, 2:39.

Arkin, A., Ross, J., and McAdams, H. H. (1998). Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli cells. *Genetics*, 149:1633–48.

Back, A., Guckenheimer, J., Myers, M., Wicklin, F., and Worfolk, P. (1992). DsTool: Computer assisted exploration of dynamical systems. *Notices Amer. Math. Soc*, 39(4):303–309.

Bergmann, F. and Sauro, H. (2008). Comparing Simulation Results of SBML Capable Simulators. *Bioinformatics*.

Bergmann, F. T. and Sauro, H. M. (2006). Human Readable Model Definition Language. `http://sys-bio.org/sbwWiki`.

Bergmann, F. T., Vallabhajosyula, R. R., and Sauro, H. M. (October 2006). Computational Tools for Modeling Protein Networks. *Current Proteomics*, 3:181–197(17).

Bhalla, U. S. (2002). Use of Kinetikit and GENESIS for modeling signaling pathways. *Methods Enzymol*, 345:3–23.

Blinov, M. L., Faeder, J. R., Goldstein, B., and Hlavacek, W. S. (2004). BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291.

Bornstein, B., Keating, S., Jouraku, A., and Hucka, M. (2008). LibSBML: an API Library for SBML. *Bioinformatics*, 24(6):880.

Bower, J. and Beeman, D. (1998). *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System*. Springer-Verlag New York, Inc. New York, NY, USA.

Broderick, G., Ru'aini, M., Chan, E., and Ellison, M. J. (2005). A life-like virtual cell membrane using discrete automata. *In Silico Biol*, 5(2):163–178.

Burns, J. (1969). Steady states of general multi-enzyme networks and their associated properties. Computational approaches. *FEBS Lett*, 2 Suppl 1:S30–S33.

Burns, J. A. (1971). *Studies on Complex Enzyme Systems*. PhD thesis, University of Edinburgh. `http://www.sys-bio.org/BurnsThesis`.

Carnevale, N. and Hines, M. (2006). *The NEURON Book*. Cambridge University Press.

Chance, B. (1943). The Kinetics of the Enzyme-Substrate Compound of Peroxidase. *Journal of Biological Chemistry*, 151(2):553–577.

Cheng, J., Scharenbroich, L., Baldi, P., and . Mjolsness (2005). Sigmoid: A Software Infrastructure for Pathway Bioinformatics and Systems Biology. *IEEE Intelligent Systems*, 20(3):68–75.

Chickarmane, V., R., A., Sauro, H. M., and Nadim, A. (2007). A Model for p53 Dynamics Triggered by DNA Damage. *SIAM Journal on Applied Dynamical Systems*, 6(1):61–78.

Coggan, J. S., Bartol, T. M., Esquenazi, E., Stiles, J. R., Lamont, S., Martone, M. E., Berg, D. K., Ellisman, M. H., and Sejnowski, T. J. (2005). Evidence for ectopic neurotransmission at a neuronal synapse. *Science*, 309(5733):446–451.

Cook, D. L. and Atkins, W. M. (1997). Enhanced detoxication due to distributive catalysis and toxic thresholds: a kinetic analysis. *Biochemistry*, 36(36):10801–10806.

Cook, D. L., Farley, J. F., and Tapscott, S. J. (2001). A basis for a visual language for describing, archiving and analyzing functional models of complex biological systems. *Genome Biol*, 2(4)(4).

Cox, C., Peterson, G., Allen, M., Lancaster, J., McCollum, J., Austin, D., Yan, L., Sayler, G., and Simpson, M. (2003). Analysis of Noise in Quorum Sensing. *Omics A Journal of Integrative Biology*, 7(3):317–334.

Curti, M., Degano, P., and Baldari, C. T. (2003). Causal pi-Calculus for Biochemical Modelling. In *CMSB '03: Proceedings of the First International Workshop on Computational Methods in Systems Biology*, pages 21–33, London, UK. Springer-Verlag.

