Dynamic Modeling Using MCSim and R

R. Woodrow Setzer
National Center for Computational Toxicology
US Environmental Protection Agency
RTP, NC

setzer.woodrow@epa.gov
http://orcid.org/0000-0002-6709-9186

SOT Biological Modeling Specialty Section
Webinar
4 February 2016

The views expressed in this presentation are those of the authors and do not necessarily represent the views or policies of the US Environmental Protection Agency
GNU MCSim

Frederic Y. Bois

R Core Team

Solving Differential Equations in R: Package deSolve

Karline Soetaert
Netherlands Institute of Ecology

Thomas Petzoldt
Technische Universität Dresden

R. Woodrow Setzer
US Environmental Protection Agency
• **Advantages to MCSim and R**
  – Very flexible and fast
  – Supports the full model-development cycle
  – R packages facilitate
  – Coding in domain-appropriate languages:
    • MCSim for dynamic (e.g., PBPK) models
      – SBML possible
    • R for
      – Simulation
      – Graphics
      – Statistics (summarization, estimation)
      – Sensitivity analysis (local and global)
  – Free, Open Source software with support:
    • Mailing lists for general R support
    • Revolution R for commercial R support
    • R-sig-dyn-mod mailing list for dynamic modeling support in R
    • Help-mcsim mailing list for MCSim help
• **Disadvantages to MCSim and R**
  – Requires coding: similar to ACSL-X, Matlab, in contrast to, *e.g.*, SimuLink
  – Languages are general-purpose: not specific to PBPK, for instance.
  – Somewhat difficult learning curve
  – Installation is more “hands-on” than commercial programs
    • requires installing multiple programs and packages
    • some extra steps in Windows
      – Install Rtools, plus some extra scripts for running MCSim with R
      – May require work-arounds for some security setups in an enterprise environment on Windows (but, everything can be installed in user space, *e.g.*, in Windows in C:\Users\wsetzer\Applications)
      – Requires some manual setting of environment variables
Modeling Workflow Steps

MCSim Model

## R code
Translate, compile and load model

## R code
Define parameters and run model (wrapper?)

## R code
• Graphics
• Optimization
• Sensitivity Analysis
• ...

> ...
# perc.model
# A four-compartment model of Tetrachloroethylene (PERC) toxicokinetics.
# Copyright (c) 1993-2008 Free Software Foundation, Inc.
# Distributed with MCSim
#

# States are quantities of PERC and metabolite formed, they can be output
States = {Q_fat, # Quantity of PERC in the fat (mg)
         ...
         Q_met}; # Quantity of metabolite formed (mg)

Outputs = {C_alv, # mg/l in the alveolar air
           ...
           C_exh_ug}; # ug/l in the exhaled air

Inputs = {C_inh, # Concentration inhaled (ppm)
          Q_ing}; # Quantity ingested (mg)

# Constants
# Exposure modeling
# ---------------

InhMag = 0.0; # inhaled concentration
Period = 0.0; # period of the exposure/no exposure cycle

Exposure = 0.0; # exposure duration within a period

C_inh = PerDose (InhMag, Period, 0.0, Exposure);
# Scaled parameters

BodyWt = 0;

V_fat = 0; # Actual volume of tissues

...Initialize {
    BodyWt = LeanBodyWt / (1 - Pct_M_fat);
    V_fat = Pct_M_fat * BodyWt/0.92; # density of fat = 0.92 g/ml

...}

} # End of model initialization

Dynamics {

    Cout_fat = Q_fat / (V_fat * PC_fat);
    C_alv = C_art / PC_art;
    Cout_wp = Q_wp / (V_wp * PC_wp);
    C_art = (Flow_alv * C_inh / PPM_per_mg_per_l + dQ_ven) / (Flow_tot + Flow_alv / PC_art);

...}

dt (Q_exh) = Flow_alv * C_alv;

dt (Q_fat) = Flow_fat * (C_art - Cout_fat);

dt (Q_wp) = Flow_wp * (C_art - Cout_wp);

} # End of Dynamics

Parameters declared as functions of other parameters. First declare as constant with dummy value, then define in Initialize

Dynamics where auxiliary variables (e.g. C_alv) and derivatives (w dt() operator) are defined. Auxiliary variables can be saved for view by declaring the in Outputs. Order matters.
CalcOutputs {

    # Liver concentration
    C_liv = Q_liv / V_liv;

    # Fraction of TCE metabolized per day
    Pct_metabolized = (InhMag ?
        100 * Q_met /
        (1440 * Flow_alv * InhMag * mg_per_l_per_PPM) : 0);

    C_exh_ug = C_exh * 1000;  # milli to micrograms

} # End of output calculation

Additional outputs, functions of parameters and state variables.

