National Simulation Resource

NSR

University of Washington

Circulatory Transport and Exchange

http://nsr.bioeng.washington.edu

Volume 10
Number 1
July, 2001

NSR is funded by NIH grant RR-01243, *Simulation Resource in Circulatory Transport and Exchange*

Designing and constructing a set of fundamental cell models

As a part of a multi-university, multi-investigator program on integrated function of the complex behavior of living cells, we are targeting the metabolism and regulation of the endothelial cell and the cardiomyocyte and the interactions between these two cell types. The program includes an educational and dissemination effort. To do this we are developing seven carefully staged computational "eternal" Cell Models and their major energy-related submodules.

The first six of these models are "eternal" in that there is no protein anabolism or catabolism, thus all enzymes remain at fixed concentrations. The emphasis is on the physico-chemical balances (charge, ionic regulation, mass, volume, energy, reducing equivalents, carbon, oxygen, phosphorus and so on), on the limitations to the kinetics imposed by the energetics, and the responses to transients rather than steady-state analysis. The models will serve as a base to link metabolism with mechanics, signaling and gene expression.

Thoroughly documented model components are being made available as a distributable Biological Component Library, "open source" on our website, so that they can be built upon by the scientific community with confidence. These primitive models will serve as tools to link overall cellular function to the underlying physiology and its regulation at the proteomic and genetic levels (J.B. Bassingthwaighte, "The modeling of a primitive 'sustainable' cell", *Phil. Trans. Roy. Soc. Lond. A,* 359, 1055–1072, 2001, or at www.pubs.royalsoc.ac.uk).

These efforts are a part of the Physiome Project sponsored by our NIH/NCRR National Simulation Resource (http://nsr.bioeng.washington.edu/Models).

by James B. Bassingthwaighte

2001 ANNUAL WORKSHOP: KINETIC MODELING of biochemical, biophysical and physiological systems

The National Simulation Resource course on applying modeling to the analysis of convection, transmembrane transport, reaction, energetics and electrophysiology will be held September 17–19, 2001, at the University of Washington, Seattle.

This course will train investigators to use simulation and modeling as aids in understanding biological systems, and as tools for analyzing data. Three main foci of the course are (i) developing model applications, (ii) general principles governing intravascular convection, membrane transport, and metabolic reactions in studies of whole organs, and (iii) analysis of time-course data from outflow dilution dynamic imaging studies (PET and NMR).

The methods, though focused on physiological transport, are general and can be applied to modeling the Physiome. "Hands on" computer work during the workshop will use JSIM, a tool for converting the equations of models into systems interfacing with equation solvers and XSIM, a graphic user interface for simulation control and analysis developed at NSR. Participants may bring problems and data from their research to be considered during the workshop.

The course will cover these topics:

- component-based model assembly;
- computer implementation techniques;
- general enzyme kinetics;
- membrane transport and receptor kinetics;
- energetics;
- electrophysiology;
- organ heterogeneity;
- cellular metabolism;
- functional imaging with kinetic models;
- fitting models to data.

Tuition is $300 and is due upon registration. Enrollment is limited to twenty participants. Register via our WWW site at http://nsr.bioeng.washington.edu, or by contacting Dr. Zheng Li at 206-685-2005 or email to zhengli@bioeng.washington.edu.

by Gary Raymond

Contents

Staff

Jim Bassingthwaighte, Director
jbb@bioeng.washington.edu

Zheng Li, Assoc. Dir., Modeling
zhengli@bioeng.washington.edu

Hong Qian, Assoc. Dir., Mathematics
hongq@bioeng.washington.edu

Gary Raymond, Applications
garyr@bioeng.washington.edu

Erik Butterworth, XSIM
butterw@bioeng.washington.edu

William Chan, Software Librarian
wchan@bioeng.washington.edu

Yasmin Lucero, Program Mgr.
yasmin@bioeng.washington.edu

Renata Chmielowski, Secretary
renata@bioeng.washington.edu

Eric Lawson, Publications
eric@bioeng.washington.edu

Postdoctoral Associates:
Ranjan Dash
rdash@bioeng.washington.edu
Anamika Sarkar
anamika@bioeng.washington.edu

Graduate Student:
Michael Kellen
mkellen@…

JSIM and XSIM

JSIM, NSR's new modeling and data-analysis tool, is directed toward three major goals: (1) platform independence, (2) dynamic equation-based and component-based model assembly and (3) facilitated collaboration. JSIM will eventually subsume most functions of XSIM, and XSIM will then be maintained with no additional functionality.

