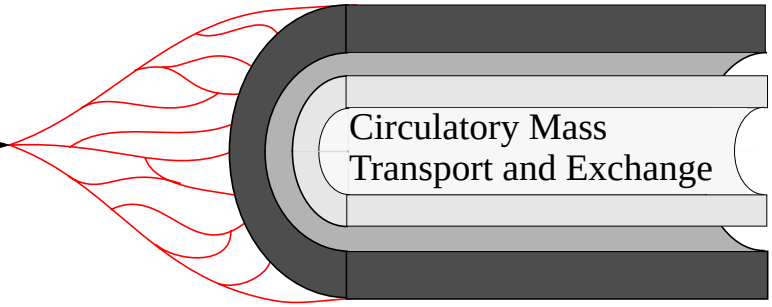
National Simulation Resource
NSR
University of Washington



Volume 2
Number 1
March, 1992

Where is this resource in the scheme of things?

The arena of modeling is limitless. Every well phrased scientific hypothesis is a mathematical model, or at least can be translated into one. Most mathematical models represent reductionist approaches to describing a system of some sort. The degree of reduction in most biological models can only be regarded as huge, implying that the model is at best a poor representation of the reality. Some models are close to the reality: molecular structural models and molecular dynamic models are among these, as are some biophysical models of membranes. In general we can look at a sequential categorization of biological models that span the possibilities. At the most primitive level is the genome, the listing of the sequences of base pairs in the genes and chromosomes of the species from bacteria to humans. The genome is the model into which there is currently the largest expenditure of effort; but when it is defined, can we then define biology?

My categories of models are the genome, the morphonome, the physionome, the psychonome, the socionome, and the geonome. The morphonome is the description of structures: molecular, cellular, tissue, organ, and organism. The physionome is the quantitative description of the behavior of these structural elements in an integrated setting, the dynamics of the intact organism from cell to sentient being. The psychonome is the quantitative description of the psychological dynamics of the intact organism as an individual within its environment. The socionome is the description of behaviors of groups of individuals, of the nature of the interactions between social groups or societies. Finally, the geonome is the description of our environment, planet earth.

In this scheme of things, the contributions of our Resource Facility are minuscule. We attempt to deal only with a tiny piece of the phy-

continued on page 4

Three ODE solvers are available

Ordinary Differential Equations (ODEs) that describe the rate of change of a quantity with respect to time often express mathematical models of biological systems. Three ODE solvers have been implemented under our simulation interface, SIMCON, to permit users to formulate models as ODEs and use SIMCON to explore the behavior of the models and to fit the model output to experimental data.

These ODE solvers operate in interactive and batch modes, and allow the user to express a model in terms of up to 100 ODEs of the form dy/dt = f(t). Using a SIMCON flag, the user selects the solver to be used for the equations. This selection is based on the form of the equations and on the range of values of the coefficients (stiffness).

The solvers implemented are

1. LSODE, Livermore Solver for Ordinary Differential Equations. This solves ODEs for stiff and non-stiff systems. For stiff systems it treats the Jacobian matrix, df/dy, as either a full or a banded matrix and as either user-supplied or internally approximated by difference quotients. LSODE uses Adams predictor-corrector methods for non-stiff systems, and Backward Differentiation Formula (BDF) methods for stiff systems. Linear systems that arise are solved by direct methods (LU factor/solve).

2. LSODA, the Livermore Solver for Ordinary Differential equations with Automatic method switching. Otherwise similar to LSODE, this solver automatically selects between the non-stiff (Adams) and stiff (BDF) methods. Initially, it uses the non-stiff methods and dynamically monitors the data in order to determine if or when a shift of methods is appropriate.

3. RKF45, Fehlberg fourth order Runge-Kutta with fifth order correction time. This solves non-stiff and mildly-stiff differential

continued on page 2

Jim Bassingthwaighte
Director
206-685-2005
jbb@nsr.bioeng.washington.edu

Tom Bukowski
Lab Techniques, Data Analysis
206-685-2019
buko@nsr

Joseph Chan
Optimization, Numerics
206-543-5418
joseph@nsr

Harvey Friedman
Library, Code Standards
206-685-7433
fnharvey@nsr

Rick King
Facility Management
SIMCON Information
206-685-2007
rick@nsr

Keith Kroll
Modeling Analysis, Applications
206-685-2008
keith@nsr

Eric Lawson
Publications
206-685-2010
eric@nsr

Jim Ploger
Lab, Data Acquisition
206-685-2018
ploger@nsr

Gary Raymond
Application Programing
206-543-5417
garyr@nsr

Larry Weissman
System Management
206-685-2011
larryw@nsr

```
      SUBROUTINE  fdydt(neqn, t, y, ydot)
C
C neqn is the number of equations, t is the independent variable.
C y & ydot are vectors of the dependent variables and their derivatives.
      INTEGER     neqn
      REAL        t, y(neqn), ydot(neqn)
C
C upar is the vector of model parameters used to calculate the
C coefficients for the equations.
      REAL        upar(100)
      COMMON     /fdydtc/  upar
C
C Insert here statements to compute ydot in terms of t, y, and upar.
      RETURN
      END
```

