Intended for

Denka Performance Elastomer LLC, Request for Correction

Exhibit A

Date

July 15, 2021

SUPPLEMENTAL MATERIALS E
MODEL FILES

Contents

In Vitro Model Files 4
Invitro.csl (acslX model code) 4
MCMC Run Scripts (m-files that run the analysis) 7
Female Mouse Liver 7
B6 female mouse liver incubation data 8
Female Mouse Lung 10
B6 female mouse lung incubation data 11
Female Rat Liver 13
CDFS female rat liver incubation data 14
Female Rat Lung 16
CDF female rat lung incubation data 17
Female Rat Kidney 19
CDF female rat kidney incubation data 20
Male Mouse Liver 22
B6 male mouse liver incubation data 23
Male Mouse Lung 25
B6 male mouse lung incubation data 26
Male Mouse Kidney 28
B6 male mouse kidney incubation data 29
Male Rat Liver 31
CDF male rat liver incubation data 32
Male Rat Kidney 35
CDF male rat kidney incubation data 36
Mixed Human Liver 38
Mixed human liver incubation data 39
MCMC Control Scripts (establish priors and likelihood, called in MCMC Run Scripts) 41
invitromcmc_sat.m (used for female mouse liver and lung, female rat liver and kidney, male mouse and rat kidney) 41
invitromcmc_satfrlng.m (used for female rat lung with Km fixed to female mouse Km) 46
mmouselivinvitroa0.m (used for male mouse liver with initial amount in vial included) 51
mmouselnginvitroa0.m (used for male mouse lung with initial amount in vial included) 57
mratlivinvitroa0.m (used for male rat liver with initial amount in vial included) 63

invitrohumlivsata0.m (used for mixed human liver with initial amount in vial included) 69
In Vivo Model Files 75
chloroprene.model (R model code) 75
Mouse.R (Base model parameters for mouse) 80
Fmouse.R (Female mouse specific model parameters) 82
Human.R (Base human model parameters) 83
Mhuman.R (mixed human specific model parameters) 85
Fmouse_InVivo.R (simulates 15-day female mouse kinetic study with plots generated for the time-course blood data) 86
Fmouse_metric.R (simulates the bioassay study for two weeks to accumulate daily average dose metrics) 90
Human_Continuous.R (simulates continuous exposure – 7 days/week, 24 hrs/day and reports dose metrics on an average per day basis) 92

In Vitro Model Files

Invitro.csl (acslX model code)

PROGRAM: INVITRO.CSL

```
!MODIFIED FROM Yang et al. 2012 in vitro model
!MODEL TO SIMULATE BETA CHLOROPRENE UPTAKE
!AND METABOLISM IN A TWO-COMPARTMENT
!VIAL SYSTEM CONTAINING MICROSOMES
!Includes flux of chloroprene between air and media



VARIABLE TIME

INITIAL

CONSTANT VMAX1=0.         !MAX RATE OF MET. (uMOL/HR/mg protein)
CONSTANT KM1=0.1          !MICHAELIS CONSTANT (umol/L)
CONSTANT RLOSS = 0.001424   !Background loss rate (L/hr)

CONSTANT P1=0.69          !MEDIA/AIR PARTITION for CD

CONSTANT A10=0.           !INITIAL AMOUNT IN VIAL (uMOL)

CONSTANT VVIAL=0.01165     !VOLUME OF VIAL (L); Vial volume= 11.65 ml
CONSTANT VMED=0.001       !VOLUME OF MEDIA (L); Liquid voume
 VAIR=VVIAL-VMED                          !HEADSPACE
CONSTANT PROT = 1.0       !AMOUNT OF PROTEIN (mg)

CONSTANT TF=0             !TIME OF FIRST SAMPLE (hr); kept same
CONSTANT TI=0.2           !INTERVAL BETWEEN SAMPLES (hr)kept same
CONSTANT VINJ=0.0002      !INJECTION VOLUME (L); based on Matt email

CONSTANT KG1 = 0.11         !L/hr
 KL = KG1/0.69     !KL is set to match media:gas PC

!Initial Conditions
 TV = VAIR+VMED
 AA10 = A10*(VAIR/(VAIR+P1*VMED))                    !amount initially in vial (umol)
 CA10 = AA10/VAIR                !initial concentration in air (umol/L)
 AM10 = CA10*P1*VMED             !amount initially in media
 CM10 = AM10/VMED                        !initial concentration in media (umol/L)

 A1I=0.  !initialize injection volume loss from repeated sampling of vial
```

```
!TIMING COMMANDS
CONSTANT TSTOP=1        !LENGTH OF EXPOSURE (HOURS)
!CONSTANT POINTS=100.    !NO. OF POINTS IN PLOT

TS=TF
SCHEDULE step .AT. TF

END                 !END INITIAL

DYNAMIC
CINTERVAL CINT=0.01     !COMMUNICATION INTERVAL
MAXTERVAL MAXT = 0.001
ALGORITHM IALG=2

  DERIVATIVE

  TERMT(TIME.GE.TSTOP)

!CD KINETICS (umoles/hr)
  RA1M = ((VMAX1*CM1)/(KM1+CM1))*PROT  !rate of metabolism saturable
  RRLoss = RLOSS*CA1                !rate of loss from vial
  A1M = INTEG(RA1M,0.)              !Amount metabolized saturable
  ARLOSS = INTEG(RRLoss,0.)         !Rate of loss to system
  RAG_L = KG1*CA1                   !Rate of amount leaving gas to liquid
  RAL_G = KL*CM1                    !Rate of amount leaving liquid to gas

  RAA1 = RAL_G - RAG_L - RRLoss       !Rate of change in vial air
   AA1 = integ(RAA1, AA10) - A1I
   CA1 = max(AA1/VAIR, 1.0e-7)

  RAM1 = RAG_L - RAL_G - RA1M    !Rate of change in vial media
   AM1 = integ(RAM1, AM10)
   CM1 = AM1/VMED

    A1 = CA1*VAIR+CM1*VMED              !Total amount in vial


!MASS BALANCE
  CHECK1 = A10 - (AM1+AA1+A1M+A1I+ARLOSS)

DISCRETE step
PROCEDURAL
!Routine for sample loss
    A1I= A1I+CA1*VINJ
    SCHEDULE step .AT. TS+TI
    TS=TS+TI
END                 !END PROCEDURAL
END                 !END DISCRETE
```

```
END             !END DERIVATIVE
END             !END DYNAMIC
END             !END PROGRAM
```