A broad outline of our plans for JSIM was presented in the last Newsletter. Since then, we have developed specific architectural designs. Those designs and their advantages are discussed under the heading "JSIM Architecture", below. We have also released software that implements the first stage of the overall plan. The features available in this first-stage release are outlined under the heading "JSIM Current Distribution". "JSIM Future Plans" presents a preview of capabilities that will be added in the near future.

Other articles in this newsletter describe uses of JSIM. Gary Raymond's article, "Modeling simplified with JSIM", presents a simple example of the use of JSIM's mathematical modeling language (MML). Mike Kellen's article, "Facilitated transport models with JSIM", describes how some simple models, easily developed with JSIM, may become the building blocks of whole cell and organ models.

JSIM Architecture

The JSIM architecture consists of five conceptual levels. Each level interacts (almost) exclusively with neighboring levels. At the core of the system, the Mathematical Modeling Language (MML) describes a set of equations in abstract mathematical form.

JSIM's Layered Architecture

1. GUI	Graphic User Interface
2. BML	Biological Component Modeling Language
3. MML	Mathematical Modeling Language
4. NMD	Numeric Method Dispatcher
5. NML	Numeric Method Library

MML is declarative, not procedural. To generate a solution for a problem posed in MML, JSIM's Numeric Method Dispatcher (NMD) assembles numerical methods into executable code after searching the Numeric Method Library (NML) for applicable methods. This allows modelers without knowledge of software development or numerical methods to create computational models from mathematical equations.

The biological component modeling language (BML) extends MML to allow construction of models from prefabricated abstract structures representing biological components, rather than from mathematical equations. Each structure is internally represented by a set of equations, but the BML writer need not be familiar with the underlying equations to create useful models. Models prepared in BML are translated to models in MML, which is common to both. BML components are dynamically defined and may be developed by diverse research groups. This provides a powerful method for integrating the efforts of separate research groups.

No modern application is complete without a graphic user interface (GUI). For JSIM, although interested users will be able to write their own GUIs for specific applications, the GUI distributed by NSR will feature:

1. platform independence (uses Java);
2. all run-time control functions currently available in XSIM;
3. improved graphic manipulation capabilities;
4. dynamic model construction.

JSIM's layered architecture facilitates cooperation by allowing experts in different fields to contribute without having detailed knowledge of all aspects of the architecture. For example, biological modelers need not be familiar with numerical methods, and numerical method specialists need not be widely versed in biology. Mathematical modelers need not understand the details of the underlying C or Java code.

JSIM Current Distribution

NSR is now distributing a beta-test version of JSIM (at http://nsr.bioeng.washington.edu/Software). For this version, an MML file is created in a text editor and submitted to JSIM via a command-line interface. JSIM then attempts to generate a working XSIM model from the equations described by the MML. If unsuccessful (if, for instance, some equations cannot be solved by numeric methods known to XSIM), JSIM issues a fatal diagnostic message. If successful, the newly created model is launched in XSIM, and all further data analysis takes place within the XSIM environment. Because XSIM must be used to run the generated model, this JSIM distribution is limited to platforms that support XSIM (i.e., Solaris, Irix and Linux).

The distributed NMD will solve problems that can be reduced to any of the following sub-problems: (1) algebraic subsitution and reordering; (2) groups of simultaneous first-order ordinary differential equations (ODEs) with initial conditions; (3) groups of simultaneous first-order partial differential equations (PDEs) in time and one spatial dimension with initial and boundary conditions. Recirculating ODE problems are supported, but recirculating PDE problems are not. Variables may be declared with various physical units (e.g., cm/sec^2), with automatic unit conversion in calculations. Symbolic derivatives are also supported in expressions.

MML files created for this beta-release of JSIM will be compatible with more sophisticated, future JSIM versions that feature run-time control and improved graphics.

JSIM Future Plans

Plans for JSIM over the next six months involve BML, NMD and GUI enhancements.

