Simulation Languages for PBPK Modeling

Interpretation of biomonitoring data using physiologically based pharmacokinetic modeling

Center for Human Health Assessment
September 25-29, 2006
Numerical Integration Algorithms: “Inching along in the Dark”

Mathematically, PBPK Models represent a system of ordinary differential equations (ODEs) defining an initial value problem.

A number of simulation languages are available to solve this type of problem numerically.

The approach generally used to solve ODE initial value problems is referred to as “numerical integration”.
Numerical Integration Algorithms: “Inching along in the Dark”

\[ Y(t + \delta) = Y(t) + Y' \cdot \delta \]
Numerical Integration Algorithms: “Inching along in the Dark”

The larger the step size, the larger the potential for systematic error.
Numerical Integration Algorithms: “Inching along in the Dark”

Automatic step size adjustment based on error tolerance
PBPK models with widely varying compartment time constants behave as stiff systems, fooling automatic step-size algorithms.

Comparison of the performance of a stiff (A) and nonstiff (B) ODE solver for a problem exhibiting steady-state behavior (broken line). The time step of the nonstiff program is severely constrained during the steady-state period. The nonstiff program had about 75 times as many steps and took about 50 times as long to solve the problem.

Stiff ODE solvers:
- Gear (LSODES)
- Rosenbrock
Popular Simulation Languages for PBPK Modeling

acsI Xtreme: AEgis Technologies Group, Inc.
http://www.aegistg.com

Berkeley Madonna: University of California at Berkeley
http://www.berkeleymadonna.com

MATLAB/Simulink: The MathWorks, Inc.
http://www.mathworks.com

ModelMaker: ModelKinetix
http://www.modelkinetix.com
## LANGUAGE COMPARISON 1: BASIC FEATURES

<table>
<thead>
<tr>
<th>“Feature”</th>
<th>ACSL</th>
<th>acslXtreme</th>
<th>MATLAB</th>
<th>MATLAB/SIMULINK</th>
<th>Model Maker</th>
<th>SCoP</th>
<th>Berkeley Madonna</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphical interface</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Equation interface</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Built-in integration algorithm</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Discrete Blocks</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Scripting</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y *</td>
<td>N +</td>
</tr>
<tr>
<td>Code sorting</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>n/a</td>
<td>n/a</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

* Menu keystroke list
+ Under development
Example of Statement Order in Unsorted Code

function dy = sodes(tspan,y0)
%MATLAB styrene PBPK model ODE definition file

global c e %Make constants and variables externally available

%Transfer initial condition information to state variables
e.at = y0(1);
e.af = y0(2);
e.am = y0(3);
e.al = y0(4);
e.adose = y0(5);

%Styrene in tissue (rest of body) compartment
e.ct = e.at/e.vt; % (mg/l)
e.cvt = e.ct/c.pt; % tissue venous blood (mg/l)

%Styrene in fat compartment
e.cf = e.af/e.vf; % (mg/l)
e.cvf = e.cf/c.pf; % Fat venous blood (mg/l)
%Styrene in liver compartment
\[ e_{cl} = \frac{e_{al}}{e_{vl}} \]  \(\text{%(mg/l)}\)
\[ e_{cvl} = \frac{e_{cl}}{c_{pl}} \]  \(\%\text{Liver venous blood (mg/l)}\)

%Styrene in blood
\[ e_{cv} = \frac{(e_{qf} * e_{cvf} + e_{ql} * e_{cvl} + e_{qt} * e_{cvt})}{c_{qc}} \]  \(\%\text{Mixed venous (mg/l)}\)
\[ e_{ca} = \frac{(c_{qc} * e_{cv} + c_{qp} * e_{ci})}{(c_{qc} + (c_{qp} / c_{pb}))} \]  \(\%\text{Arterial (mg/l)}\)

%Exhaled styrene
\[ e_{cx} = \frac{e_{ca}}{c_{pb}} \]  \(\%\text{Alveolar (mg/l)}\)