Fig. 1. Template for subroutine to calculate derivatives.

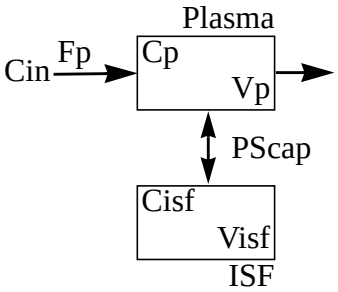


Fig. 2. Two compartment model with plasma and interstitial fluid, ISF, compartments. Fp, plasma flow; Cp, plasma concentration; Vp, plasma volume; Cisf, ISF concentration; Visf, ISF volume.

equations when derivative evaluations are not expensive.

In order to use the solvers, the user must supply a set of parameter values and a FORTRAN subroutine that calculates the derivatives. A template for the subroutine is given in Fig. 1.

As an example of setting up a solution, consider the two compartment system shown in Fig. 2 consisting of a plasma compartment and an interstitial fluid compartment.

The equations for the model in Fig. 2 are

$$\dot{C}_P = -\frac{F_p + PS_{cap}}{V_p}C_p + \frac{PS_{cap}}{V_p}C_{isf} + \frac{F_p}{V_p}C_{in}, \quad (1)$$

and

$$\dot{C}_{isf} = \frac{PS_{cap}}{V_{isf}}(C_p - C_{isf}) \quad , (2)$$

where $\dot{C}_p$, and $\dot{C}_{isf}$ are the derivatives of the concentrations in the plasma and ISF respectively, and the other symbols are those shown in Fig. 2.

The *fdydt* subroutine to solve the equations for this model is shown in Fig. 3. In this example, the dependent variables and their time derivatives are named *c* and *cdot*, respectively. The model uses four parameters from the *upar* array. For convenience, the FORTRAN equivalence statement is used to associate the names given in Fig. 2 to the locations used. As this model uses a forcing function, the first block of code calculates the value of the input function. Here a step function is used, but a function generator like *cinput* (a subroutine available from NSR) or a more complex scheme such as taking data from a curve in a reference data file could be used. The second block of code calculates the time derivatives. Since quantities such as flow and membrane PS products usually have units of ml/(g · min) and the SIMCON time step is in seconds, the volume is divided by 60 sec/min to give the correct units.

In order to use the solver, the user must also provide a SIMCON parameter array. The parameters used by the solver are given in Table 1. Parameter 1 specifies the solver to be used. In making the selection, the user must balance the needs of computational speed and solution accuracy. If the derivatives can be computed in a few arithmetic calculations (as in the example in Fig. 3) and the equations are, at most, mildly stiff, then RKF45 is an appropriate solver. If the equations are more complex but the stiffness is known, LSODE is an appropriate solver. Since it responds to the current condition of the equations, LSODA is the most robust solver, but it is significantly slower than LSODE and should only be used when the stiffness is not known.

Parameter 2 specifies the number of equations to be solved. Parameters 3 and 4 specify the error tolerances for the error tests. The recommended values of 0.00001 and 0.0, respectively, are chosen to ensure that the global solution has an error of less than 0.1%.

Parameter 5 indicates the stiffness of the equations. A set of equations is stiff if the ratio

```
      SUBROUTINE  fdydt(neqn, t, c, cdot)
C
C ODE for a 2 compartment model. Equations are:
C    Cdot(1)  = -C(1)*(Fp+PScap)/Vp + C(2)*(PScap/Vp) + Cin*(Fp/Vp)
C    Cdot(2)  = (C(1)-C(2))*(PScap/Visf)
C
      INTEGER     neqn
      REAL        t, c(2), cdot(2)
      REAL        upar(100), fp, vp, pscap, visf, cin
      COMMON     /fdydtc/  upar
      EQUIVALENCE (upar(1), fp), (upar(2), vp), (upar(3),pscap)
      EQUIVALENCE (upar(4), visf)
C
C Calculate the input function for the current time.
      cin = 1.0
      IF (t. LE. 0.0 .OR. t .GT. 1.0) cin = 0.0
C
C Calculate the time derivatives.
      cdot(1) = (vp/60.0)*(-(fp+pscap)*c(1) + pscap*c(2) + fp*cin)
      cdot(2) = (visf/60.0)*pscap*(c(1)-c(2))
      RETURN
      END
```