Preliminary implementation of a BML for blood–tissue exchange models is now being tested at NSR. At this time, BML and MML constructs cannot be combined. This situation will soon change, allowing more powerful model construction and the beginnings of inter-research group collaborative support.

Improvements in the NMD will allow a greater variety of boundary conditions and recirculation PDE problems to be processed than are handled by the existing PDE solver. This, in turn, will allow axially distributed regions to be incorporated into the BTEX BML. Gary Raymond and Dr. Ranjan Dash are investigating additional PDE solvers that may soon be incorporated into JSIM.

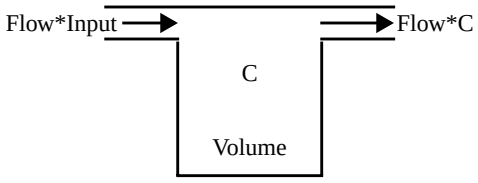
Dr. Zheng Li is developing a run-time control GUI for JSIM models, employing public domain Java graphic libraries. See "Status of JSIM graphic user interface development" on page 5. When the GUI is working, JSIM will be available to users running MS-Windows™ and the MacIntosh OS. Also, UNIX™ users will have many more sophisticated data-display options than are now available under XSIM.

by Erik Butterworth

Modeling simplified with JSIM

Before this year, if researchers wanted a new physiological or biochemical model, making the model was an arduous task that involved writing computer code, testing the model, and interfacing it with a simulation control system. JSIM has simplified this task. To show how, the JSIM code for (1) a one-compartment model, (2) a one-compartment model with units for parameters and variables included, and (3) a one-region model with advection and diffusion is shown below. The purpose of presenting the examples is to show the ease of generating models and interfacing them with XSIM, NSR's Xwindows-based simulation controller.

Example 1: The concentration, *C*, in a one-compartment model is solved as an ordinary differential equation (ODE). Note that a double slash denotes the beginning of a comment.



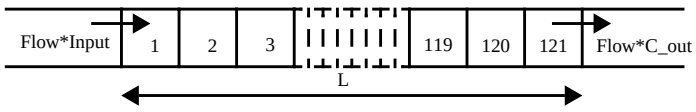

```
JSim v1.1          // Example 1: Required statements in bold face, rest optional
math example1 {    // Name the model "example1"
                   // Solve d(Volume*C)/dt = Flow*Input-Flow*C
                   // The change in the amount is what flows in minus what flows out
realDomain t; t.min=0; t.max=40; t.delta=0.1;   // Define domain of problem
real   Flow=1/60, Volume=0.07;                  // Define Parameters
extern real Input(t);                     // Input function provided by XSIM
real C(t);                                // State variable
  when(t=t.min) {C = 0;}                  // Initial condition
  C:t = Flow/Volume*(Input-C);            // The ordinary differential equation (ODE)
}
```

In Example 1, the model is named "example1". The domain of the model is time, named "t", and will run from zero to thirty seconds in increments of 0.1 seconds. Two parameters are defined, "Flow" and "Volume". The input to the single compartment is named "Input" and will be provided by the XSIM interface. The concentration, *C*(*t*), has an initial value of zero. The derivative of *C*(*t*) with respect to time is C:t, and its equation is given. All parameters and values are assumed to be dimensionally consistent. However, dimensional checking and conversion can be added with the additional of a few statements and dimensional specification. This is shown in Example 2. All modifications of Example 1 are given in boldface type.

```
JSim v1.1                 // Example 2: Changes from example 1 in boldface
import nsrunit;
unit conversion on;
math example2 {           // Name the model "example2"
                          // Solve d(Volume*C)/dt = Flow*Input-Flow*C
                          // The change in the amount is what flows in minus what flows out
realDomain t sec; t.min=0; t.max=40; t.delta=0.1;     // Define  domain of problem
real   Flow=1 ml/(g*min), Volume=0.07 ml/g;     // Define Parameters with units
extern real Input(t) mmol/ml;                   // Input function provided by XSIM
real C(t) mmol/ml;                              // State variable
  when(t=t.min) {C = 0;}                        // Initial condition
  C:t = Flow/Volume*(Input-C);                  // The ordinary differential equation
(ODE)
}
```

The time dimension unit for both the domain, *t*, and Flow can be different now—flow will be entered in ml/(g*min) but, in the problem calculation, will automatically be rescaled to ml/(g*sec). JSIM flags and prohibits errors in the dimensionality of terms.