MCMC Run Scripts (m-files that run the analysis)

Female Mouse Liver

FMouse_LiverMCMC.m

```
% Simulates the MCMC for Female mouse liver

load @file=invitro.dll @format=model

prepare @clear
prepare @all


WESITG=0;
WEDITG =0;

TSTOP = 1.0 ;
CINT = 0.2 ;
MAXT = 0.01;
TF=0.0; TI=0.2; %Sample Collection start; interval


%Volumes (L) - simulation specific
VVIAL = 0.01165 ;
VMED = 0.001 ;
VINJ = 0.0002 ;
VAIR=VVIAL-VMED ;

%Simulation specific protein concentration
PROT=1.0 ;   %Protein mg/ml


%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;       %L/hr
VMAX1 = 0.0 ;      %umol/hr/mg protein
KM1 = 1.0 ;        %umol/L
P1 = 0.69 ;        %Liquid:air PC




seedrnd(45526)

use ControlData.m

global _ca1
global _time
```

```
global data
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

B6 female mouse liver incubation data

```
%Data reported in Yang et al. Toxicology in Vitro 26 (2012) 1047–1055
%Time  540 ppm          270 ppm          150 ppm          50 ppm 10 ppm
%Headspace Conc. (mg/ml)
B6FmiceLiver = [0        21.378  9.863   5.465   1.867   0.422
0.2      16.789  6.07    2.492   0.411   0.052
0.4      13.771  3.834   1.082   0.081   0.013
0.6      10.624  2.491   0.488   0.018   NaN
0.8      9.99    1.715   0.229   0.007   NaN
1        8.902   1.185   0.131   NaN     NaN];

b = size(B6FmiceLiver);

data = B6FmiceLiver(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = B6FmiceLiver(:, 1);

AA=data(1,:)*(VAIR+P1*VMED);
CCC = [AA];
data = log(data);


function preds = getpreds(Vmax, Km, A10)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);

  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
```

```
   setmdl("A10", A10);
   setmdl("RLOSS", exp(lossR));


  data @clear
  data("SAMPTIMES", ["T"], tindex);

  start @nocallback

  preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
    idx = find(_time == tindex(i));
    if(idx ~= [])
      preds(i) = max(0.0, _ca1(idx));
    end
end

preds = log(preds);

end

use ".\MCMCscripts\invitromcmc_sat.m"

chains = runmcmc();


save @file=fmouseliver1.dat @format=ascii @separator=tab chains
```

Female Mouse Lung

FMouse_LungMCMC.m

```
% Female Mouse Lung In Vitro MCMC simulation file


load @file=invitro.dll @format=model

prepare @clear
prepare @all



WESITG=0;
WEDITG =0;

TSTOP = 1.0 ;
 CINT = 0.2 ;
 MAXT = 0.01;
TF=0.0; TI=0.2; %Sample Collection start; interval

%Volumes (L) - simulation specific
VVIAL = 0.01165 ;
VMED = 0.001 ;
VINJ = 0.0002 ;
VAIR=VVIAL-VMED ;