Fig. 3. Subroutine to calculate derivatives for a two compartment model. The input function, cin, is a step function with unity area.

of its maximum eigenvalue to its minimum eigenvalue is larger than 10. If the ratio is between 2 and 10, the equations are mildly stiff. If the user does not know the stiffness, the LSODA solver should be used as it examines the equations and automatically selects the appropriate method of solution.

P-array 301-400 gives the initial conditions of the dependent variables, and P-array 201-300 gives the time course of the dependent variables (i.e., the results). P-array 401-500 is the basis for calculation of coefficients of the equations. The values placed in these locations are copied into the *upar* vector of the *fdydtc* common block and, thus, are available for use in the *fdydt* subroutine. Parameter 401 goes to *upar(1)*, 402 to *upar(2)*, etc.

P-array index	Name	Usage
1	mode	Integrator selector: 1 = LSODE, 2 = LSODA, 3 = RKF45
2	neqn	Number of differential equations.
3	rtol	Relative error tolerance for the local error test. (Recommended value is 0.00001.)
4	atol	Absolute error tolerance for the local error test. (Recommended value is 0.0.)
5	stiff	Stiffness indicator: 0 = non-stiff, 1 = stiff.
201-300	yi(t)	Value of y(i) at the current time, i = 1, ... , 100.
301-400	yi(t=0)	Initial condition for y(i) at time = 0, i = 1, ... , 100.
401-500	upar(i)	Model parameters for the user model, i = 1, ... , 100. These are copied to the *upar* vector in the *fdydtc* common block.

Table 1: Usage of the SIMCON P-array by the interactive version of the ODE solver.

To create an executable program of a SIMCON model using the ODE solver package, the user supplied derivative routine is linked together with the SIMCON program, the ODE library, and the additional libraries required by SIMCON and the ODE solver. If the user-supplied derivative routine is in a module named *fdydt.f*, the command to compile the routine and create an executable program named *2comp* is

```
f77 -o 2comp fdydt.f \
  /usr/local/lib/simcon/simcon.o \
  -lsimode -lsim -lnsr -lcm
```

A SIMCON ODE program, like all SIMCON programs, is presently restricted to a maximum of four parameters optimized simultaneously via the automated SIMCON optimizer. A batch version of the ODE solver is also available. This allows simultaneous optimization of up to thirty parameters in noninteractive mode. In a batch program the user supplies the same routine to calculated derivatives, *fdydt*, and a SIMCON parameter array. In this case, however, P-array locations 9 through 99 specify the parameters to be optimized. The usage of these parameters is detailed in Table 2. When using the batch version, the values in p(9)-p(99) take precedence over the locations normally used by SIMCON to specify the parameters to be optimized, p(182)-p(194). (An exception to this is the case when p(9) is set to 0 as shown in Table 2.) The P-array locations used for stopping criteria, p(132)-p(135), the location of the model output, the index of the reference data curve, and to specify data weighting, p(195)-p(198), are used the same way by both the batch and interactive versions.

To create an executable program using the batch version of the ODE solver package, the derivative routine is linked to the batch ODE library and additional libraries required by the solver. If the user-supplied derivative routine is in a module named *fdydt.f*, the command to compile the routine and create an executable program named *b2comp* is

```
f77 -o b2comp fdydt.f -lfitode \
  -lsim -lnsr -lcm
```

After the program is created, the syntax of the command to execute it is

```
b2comp [-pf parfile] [-df datfile]
```

where the optional arguments *parfile* and *datfile* name files containing the parameter values and reference data, respectively. If no parameter value file is specified, the file *simcon.par* is used. If no reference data file name is specified, the name specified in the parameter value file is used. Note that the format of the parameter value file is that used by SIMCON when the user saves a run. Thus, the recommended method for making a batch ODE program and its parameter and data files is to first make an interactive version that is used for debugging and testing. When the user is satisfied that the derivative-calculating routine, the parameter values, and the data are correct, SIMCON is used to save the parameters and data. These SIMCON output files are then used as input to the batch version.