In Example 3, a one-region distributed model, units will be retained, but the problem will be shifted from a stirred-tank model to a distributed model with diffusion, a partial differential equation (PDE).



A distributed model replaces the single stirred tank with this equation:

$$\frac{\partial C}{\partial t} = -(Flow \cdot L/Volume) \cdot \frac{\partial C}{\partial x} + D_p \cdot \frac{d^2 C}{dx^2},$$

where *L* is the length of the capillary. In the model formulation given below, *C:x* is the derivative with respect to *x*, *C:x:x* is the second derivative of *C* with respect to *x*, and D_p is the diffusion coefficient. Since this is a second-order PDE, we need two boundary conditions, an inflow value on the left, and a reflective boundary condition on the right. The coefficient for advecting the gradient, *Flow*L/Volume*, is a velocity.

```
JSim v1.1                  // Example 3: Distributed model and partial differential equations
import nsrunit;
unit conversion on;
math example3 {            // Changes to program in boldface
                           // Solve d(C)/dt = -Flow*L/Volume*dC/dx + Dp*d(dC/dx)/dx
                           // The change is the gradient of the flow plus diffusion
realDomain t sec; t.min=0; t.max=40; t.delta=0.1;     // Define domain of problem
realDomain x cm; x.min=0; x.max=0.1; x.ct=51;         // Define spatial mesh
real    Flow=1 ml/(g*min) , Volume=0.07 ml/g;         // Parameters
real    Dp =1.0E-4 cm^2/sec;
extern real Input(t) mmol/ml ;                        // Input function provided by XSIM
real C(t,x) mmol/ml , C_out(t) mmol/ml ;              // State variables
  when (x=x.min) { C=Input;}                          // Boundary conditions
  when (x=x.max) { C:x=0; C_out=C; }
  when(t=t.min) {C = 0; }                             // Initial condition
  C:t = -Flow*x.max/Volume*C:x + Dp*C:x:x ;           // Partial Differential Equation
}
```

JSIM takes the model statements and creates the model, running the statements through equation solvers and into a full simulation interface which provides model runs, optimization of parameters to fit the model to data, and sensitivity analysis (the relative change in a model output when parameters are perturbed).

These three simple examples illustrate many of the features of JSIM: ease in developing complex models; checking of equations for dimensional consistency, and providing for automatic scaling of parameters when dimensions are in different units; simplified coding of differential equations for physiological, biochemical, and biomechanical systems.

Download JSIM from the NSR website at http://nsr.bioeng.washington.edu.

by Gary Raymond

Facilitated transport models with JSIM

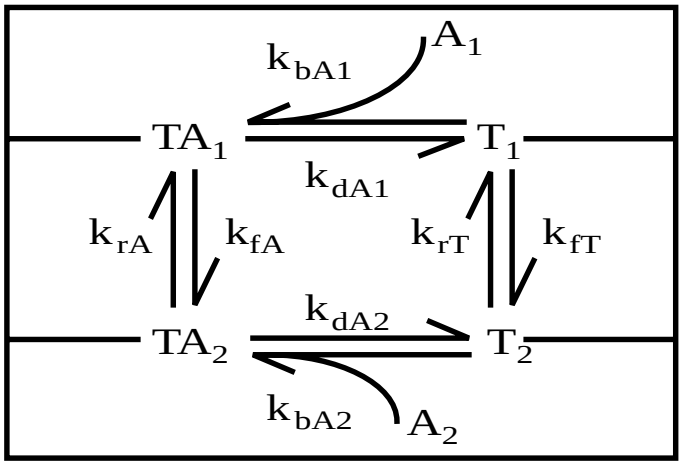
New software tools such as JSIM make the development of complex biological models easier. For example, consider a series of models of solute transport across membranes by carrier proteins. Although relatively simple individually, the models are nonetheless useful for teaching both basic biophysical principles and modeling techniques. Also, when these simple models are later joined together with other simple models, they become the building blocks of cell and organ models used in scientific research.