%Differential equations
\[ e_{rat} = e_{qt} * (e_{ca} - e_{cvt}) \]  \(\%\text{Tissue - "rest of body" (mg/hr)}\)
\[ e_{raf} = e_{qf} * (e_{ca} - e_{cvf}) \]  \(\%\text{Fat (mg/hr)}\)
\[ e_{ram} = c_{vmax} * e_{cvl} / (c_{km} + e_{cvl}) \]  \(\%\text{Metabolism (mg/hr)}\)
\[ e_{ral} = e_{ql} * (e_{ca} - e_{cvt}) - e_{ram} \]  \(\%\text{Liver (mg/hr)}\)
\[ e_{rdose} = c_{qp} * (e_{ci} - e_{cx}) \]  \(\%\text{Net absorption (mg/hr)}\)

\[ dy = [e_{rat}; e_{raf}; e_{ram}; e_{ral}; e_{rdose}] \]  \(\%\text{For use by ODE solver}\)
Calling the Integration Algorithm

while hours < c.tstop
   %Keep track of time
   hoursold = hours; hours = hours + e.cint;
   days = hours/24; hod = hours-floor(days)*24;    %hour of day
   weeks = days/7; dow = (hours - floor(weeks)*168.)/24;    %day of week 1 - 7

   %Turn exposure on or off
   if hod >= c.offset & hod <= c.offset + c.length & dow <= 5    %expose 5 days per week
      e.ci = c.conc*c.mw/24450.;            %Exposure concentration (mg/l)
   else e.ci = 0.; end

   %Numerical integration
   tspan = [hoursold hours];                 %interval over which model is integrated
   y0 = [e.at; e.af; e.am; e.al; e.adose];   %initial conditions for the state variables
   [T,Y] = ode15s('sodes',tspan,y0);         "%sodes.m" contains the model's equations

   %Save data for plotting
   counter = counter + 1;
   data(1,counter) = hours; data(2,counter) = e.at; data(3,counter) = e.af;
   data(4,counter) = e.am; data(5,counter) = e.al; data(6,counter) = e.cv;
end
# LANGUAGE COMPARISON 2: COST/SPEED

<table>
<thead>
<tr>
<th>“Feature”</th>
<th>ACSL</th>
<th>acslXtreme</th>
<th>MATLAB</th>
<th>MATLAB/ SIMULINK</th>
<th>Model Maker</th>
<th>SCoP</th>
<th>Berkeley Madonna</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost *</td>
<td>n/a</td>
<td>$12,000</td>
<td>$4,000</td>
<td>$6,000</td>
<td>$360</td>
<td>$150</td>
<td>$300</td>
</tr>
<tr>
<td>Compiled (faster)</td>
<td>Y</td>
<td>Y</td>
<td>Y +</td>
<td>N*</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Interpreted (more convenient)</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y*</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

* Partially compiled
+ Extra cost

- Student versions
- Academic discounts
## LANGUAGE COMPARISON 3: ADVANCED FEATURES

<table>
<thead>
<tr>
<th>“Feature”</th>
<th>ACSL</th>
<th>acslXtreme</th>
<th>MATLAB</th>
<th>MATLAB/SIMULINK</th>
<th>Model Maker</th>
<th>SCoP</th>
<th>Berkeley Madonna</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization</td>
<td>Y+</td>
<td>Y</td>
<td>Y+*</td>
<td>Y+*</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Sensitivity analysis</td>
<td>Y*</td>
<td>Y</td>
<td>Y*</td>
<td>Y*</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Monte Carlo</td>
<td>Y*</td>
<td>Y*</td>
<td>Y*</td>
<td>Y*</td>
<td>Y</td>
<td>N</td>
<td>Y*</td>
</tr>
<tr>
<td>Units checking</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Database of physiological values</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

* User supplied/developed
+ Extra cost
Summary: Simulation Languages for PBPK Modeling

ACSL / SCoP: popular in past
no longer supported / upgraded

acsI Xtreme: powerful, expensive
becoming more MATLAB-like

Berkeley Madonna: fast, fun, flexible
scripting under development

MATLAB/Simulink: powerful, labor intensive

ModelMaker: good Monte Carlo front-end
no scripting