— Joseph Chan, Rick King

P-array index	Name	Usage
9	Nopt	Number of parameters to be optimized. If zero, the parameters normally used by SIMCON, p(182) - p(194), specify the parameters to be optimized.
10-39	Pinx(i)	The P-array index of the 400 + i^{th} P-array to be optimized.
40-69	Pmin(i)	The minimum allowable value of the i^{th} parameter to be optimized.
70-99	Pmax(i)	The maximum allowable value of the i^{th} parameter to be optimized

Table 2: Usage of the SIMCON P-array by the batch version of the ODE solver. Note that these locations are used in addition to those used by the interactive version (see Table 1).

"Where?" from page 1

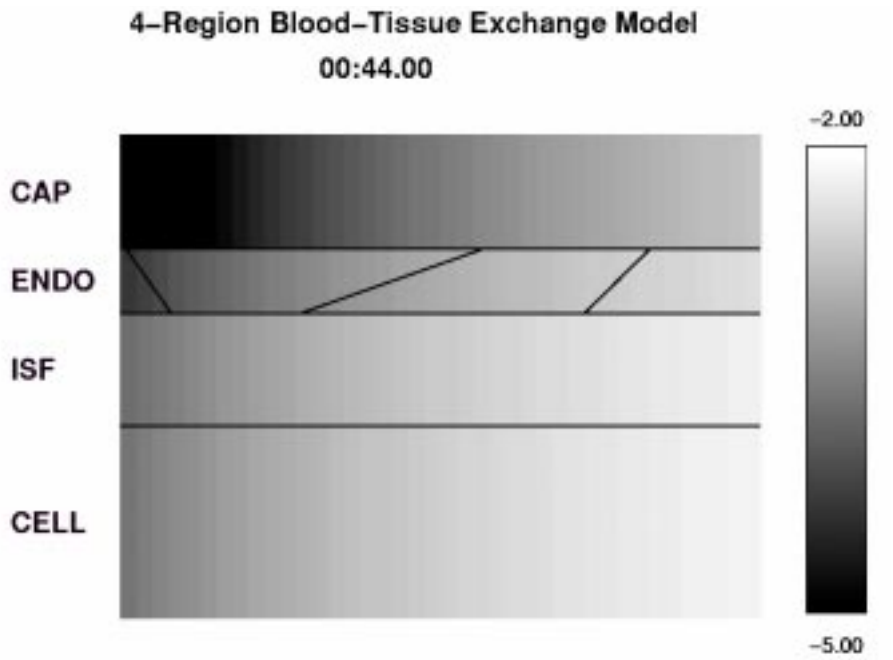
sionome, the kinetics of the exchanges of molecules across the membranes of the tissues of the body and the flows that carry solutes from organ to organ and tissue to tissue, and of a few of the reactions that molecules undergo with the tissues and cells. Even so we feel proud that our contributions have some originality and are useful to others in understanding physiology at an ever deeper level. Our models are obviously reductionist, each being a simplification of reality. And yet each improvement brings us a little closer to reality. One to one relationships, between our models and carrier proteins or channels, or membranes, or cells, or capillary-tissue units, etc., is not practical or sensible.

What we can do and are doing is developing strategies that capture the essence of the behavior of the individual functional units of a system, and to begin to account for their heterogeneities. This is where fractals come in. Fractals have the virtue of representing the recursive nature of the growth processes leading to the structures, and of defining the correlations over space amongst natural structures. They also appear to characterize the temporal correlations in the dynamics of our chaotic processes, the natural fluctuations in cellular and organ functions that are the natural consequence of having complex, multiply redundant control systems operating at low gain and without set points. Thus the new modeling efforts in the resource tend toward the nonlinear systems of carrier-mediated transport, the complexities of enzymatically regulated reactions, and their integration into manageable models.

— Jim Bassingthwaighte

New SIMCON graphics capabilities

The development of the new point-and-click simulation interface (see our companion article on XSIM) relies on development of more meaningful ways to display simulation output. The foundation for this latter development was laid in the summer of 1988, when Simulation Resource programmer Aaron Joseph implemented a library of graphical objects (independent graphical representations of simulation data) that could display simulation output dynamically on a display connected to a UNIX minicomputer. That computer and its display are gone, but the graphical objects have been ported to the X11 window system. Currently these objects are called from models that run under the command-driven SIMCON interface. Eventually, they will become an integral part of the point-and-click XSIM interface.



Enhanced display of a four region blood–tissue exchange model. The central panel shows the axial concentrations in the capillary, endothelial, interstitial fluid, and parenchymal cell regions. Flow is from left to right through the capillary. The scale, shown on the bar to the right, is logarithmic with the maximum being 10^{-2} and the minimum 10^{-5} g/mL. The current simulation time in seconds appears below the title.