In facilitated transport models, transport of a substrate across a membrane occurs through three reversible steps: (1) binding of the substrate to the carrier protein, (2) a conformation change of the transporter-substrate complex, and (3) release of the substrate on the opposite side of the membrane. This simple biophysical model displays several characteristics of real transporters. Transport is saturable: it depends on a finite number of carrier proteins for substrate exchange. Binding of substrate to transporter is capacitive, which means that binding of free solute to the transporter causes a reduction in free solute concentration. Multiple species can use the same transporter, resulting in competition for binding sites. Although coupling to an external energy source is not explicitly included, appropriate choices of kinetic rate constants can simulate active transport (at an assumed constant driving potential).

An online tutorial explains these scientific concepts and introduces new users to the modeling platform, using sample simulations that can be run interactively over the internet. All stages of the model development are explained, including the development of the governing equations, the writing of the mathematical modeling language file, and the limitations of the numerical methods used to solve the equations. Three different facilitated transport models provide examples of the inherent trade-off between the completeness of a model and the complexity of the computation.

We expect that the sample models will mature as the simulation environment expands. With the development of a biological modeling language, for example, these models will become part of a set of reusable components that can be used to quickly build more complex biological models. In addition, with the development of a graphic model building environment, we expect JSIM users will be able to simply drag and drop an icon representing an active transporter into a graphical representation of a membrane separating compartments or spatially distributed regions. We also expect JSIM users will be able to access the properties of specific transporters catalogued in online databases to assist in developing physiologically realistic simulations. See http://nsr.bioeng.washington.edu/Models for the NSR's collection of online models, including a link to the facilitated transport tutorial.

by Michael Kellen



Transport of a single substrate, *A*, across a membrane by a transporter, *T*. The kinetics of the process are controlled by eight unidirectional rate constants.

Status of JSIM graphic user interface development

The JSIM Graphic User Interface (GUI), a preview version of which will be available in the third quarter of 2001, is designed to be a "one-stop" environment in which to perform multiple tasks in model simulation and analysis. These tasks include dynamic model configuration, model runtime control for simulation and data analysis, and visualization and communication with a variety of data sources or databases. The GUI is written entirely in Java: it is platform-independent, and has been tested on UNIX and Microsoft Windows systems (Windows 98 and Windows 2000).

Dynamic Model Configuration. Currently, an equation-based configuration tool can be used to edit a text file that describes the mathematical model (the JSIM model file), to generate and compile the Java model code, and to load the Java model class into the environment. A graphic configuration tool is being developed.

Model Runtime Control. Users can load multiple models into the same environment, change model parameter values and perform model simulation runs. Users can choose to make it a standalone or a distributed system. Java Remote Method Invocation (RMI) is implemented as the basic communication protocol for distributed computing. RMI supports remote object activation, so model programs can be made as computable objects sitting anywhere on the network. The client program (user interface) can use any computational model presented on the network as a remote object using RMI with a downloadable interface for activation methods. This provides a means to use high-power computer servers on the remote site.

Data Analysis. Tools to estimate parameters using automatic optimization algorithms and sensitivity analysis are being developed.