Cyganowski, S., Kloeden, P., and Ombach, J. (2001). *From Elementary Probability to Stochastic Differential Equations with MAPLE*. Springer.

Deckard, A., Bergmann, F. T., and Sauro, H. M. (2006). Supporting the SBML layout extension. *Bioinformatics*, 22(23):2966–2967.

Degenring, D., Röhl, M., and Uhrmacher, A. M. (2004). Discrete event, multi-level simulation of metabolite channeling. *Biosystems*, 75(1-3):29–41.

Dhar, P., Meng, T. C., Somani, S., Ye, L., Sairam, A., Chitre, M., Hao, Z., and Sakharkar, K. (2004). Cellware–a multi-algorithmic software for computational systems biology. *Bioinformatics*, 20(8):1319–1321.

Doedel, E. J. (1981). Auto: a program for the automatic bifurcation analysis of autonomous systems. In *Proc. Manitoba Conf. Num. Math. Comput., 10th, Winnipeg, Canada*. [Congressus Numeratium, 30:265–284].

Ermentrout, B. (2002). *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to Xppaut for Researchers and Students*. Society for Industrial Mathematics; 1st edition.

Evans, T. W., Gillespie, C. S., and Wilkinson, D. J. (2008). The SBML Discrete Stochastic Models Test Suite. *Bioinformatics*, 24(2):285–286.

Fell, D. A. and Small, J. R. (1986). Fat synthesis in adipose tissue: an examination of stoichiometric constraints. *Biochem. J.*, 238:781–786.

Finney, A. and Hucka, M. (2003). Systems biology markup language: Level 2 and beyond. *Biochemical Society Transactions*, 31(6):1472–1473.

Garfinkel, D. (1968). A Machine-Independent Language for the Simulation of Complex Chemical and Biochemical Systems. *Comput. Biomed. Res.*, 2:31–44.

Garfinkel, D., Garfinkel, L., Pring, M., Green, S. B., and Chance, B. (1970). Computer applications to biochemical kinetics. *Annu Rev Biochem*, 39:473–498.

Garny, A., Kohl, P., and Noble., D. (2003). Cellular Open Resource (COR): A public CellML based environment for modeling biological function. *Int.J.Bifurcat.Chaos*, 12:3579–3590.

Gauges, R., Kummer, U., Sahle, S., and Wegner, K. (2006). A Model Diagram Layout Extension for SBML. *Bioinformatics*, 22(15):1879–1885.

Geva-Zatorsky, N., Rosenfeld, N., Itzkovitz, S., Milo, R., Sigal, A., Dekel, E., Yarnitzky, T., Liron, Y., Polak, P., Lahav, G., and Alon, U. (2006). Oscillations and variability in the p53 system. *Mol Syst Biol*, 2:2006–2006.

Gillespie, D. T. (1976). A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *J.Comp. Phys.*, 22:403–434.

Gillespie, D. T. (2007). Stochastic simulation of chemical kinetics. *Annu Rev Phys Chem*, 58:35–55.

Ginkel, M., Kremling, A., Nutsch, T., Rehner, R., and Gilles, E. D. (2003). Modular modeling of cellular systems with ProMoT/Diva. *Bioinformatics*, 19(9):1169–1176.

Hattne, J., Fange, D., and Elf, J. (2005). Stochastic reaction-diffusion simulation with MesoRD. *Bioinformatics*, 21(12):2923–2924.

Hedley, W. J., Melanie, N. R., Bullivant, D., Cuellar, A., Ge, Y., Grehlinger, M., Jim, K., Lett, S., Nickerson, D., Nielsen, P., and Yu, H. (2001). CellML Specification. Available via the World Wide Web at `http://www.cellml.org`.

Henzinger, J. F. . T. A. (2007). Executable cell biology. *Nature Biotechnology*, 25:1239–1249.