The most elaborate of the graphical objects displays the state of metabolites distributed within a single BTEX (blood tissue exchange unit) panel. Other objects in the library are a color bar, line graph, text block, clock, and image grid. The image grid displays a rectangular area of chemical concentrations that can be updated dynamically.

Multiple independent instances of each object can be displayed simultaneously. In the oxygen-water exchange model, for example, one can display separate BTEX panels for oxygen and water, along with color bars that show the correspondence between color and concentration. The user can watch oxygen enter the capillary, diffuse through the endothelium and interstitial fluid to the parenchymal cells in one panel, while the second BTEX panel displays the generation of water in the parenchyma and its subsequent washout to the capillaries. At the same time, the output of oxygen and water from the capillary bed are plotted in a line graph object in another corner of the screen.

Implementing the graphical object library presented some technical challenges. For example, the SIMCON interface is driven by line oriented commands rather than by keyboard or mouse "events." Hence the graphical component requires a separate graphics process that receives X11 requests from SIMCON through a UNIX pipe. This process is started automati-

cally when graphics are first drawn. Further, all X11 applications must be able to handle screen redrawing caused by resizing and restacking windows. The implementation handles these events in a fast, memory-efficient manner.

The graphical object library will evolve in several ways. Additional objects will be added to represent three-dimensional data (contour plots, mesh plots) and more elaborate exchange units derived from the basic four-region BTEX. A major consideration in developing these objects is that they will be the sole means of representing simulation output in XSIM, so we expect that little of the code will have to be redone. Since XSIM will be a canonical event-driven X11 program, the separate graphics process and pipe will not be needed.

Implementation Details: The current graphical object library will run on any X11 server, although color or grey scale is required to properly display some objects (BTEX panel, color bar, image grid). The graphical process uses only functions found in Xlib, the lowest level programming interface to the X11 package, or the intrinsics library (Xt). The SIMCON interface and models do not require linking to any X11 library. All of the new graphical code is written in C, but the graphical objects routines are Fortran-callable to allow them to be invoked from the simulation models. Several reference implementations are available to serve as templates for adapting existing models.

— Larry Weissman

XSIM: An X11-based simulation interface

XSIM, the next generation simulation interface of NSR, is well under way. Our present interface, SIMCON, was designed in the 1980's for character-oriented terminals capable of simple line graphics: XSIM will run on X11 screens and use the popular "point and click" approach. Other useful features are planned, and some of them will be back-ported to SIMCON. Among these features are the ability to change models in midsession, and support for non-numeric elements in the parameter database.

The main design decisions for XSIM center on how the screen will look at different times as the user navigates through groups of simulation parameters. One complication is that part of the screen's appearance is determined by the interface and the rest by the currently selected model. In order to simplify the development of models, the appearance of the screen will be determined by configuration files consisting of plain text that might look something like this:

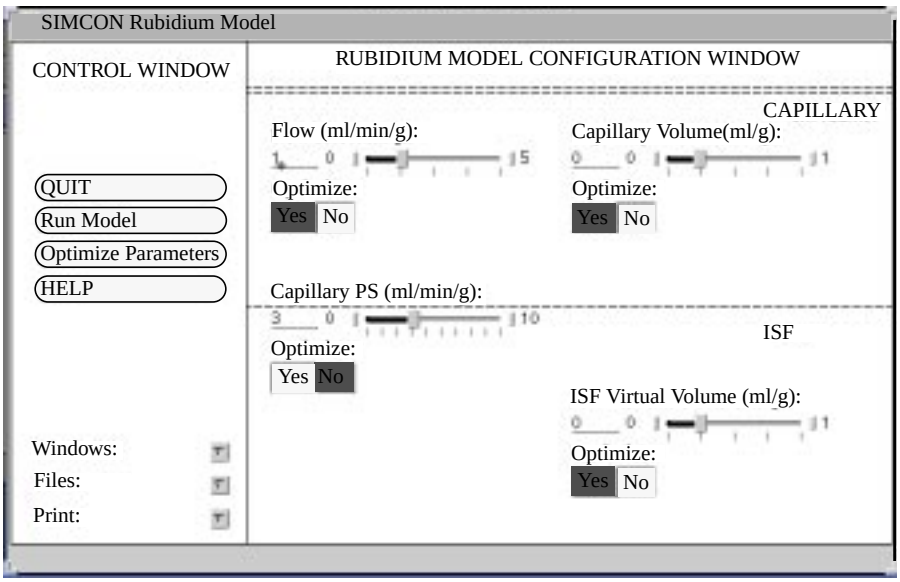
```
define_parameter {
    name            = 'Initial time',
    type            = real,
    p_array_offset  = 128,
    initial_value   = loaded,
    optimize        = no,
    lower_bound     = 0,
    window          = 'time_group',
    window_llx      = 400,
    window_lly      = 150,
    …
}
```

This type of configuration file allows programmers to easily change the screen's appearance without having to deal with the intricacies of the X11 window system.