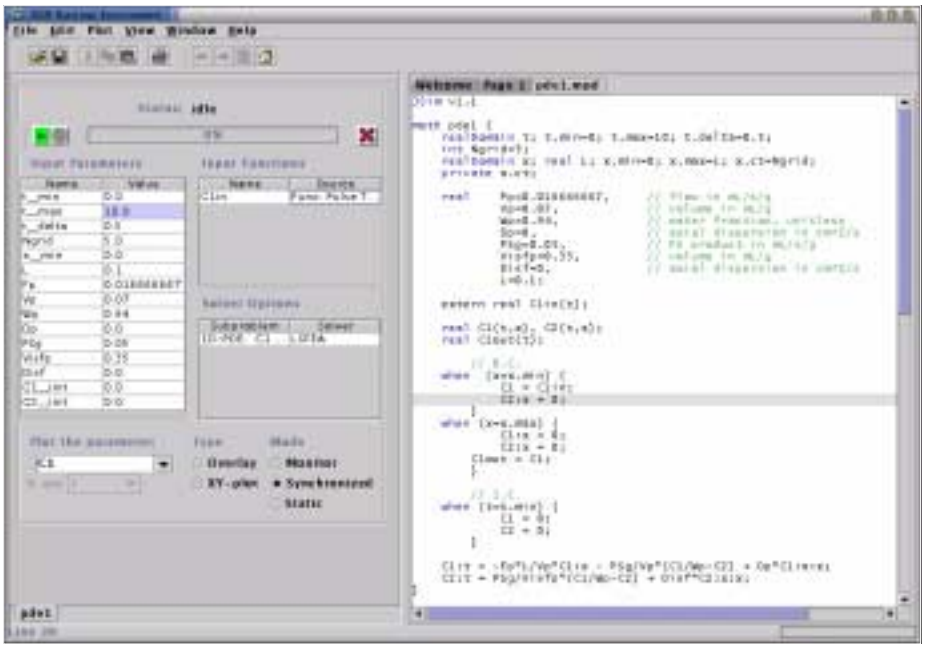
Visualization tools. Currently, two Java libraries are implemented under the GUI. One is the JAS package (jas.freehep.org), which we use for 2-D histogram and scatter plots. The other is the VisAD package from the SSEC Visualization Project (at the University of Wisconsin–Madison Space Science and Engineering Center, the Unidata Program Center, and the National Center for Supercomputer Applications). We use VisAD for visualization of 5-D data in the form of multi-variable, time-dependent functions on a 3-D special domain. The displays include 2-D colormaps, 3-D plots and animated 2-D/3-D plots.

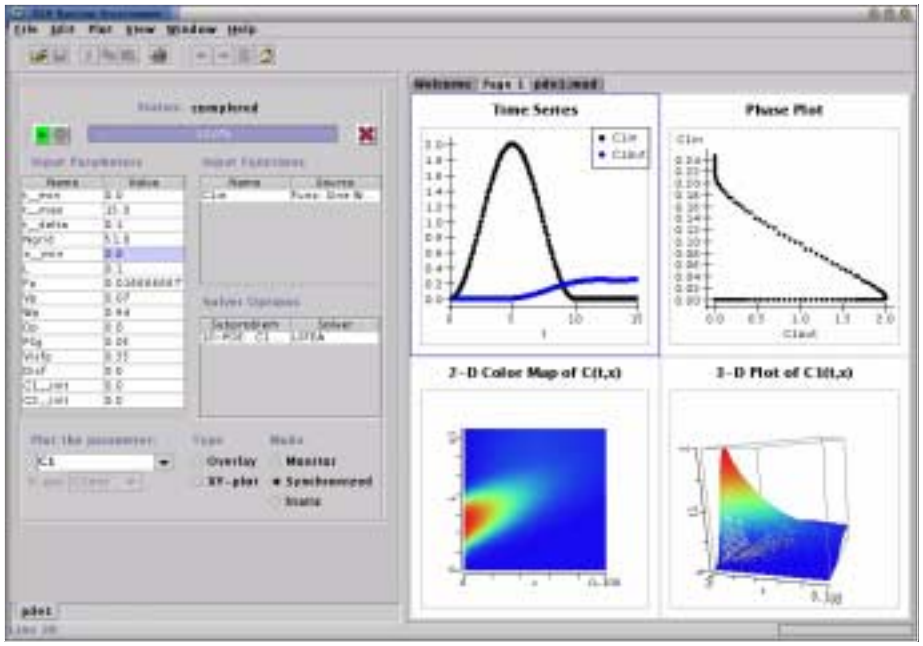
Storage and Database Connectivity. These are under development. Our design goals are (1) to allow users to preserve the state of their model simulation tasks including model parameter space and data in flat files and in databases for later use; (2) to allow users to retrieve physical parameter values (shared with many models) in databases for use in the current simulation run.

Documentation and Online Help Tools. JSIM online help, which was made with JavaHelp software, can be viewed with the JSIM GUI using a built-in browser, and can be delivered over the network. The JavaHelp system is implemented using Java Foundation Classes software components, which allow developers flexibility and ease in making custom user interfaces and functionality. For example, the JavaHelp system can be displayed in its own window or embedded in an application.

Extensibility. This is under development. The GUI has been designed to make it possible to extend its functionality by writing experiment- or user-specific modules, called plugins. This provides a means for sophisticated users to add functionality to the environment.

by Zheng Li





NSR Publications
University of Washington
Bioengineering, Box 357962
Seattle, WA 98195-7962