Hindmarsh, A. C. (1983). ODEPACK, a systematized collection of ode solvers in scientific computing. In Stepleman, R., editor, *Scientific Computing*, pages 55–64. North-Holland, Amsterdam.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., and Woodward, C. S. (2005). SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, 31(3):363–396.

Hines, M. (1993). *NEURON - program for simulation of nerve equations.*

Hofmeyr, J. H. and van der Merwe, K. J. (1986). METAMOD: software for steady-state modelling and control analysis of metabolic pathways on the BBC microcomputer. *Comp. Appl. Biosci.*, 2:243–249.

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(24):3067–3074.

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J. H., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novère, N., Loew, L. M., Lucio, D., Mendes, P., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff, J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., and Wang, J. (2003). The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models. *Bioinformatics*, 19:524–531.

Kacser, H. and Burns, J. A. (1973). The Control of Flux. In Davies, D. D., editor, *Rate Control of Biological Processes*, volume 27 of *Symp. Soc. Exp. Biol.*, pages 65–104. Cambridge University Press.

Kauffman, K. J., Prakash, P., and Edwards, J. S. (2003). Advances in flux balance analysis. *Curr Opin Biotechnol*, 14(5):491–496.

Keating, S. M., Bornstein, B. J., Finney, A., and Hucka, M. (2006). SBMLToolbox: an SBML toolbox for MATLAB users. *Bioinformatics*, 22(10):1275–1277.

Kholodenko, B. N. (2006). Cell-signalling dynamics in time and space. *Nat Rev Mol Cell Biol*, 7(3):165–176.

Kitano, H., Funahashi, A., Matsuoka, Y., and Oda, K. (2005). Using process diagrams for the graphical representation of biological networks. *Nat Biotechnol*, 23(8):961–966.

Klamt, S., Saez-Rodriguez, J., and Gilles, E. D. (2007). Structural and functional analysis of cellular networks with CellNetAnalyzer. *BMC Syst Biol*, 1:2–2.

Klipp, E., Herwig, R., Kowald, A., Wierling, C., and Lehrach, H. (2005). *Systems Biology in Practice*. Wiley-VCH Verlag, Weinheim.

Kolpakov, F. (2004). BioUML—open source extensible workbench for systems biology. pages 25–30.

Kraus, M., Lais, P., and Wolf, B. (1992). Structured biological modelling: a method for the analysis and simulation of biological systems applied to oscillatory intracellular calcium waves. *BioSystems*, 27:145–169.

Kubicek, M. (1976). Algorithm 502. Dependence of solution of nonlinear systems on a parameter. *ACM Trans. on Math. Software*, 2:98–107.

Kubicek, M. and Marek, M. (1983). *Computational Methods in Bifurcation Theory and Dissipative Structures.* Springer Series in Computational Physics. Springer-Verlag.

Kurata, H., Inoue, K., Maeda, K., Masaki, K., Shimokawa, Y., and Zhao, Q. (2007). Extended CADLIVE: a novel graphical notation for design of biochemical network maps and computational pathway analysis. *Nucleic Acids Res*, 35(20).

Le Novère, N. (2006). Model storage, exchange and integration. *BMC Neurosci*, 7 Suppl 1:S11.

Le Novère, N., Finney, A., Hucka, M., Bhalla, U. S., Campagne, F., Collado-Vides, J., Crampin, E. J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J. L., Spence, H. D., and Wanner, B. L. (2005). Minimum Information Requested In theAannotation of biochemical Models (MIRIAM). *Nature biotechnology*, 23(12):1509–1515.

Le Novère, N. and Shimizu, T. S. (2001). STOCHSIM: modelling of stochastic biomolecular processes. *Bioinformatics*, 17(6):575–576.

Lee, J. M., Gianchandani, E. P., and Papin, J. A. (2006). Flux balance analysis in the era of metabolomics. *Brief Bioinform*, 7(2):140–150.