An important by-product (and difference from the SIMCON interface) is that only the significant database parameters will be accessible. In fact, the concept of the "P-array" could become obsolete for model users, since it is only an internal detail of the implementation.

The first version of XSIM will use the XView toolkit to implement the graphical interface. Although XView was developed at Sun Microsystems, it has been made available in the generic MIT X11 distribution for non-Sun platforms. Eventually the graphical interface of XSIM will be built with other toolkits to achieve wider distribution. A prototype of XSIM should be available by summer, 1992.

— Larry Weissman, Rick King



Sample XSIM window. On the left is the simulation control window. Functions such as running the model are controlled by buttons while selecting more complex functions, such as file manipulation, gives a pop-up window. On the right, the model configuration window gives an overview of the model. Parameter values can be entered directly or, optionally, controlled by sliders. Clicking on a parameter name gives a pop-up window allowing detailed control of the parameter.

MMID4 update

MMID4, the multiple path, multiple indicator dilution model, models three tracers: a vascular tracer that is restricted to the vascular space, an extracellular tracer that passes through inter-endothelial cell gaps in the capillary walls and enters the interstitial fluid space (ISF), and a permeant tracer that also crosses cell membranes and enters endothelial and parenchymal cells as well as the ISF. The twenty flow pathways of MMID4 have operators to model a non-exchanging vessel (arteriole/venule) and the capillary-tissue unit. Each pathway has a different flow in order to model the flow heterogeneity that exists even in normal organs.

A revision of MMID4 is scheduled for release in late spring. Major changes will be the addition of more nonexchanging vessel operators and more options for modeling heterogeneity of flow. Additionally, the program will give the user more flexibility in summing and scaling model outputs and in linking parameters.

Nonexchanging vessels

MMID4 now contains a single large nonexchanging vessel that feeds all the flow paths. Each path contains a medium-sized nonexchanging vessel in addition to the capillary–tissue unit. The large vessel represents the combination of arteries and veins, and the medium-sized vessels the arterioles and venules.

Additional nonexchanging vessel operators will be added to model inflow and outflow tubes, arteries, arterioles, venules, and veins; the new operators will give MMID4 greater flexibility and applicability to more experimental conditions. Fig. 1 shows the new arrangement. Input and output tubing are included to represent tubing connected to isolated organs; they may also be used for arteries and veins lying between the injection and sampling sites, but not in the "field of view" for residue function calculations. If the user does not want to use a specific nonexchanging vessel, giving it a zero volume effectively removes it from the model.

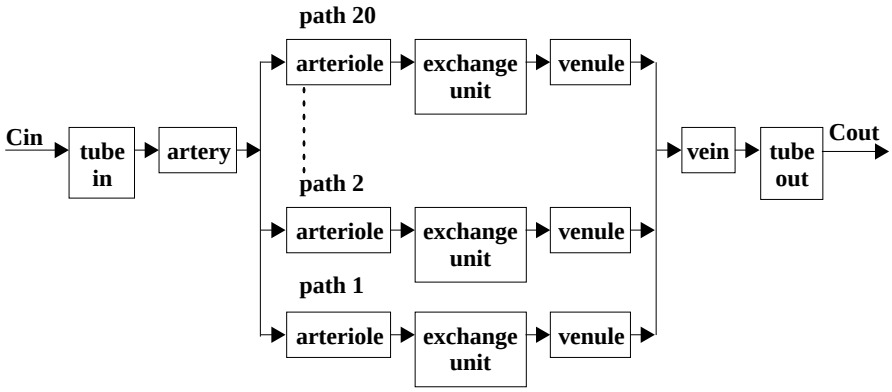


Fig. 1. Structure of vessels in the new version of the MMID4 model.

The parameters for the operator that models a nonexchanging vessel have been modified. In the current version of MMID4, users specify a volume and the pure delay fraction of the operator. In the new version, users will specify the volume and the relative dispersion of the vessel.