You can also combine a template describing the basic PK structure, and combine it
with SBML (Systems Biology Markup Language) molecular models for biochemical
interactions occurring within PK compartments. See Cheng and Bois, *EHP* 119 (12):
1712 – 1718 (2011), and example code in supplementary materials (available from
GNU MCSim website).
Get the Model Ready to Run in R

```r
library(deSolve)

mName <- "perc.model"

system(paste("mod.exe -R ", mName, " ", mName, ".c", sep = ""))

# creates perc.model.c and perc.model_inits.R

system(paste("R CMD SHLIB ", mName, ".c", sep = ""))

dyn.load(paste(mName, .Platform$dynlib.ext, sep=""))

source("perc.model_inits.R")
```

## R script, or at R command line:

- **Load deSolve package**
- **Translate MCSim code into c and R**
- **Compile and load the model into the R process**

# defines initParms(), for initializing the parameter vector;
# initStates(), for initializing the state vector;
# and Outputs, the vector of names of output values.
times = c(seq(0,280, by=1),seq(281, 1*24*60, by=30))

parms <- initParms(newParms=list(InhMag=144, Period=max(times), Exposure=240))
Y <- initStates(parms)

## Forcing Setup (save this function for reuse in other models)
PerDose <- function(mag, Period, start, ExpDuration, times) {
    Nrep <- ceiling(max(times) / Period)
    times <- rep(c(start, ExpDuration), Nrep) +
        rep(Period * (0:(Nrep - 1)), rep(2, Nrep))
    y <- rep(c(mag,0), Nrep)
    cbind(times, y)
}
Forc <- list(PerDose(parms["InhMag"], parms["Period"], 0,
        parms["Exposure"], times),
        PerDose(0, max(times), 0, 240, times))

out <- ode(Y, times, func = "derivs", parms = parms,
        dllname = mName,
        initforc = "initforc",
        forcing= Forc,
        fcontrol=list(method="constant", rule=2,f=0),
        initfunc = "initmod", nout = length(Outputs),
        outnames = Outputs)

Define vector for output times, parameters, and initial state values

Define inputs, or, in deSolve terminology, forcings

Solve the ODE system

This could all be packaged in a function, for local or global sensitivity analysis, or optimization.
## (at command line)

```r
plot(out, which=c("C_ven","C_liv"))
```

## Using ggplot2 package, probably a in script

```r
pdta <- data.frame(time = out[,"time"] / 60,
                   C_liv = out[,"C_liv"])
pdta2 <- data.frame(x=0,xend=4, y=0, yend=0)
p <- ggplot() +
geom_line(data=pdta,
          mapping=aes(x=time, y=C_liv), size=1) +
geom_segment(data=pdta2,
          mapping=aes(x=x, xend=xend, y=y, yend=yend),
          size=3, color="gray") +
scale_x_continuous("Time (hours)") +
scale_y_continuous("Concentration in Liver (mg/l)")+
theme_bw(base_size=14)
png("Cliver.png", width=5, height=5, unit="in", res=300)
print(p)
dev.off()
```
**Additional R tools for pharmacokinetics (not exhaustive):**


- *numDeriv*, by Paul Gilbert and Ravi Varadhan: accurate numerical derivatives (useful for local sensitivity analysis)

- *deSolve*, by Karline Soetaert, Thomas Petzoldt, and R. Woodrow Setzer: Solvers for initial value ode problems, allowing systems of odes to be expressed in R interpretive language, or in compiled code.

- *httk*, by Robert Pearce and John Wambaugh: Preprogrammed 1, 3, compartmental models, and 4 compartment pbpk models, with included database of *in vitro* estimated clearance for over 500 compounds, partition coefficients computed using Schmitt’s method, with database of the required chemical parameters for several hundred compounds.


- *cpk* (Clinical Pharmacokinetics), by Oscar A. Linares, David T. Daly: Provides simplified clinical pharmacokinetic functions for dose regimen design and modification at the point-of-care.
Software Sources:

- MCSim: https://www.gnu.org/software/mcsim/
- Scripts for running MCSim under R: https://www.gnu.org/software/mcsim/mcsim_under_R.zip
- R: from CRAN, e.g. https://cran.rstudio.com/
- libSBML for Windows (Linux distributions will have their own repositories): http://sourceforge.net/projects/sbml/files/libsbml/5.12.0/stable/Windows/
- deSolve, ggplot2, numDeriv, sensitivity, etc. are R packages, and installed through R’s package installation tool.

As a courtesy to the developers of these programs and packages, please be sure to cite these software tools in publications when you use them.
Summary

- MCSim + R: Code-based, like ACSL-X and Matlab, but uses compiled code
- fast, efficient, flexible, with domain-appropriate languages
- Contains the full functionality of R, currently the leading statistical package