Lemerle, C., Di Ventura, B., and Serrano, L. (2005). Space as the final frontier in stochastic simulations of biological systems. *FEBS Lett*, 579(8):1789–1794.

Levenberg, K. (1944). A method for the solution of certain nonlinear problems in least squares. *Q. Appl. Math*, 2:164–168.

Li, H., Cao, Y., Petzold, L., and Gillespie, D. (2007). Algorithms and Software for Stochastic Simulation of Biochemical Reacting Systems. *Biotechnology Progress*, Web Release Date: September 26.

Lloyd, C. M., Halstead, M. D., and Nielsen, P. F. (2004). CellML: its future, present and past. *Prog Biophys Mol Biol.*, 85:433–50.

Lloyd, C. M., Lawson, J. R., Hunter, P. J., and Nielsen, P. F. (2008). The CellML Model Repository. *Bioinformatics*.

Luciano, J. S. and Stevens, R. D. (2007). e-Science and biological pathway semantics. *BMC Bioinformatics*, 8 Suppl 3:8 Suppl 3: S3.

Machné, R., Finney, A., Müller, S., Lu, J., Widder, S., and Flamm, C. (2006). The SBML ODE Solver Library: a native API for symbolic and fast numerical analysis of reaction networks. *Bioinformatics*, 22(11):1406–1407.

Maiwald, T. and Timmer, J. (2008). Dynamical Modeling and Multi-Experiment Fitting with PottersWheel. *Bioinformatics*. doi:10.1093/bioinformatics/btn350.

Manninen, T., Makiraatikka, E., Ylipaa, A., Pettinen, A., Leinonen, K., and Linne, M. L. (2006). Discrete stochastic simulation of cell signaling: comparison of computational tools. *Conf Proc IEEE Eng Med Biol Soc*, 1:2013–2016.

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Ind. Appl. Math*, 11(2):431–441.

McCollum, J., Lancaster, J., Bouldin, D., and Peterson, G. (2003). Hardware acceleration of pseudo-random number generation for simulation applications. *System Theory, 2003. Proceedings of the 35th Southeastern Symposium on*, pages 299–303.

Mendes, P. (1993). GEPASI: A software package for modelling the dynamics, steady states and control of biochemical and other systems. *Comput. Applic. Biosci.*, 9:563–571.

Moles, C. G., Mendes, P., and Banga, J. R. (2003). Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Res*, 13(11):2467–2474.

Moniz-Barreto, P. and Fell, D. A. (1993). Simulation of dioxygen free radical reactions. *Biochem Soc Trans*, 21 (Pt 3)(3):256–256.

Moraru, I., Schaff, J. C., Slepchenko, B. M., and Lowe, L. M. (2002). The Virtual Cell: An Integrated Modeling Environment for Experimental and Computational Cell Biology. *Ann NY Acad Sci*, 971(1):595–596.

Myers, C. R., Gutenkunst, R. N., and Sethna, J. P. (2007). Python Unleashed on Systems Biology. *Computing in Science and Engg.*, 9(3):34–37.

Nelder, J. and Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308.

Neves, S. and Iyengar, R. (2002). Modeling of signaling networks. *BioEssays*, 24(12):1110–1117.

Olivier, B. and Snoep, J. (2004). Web-based kinetic modelling using JWS Online. *Bioinformatics*, 20(13):2143–2144.

Olivier, B. G., Rohwer, J. M., and Hofmeyr, J. H. (2005). Modelling cellular systems with PySCeS. *Bioinformatics*, 21:560–1.

Palsson, B. O. (2007). *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press.

Park, D. J. M. and Wright, B. E. (1973). METASIM, A General Purpose Metabolic Simulator for Studying Cellular Transformations. *Comput. Progm. Biomed.*, 3:10–26.

Pedersen, M. and Plotkin, G. (2008). A Language for Biochemical Systems. In *Computational Methods in Systems Biology*. Springer. in press, `http://homepages.inf.ed.ac.uk/s0677975/papers/lbs.pdf`.