Modeling heterogeneity of flow

Modeling of heterogeneity will be separated into two distinct parts: (1) specifying the probability density function (PDF) of fractional mass, $w(f_i)$, versus relative flow, f_i, and (2) selecting the relative flows that will be used for the pathway in the model.

To specify the PDF, users will be able to select a standardized function, either a lagged-normal or a random walk density function, or enter an arbitrary density function. The latter may come from a variety of sources such as measurements made with radioactive microspheres. In each case, the function is normalized to insure mass conservation in the model.

Permitting the user to enter the relative flow for each path used in the modeling, or to instruct MMID4 to select the relative flows, will be a new capability. In the case where MMID4 selects the flows, the user will control whether the flows are equally spaced in the flow domain, equally spaced in the transit time domain, or at some intermediate between these extremes. Relative flows equally spaced in the flow domain often best preserve the overall statistics of the PDF, but under-represent the slow flow paths and deform the tail of the output concentration-time curve. Equal transit time intervals typically give greater skewness than desired, but give a more accurate representation of the output curve when the data extend to times for which the long transit time paths have a significant contribution to the output. For each method of selecting the relative flows, the program will attempt to pre-

serve the shape of the PDF, but is constrained by the requirement of preserving the mean and area of unity. Fig. 2 shows the representation of an underlying PDF and three sets of relative flows chosen by MMID4 to represent it.

Summing and scaling model outputs

In the installed MMID4, there is an operator to sum up to six model outputs and one to scale a single model output. In the next version, these will be combined into a new operator. This new operator can be written as

$$\mathrm{SUM} = K_0 \cdot (K_1 \cdot \mathrm{Out}_1 + K_2 \cdot \mathrm{Out}_2 +$$

Three of these new operators will be included.

Linking parameters

In the installed MMID4, parameters can be link so that a change in one is automatically reflected in a change of the other. For example, the user can specify that the permeability-surface area product on the lumenal side of the endothelial cell always be equal to that on the ablumenal side (i.e. $PS_{eca} = PS_{ecl}$). This results in one fewer model parameter that the user needs to set. Several users suggested that the linker would be more useful if it included a proportionality constant. Thus, the new parameter linker will handle linkages in the form

SIMCON update

The current revision level of SIMCON is 2.6.0. This revision fixes a number of bugs and includes four minor modifications.

Modifications/Enhancements

- The default menu style is 'long' and the default edit style is 'screen mode.'
- The maximum number of parameters in a parameter group is increased from 40 to 120.
- The units of sensitivity are now percent change in the solution per percent change in the parameter value.
- The display of values of parameters being modified by loops has been altered to work correctly on nonerasable displays (e.g., xterms).

Bugs Corrected

- Sensitivity calculations do not fail when the initial value of the parameter is 0.0.
- Submenus now refresh properly after model and optimization runs.
- The sensitivity calculation option now works when the menu style is 'short' or 'medium'.
- Database menu options 's' and 'n' now operate correctly.
- The 'run-model-after-optimization' configuration parameter now has the proper effect from sub-menus and the parameter editor.
- If model errors are generated when there is insufficient space to print the messages, the user is directed to return to the main menu to get the messages.
- Reference data can be plotted in area 2 when no simulation results are plotted in that area.
- When a sensitivity run is killed, the values of the parameters are restored correctly.
- The hardcopy device type is read properly from the parameter set and is not reset to zero by starting a SIMCON session.
- When plotting reference data using lines, wrap-around is now handled correctly.
- Plot codes, p(146)-p(153), are now stored correctly in simcon.par files when all seven digits of the code are used.

Release 5.0 of the SIMCON User Guide is nearly complete and will be available by 19 April. This release will reflect changes in versions 2.4.0 through 2.6.0 of SIMCON.

Remote users may request code or documentation updates by sending electronic mail to simcon@nsr.bioeng.washington.edu. Documentation requests should include the release number from Page a of the User Guide of your manual.

— Rick King

Please help us get this newsletter to those who want it by completing and returning the form below (our address is on the reverse side). Thanks!

☐ My name is _____. Remove me from the newsletter mailing list—I no longer want to get it.