Peterson, G. D. and Lancaster, J. M. (2002). Stochastic simulation of biological cellular processes using VHDL-AMS. *Behavioral Modeling and Simulation, 2002. BMAS 2002. Proceedings of the 2002 IEEE International Workshop on*, pages 118–122.

Poolman, M. G. (2006). ScrumPy: metabolic modelling with Python. *IEE Proc Syst Biol*, 153(5):375–378.

Price, N. D., Papin, J. A., Schilling, C. H., and Palsson, B. O. (2003). Genome-scale microbial in silico models: the constraints-based approach. *Trends in Biotechnology*, 21:162–169.

Ramsey, S., Orrell, D., and Bolouri, H. (2005). Dizzy: stochastic simulation of large-scale genetic regulatory networks. *J Bioinform Comput Biol*, 3(2):415–436.

Rao, R., Wolf, D. M., and Arkin, A. P. (2002). Control, exploitation and tolerance of intracellular noise. *Nature*, 420:231–237.

Raymond, G. M., Butterworth, E., and Bassingthwaighte, J. B. (2003). JSIM: Free software package for teaching physiological modeling and research. *Exper Biol*, 280.5:102.

Reder, C. (1988). Metabolic Control Theory: A Structural Approach. *J. Theor. Biol.*, 135:175–201.

Reich, J. G. and Selkov, E. E. (1981). *Energy metabolism of the cell*. Academic Press, London.

Rost, U. and Kummer, U. (2004). Visualisation of biochemical network simulations with SimWiz. *Syst Biol (Stevenage)*, 1(1):184–189.

Runarsson, T. and Yao, X. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews.*, 35(2)(14):233–243.

Salis, H., Sotiropoulos, V., and Kaznessis, Y. N. (2006). Multiscale Hy3S: hybrid stochastic simulation for supercomputers. *BMC Bioinformatics*, 7:93–93.

Sanford, C., Yip, M. L., White, C., and Parkinson, J. (2006). Cell++– simulating biochemical pathways. *Bioinformatics*, 22(23):2918–2925.

Sauro, H. M. (1993). SCAMP: a general-purpose simulator and metabolic control analysis program. *Comput Appl Biosci*, 9(4):441–450.

Sauro, H. M. (2000). Jarnac: A System for Interactive Metabolic Analysis. In Hofmeyr, J.-H. S., Rohwer, J. M., and Snoep, J. L., editors, *Animating the Cellular Map: Proceedings of the 9th International Meeting on Bio-ThermoKinetics*. Stellenbosch University Press.

Sauro, H. M. and Fell, D. A. (1991). SCAMP: A metabolic simulator and control analysis program. *Mathl. Comput. Modelling*, 15:15–28.

Sauro, H. M., Hucka, M., Finney, A., Wellock, C., Bolouri, H., Doyle, J., and Kitano, H. (2003). Next Generation Simulation Tools: The Systems Biology Workbench and BioSPICE Integration. *OMICS*, 7(4):355–372.

Sauro, H. M. and Ingalls, B. (2004). Conservation analysis in biochemical networks: computational issues for software writers. *Biophys Chem*, 109(1):1–15.

Savageau, M. A. (1972). The behaviour of intact biochemical control systems. *Curr. Topics Cell. Reg.*, 6:63–130.

Schaff, J., Fink, C. C., Slepchenko, B., Carson, J. H., and Loew, L. M. (1997). A general computational framework for modeling cellular structure and function. *Biophys J*, 73(3):1135–1146.

Schilstra, M. J., Li, L., Matthews, J., Finney, A., Hucka, M., and Le Novère, N. (2006). CellML2SBML: conversion of CellML into SBML. *Bioinformatics*, 22(8):1018–1020.

Schmidt, H. and Jirstrand, M. (2006). Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology. *Bioinformatics*, 22(4):514–515.

Schwender, J. (2008). Metabolic flux analysis as a tool in metabolic engineering of plants. *Curr Opin Biotechnol*, 19(2):131–137.

Sedwards, S. and Mazza, T. (2007). Cyto-Sim: a formal language model and stochastic simulator of membrane-enclosed biochemical processes. *Bioinformatics*, 23(20):2800–2802.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res*, 13(11):2498–2504.

Shapiro, B. E., Hucka, M., Finney, A., and Doyle, J. (2004). MathSBML: a package for manipulating SBML-based biological models. *Bioinformatics*, 20(16):2829–2831.

Slepchenko, B. M., Schaff, J., Macara, I. G., and Loew, L. M. (2003). Quantitative Cell Biology with the Virtual Cell. *Trends in Cell Biology*, 13:570–576.

Sorokin, A., Paliy, K., Selkov, A., Demin, O. V., Dronov, S., Ghazal, P., and Goryanin, I. (2006). The pathway editor: a tool for managing complex biological networks. *IBM J. Res. Dev.*, 50(6):561–573.

Stiles, J. R. and Bartol, T. M. (2001). Monte Carlo Methods for Simulating Realistic Synaptic Microphysiology Using MCell. In Schutter, E. D., editor, *Computational Neuroscience: Realistic Modeling for Experimentalists*, pages 87–127. CRC Press.

Suderman, M. and Hallett, M. (2007). Tools for visually exploring biological networks. *Bioinformatics*, 23(20):2651–2659.

Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T. S., Matsuzaki, Y., Miyoshi F, S. K., Tanida, S., Yugi, K., Venter, J. C., and Hutchison, C. A. (1999). E-CELL: software environment for whole-cell simulation. *Bioinformatics*, 15:72–84.

Tyson, J. J., Chen, K., and Novak, B. (2001). Network Dynamics And Cell Physiology. *Nature Reviews Molecular Cell Biology*, 2:908–916.

Tyson, J. J., Csikasz-Nagy, A., and Novak, B. (2002). The dynamics of cell cycle regulation. *BioEssays*, 24:1095–1109.

Ullah, M. and Wolkenhauer, O. (2007). Family tree of Markov models in systems biology. *IET Syst Biol*, 1(4):247–254.

Vacheva, I. and Eils, R. (2006). Computational Systems Biology Platforms (Computergestützte Systembiologieplattformen). *it - Information Technology*, 48(3):140–147.

Vallabhajosyula, R. R., Chickarmane, V., and Sauro, H. M. (2006). Conservation analysis of large biochemical networks. *Bioinformatics*, 22(3):346–353.

Vallabhajosyula, R. R. and Sauro, H. M. (2007). Stochastic simulation GUI for biochemical networks. *Bioinformatics*, 23(14):1859–1861.

Vass, M., Allen, N., Shaffer, C. A., Ramakrishnan, N., Watson, L. T., and Tyson, J. J. (2004). the JigCell model builder and run manager. *Bioinformatics*, 20(18):3680–3681.

Wilkinson, D. (2007). SBML Shorthand. `http://www.staff.ncl.ac.uk/d.j.wilkinson/software/sbml-sh/`.

Wilkinson, D. J. (2006). *Stochastic Modelling for Systems Biology*. Chapman and Hall.

Williams, T., Kelley, C., et al. (1998). GNUPLOT: An Interactive Plotting Program. *Manual, version*, 3.

Wright, S. (1934). Physiological and Evolutionary Theories of Dominance. *The American Naturalist*, 68:24–53.

Yoshimi, M., Osana, Y., Fukushima, T., and Amano, H. (2004). Stochastic simulation for biochemical reactions on FPGA. *The 14th International Conference on Field Programmable Logic and Applications*, 3203:105–114.

Zi, Z. and Klipp, E. (2006). SBML-PET: a Systems Biology Markup Language-based parameter estimation tool. *Bioinformatics*, 22(21):2704–2705.