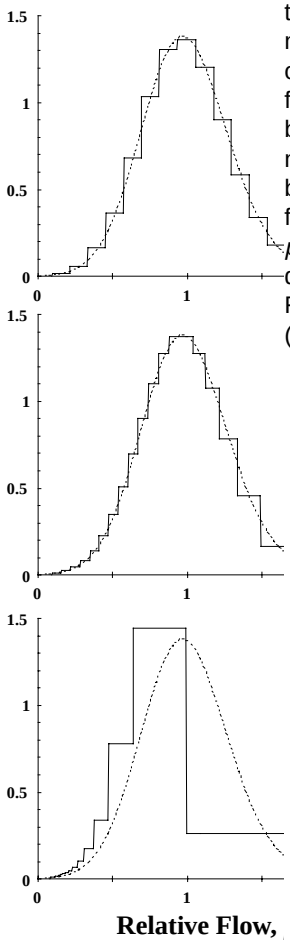
☐ Add the following to the newsletter mailing list:

Name:
Address:

Name:
Address:

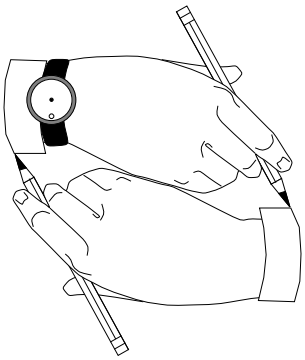
$$P_n = K \bullet P_m$$

That is, the product of K times P_m is loaded into P_n. When the scalar K is set to 1, the linker will function as in the current MMID4.



— *Gary Raymond,*
Rick King

Fig. 2. Relative flows used by MMID4 for a 20 pathway model. The model used for the flow distribution (shown by the dotted lines) is a lagged normal distribution with RD of 0.3 and skewness of 0.3. The panels show three different options for selecting the flows used by the model (shown by the solid line histograms). The RD and skewness of the flows used by MMID4 are given below. *Top panel*: Flows equally spaced in the flow domain (RD=0.3, skewness=0.29); *Bottom panel*: Flows equally spaced in the transit time domain (RD=0.46, skewness=1.0)*; Middle panel*: Flows interpolated between the two extremes (RD=0.31, skewness=0.56).



Dr. Simcon

Dear Doctor,

I keep trying to get a hardcopy of my SIMCON plot by pressing 'h' at the end of my simulation run, but nothing ever shows up on the printer. What's a body to do?!

Copyless

Dear Copyless,

Sounds to me like you don't have your 'HARDCOPY DEVICE CONTROLLER' set properly. The controller is p(139). If it's set to 0, you won't get a hardcopy no matter how hard you try. If the controller is set to 1, the plot is sent to the printer; if set to 2, the plot goes to the plot file, simcon.pos. (See Section 7.6 of the User Guide.)

Dr. Simcon

P.S. There is a bug in the current version that sometime causes the controller to be reset to 0. Check its value whenever you are having trouble.

Dear Doctor SIMCON,

I'm running SIMCON under OpenWindows. Everything seems to work OK except that I can't get any graphics output. What's up, Doc?

Graphless

Dear Graphless,

There are windows, and, then again, there are windows. You need to be careful what kind of window you are in. In order to get graphics, you need to be in an xterm window. You can't execute the program from a Command Tool or Shell Tool window and expect to get any graphics. There are a couple of ways to get a window in which you can get graphics. The preferred method is to use the command "simwin.ow2" to run the model. (See the reply to Confused below for more details.) Alternately, use the command "xterm &" to open a new xterm window; you can execute the model from this window and get graphical output.

Dr. Simcon

Dear Doctor,

I always run SIMCON models by entering their name. My neighbor uses some command named "simwin." What's this all about?

Confused

Dear Confused,

Simwin is a command that we have set up for X-window users to help them manage windows when they run a SIMCON program. If you are not on an X-windows device, don't use *simwin*, but if you are I recommend that you do. What *simwin* does is to open a new window for your model that has a graphics window that is properly sized and located so that it doesn't obscure the text window that contains your menu and command line. When you quit the program, the text and graphics windows are closed. (There is also a version that's tailored for OpenWindows. Its name is *simwin.ow2*.) The syntax of the simwin command is:

```
simwin {program} [SIMCON options]
```

where {program} is the name of the SIMCON program you want to run and [SIMCON options] are optional.

There is a problem with using *simwin*. If the program crashes and writes error messages, the window closes so fast that you won't be able to read the messages. (Gee, am I embarrassed about this one!) The work-around is to rerun the program from a regular xterm window so that you can read the messages.

Dr. Simcon

P.S. There is, of course, an on-line manual page for *simwin* that gives all the information you need. Oops, no man page for *simwin.ow2*, but all the information for *simwin* applies.

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

CENTER FOR BIOENGINEERING, WD-12
UNIVERSITY OF WASHINGTON
SEATTLE, WASHINGTON 98195