%Simulation specific protein concentration
PROT=1.0 ;   %Protein mg/ml


%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;        %L/hr
VMAX1 = 0.0 ;       %umol/hr/mg protein
KM1 = 1.0 ;         %umol/L
P1 = 0.69 ;         %Liquid:air PC




seedrnd(45526)

use ControlData.m

global _ca1
global _time
global data
```

```
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

B6 female mouse lung incubation data

```
%Data reported in Yang et al. Toxicology in Vitro 26 (2012) 1047–1055

%Time  540 ppm          270 ppm          150 ppm          50 ppm 10 ppm 1 ppm
%Headspace Conc. (mg/ml)
B6FmiceLung= [0         20.506  12.522  5.97     1.742   0.434   0.044
0.2       18.693  11.105  5.1      1.407   0.346   0.036
0.4       18.014  10.556  4.744    1.24    0.297   0.03
0.6       17.482  10.119  4.469    1.121   0.272   0.027
0.8       16.859  9.644   4.223    1.044   0.25    0.024
1         16.466  9.284   4.006    0.983   0.229   0.023];

b = size(B6FmiceLung);

data = B6FmiceLung(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = B6FmiceLung(:, 1);

AA=data(1,:)*(VAIR+P1*VMED);
CCC = [AA];
data = log(data);

function preds = getpreds(Vmax, Km, A10)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);

  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
  setmdl("A10", A10);
```

```
  setmdl("RLOSS", exp(lossR));


 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\invitromcmc_sat.m"

chains = runmcmc();


save @file=fmouselung1.dat @format=ascii @separator=tab chains
```

Female Rat Liver

FRatLiverMCMC.m

```
% Female Rat Liver In Vitro MCMC simulation file

load @file=invitro.dll @format=model

prepare @clear
prepare @all


WESITG=0;
WEDITG =0;

TSTOP = 1.0 ;
CINT = 0.2 ;
MAXT = 0.01;
TF=0.0; TI=0.2; %Sample Collection start; interval


%Volumes (L) - simulation specific
VVIAL = 0.01165 ;
VMED = 0.001 ;
VINJ = 0.0002 ;
VAIR=VVIAL-VMED ;

%Simulation specific protein concentration
PROT=1.0 ;   %Protein mg/ml


%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;        %L/hr
VMAX1 = 0.0 ;       %umol/hr/mg protein
KM1 = 1.0 ;         %umol/L
P1 = 0.69 ;         %Liquid:air PC



use ControlData.m

seedrnd(45526)



global _ca1
global _time
global data
```

```
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

CDFS female rat liver incubation data

```
%Data reported in Yang et al. Toxicology in Vitro 26 (2012) 1047–1055
%Female Rat Liver
%CDF Liver Summary
%Time  270 ppm          150 ppm          50 ppm 10 ppm 1 ppm
%Headspace Conc. (mg/ml)
FratFLiver=[0    11.007  6.243   1.935   0.465   0.052
0.2      9.091   4.46    0.844   0.141   0.015
0.4      7.661   3.274   0.36    0.048   0.006
0.6      6.621   2.479   0.188   0.022   0.003
0.8      5.831   1.958   0.103   0.011   0.002
1        5.202   1.607   0.066   0.007   NaN];


b = size(FratFLiver);

data = FratFLiver(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = FratFLiver(:, 1);

AA=data(1,:)*(VAIR+P1*VMED);
CCC = [AA];
data = log(data);


function preds = getpreds(Vmax, Km, A10)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);
```

```
  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
  setmdl("A10", A10);
  setmdl("RLOSS", exp(lossR));


 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\invitromcmc_sat.m"

chains = runmcmc();


save @file=fratliver1redo.dat @format=ascii @separator=tab chains
```

Female Rat Lung

FRatLungMCMCvmax.m

```
% Female Rat Lung In Vitro MCMC simulation file
% Km set to posterior median of the female mouse lung
% Allows estimation of the in vitro Vmax for female rat lung


load @file=invitro.dll @format=model

prepare @clear
prepare @all

WESITG=0;
WEDITG =0;

TSTOP = 1.0 ;
CINT = 0.2 ;
MAXT = 0.01;
TF=0.0; TI=0.2; %Sample Collection start; interval


%Volumes (L) - simulation specific
VVIAL = 0.01165 ;
VMED = 0.001 ;
VINJ = 0.0002 ;
VAIR=VVIAL-VMED ;

%Simulation specific protein concentration
PROT=1.0 ;   %Protein mg/ml


%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;        %L/hr
VMAX1 = 0.0 ;       %umol/hr/mg protein

P1 = 0.69 ;         %Liquid:air PC

KM1 = 2.369 ;         %umol/L

use ControlData.m

seedrnd(45526)
```

```
global _ca1
global _time
global data
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

CDF female rat lung incubation data

```
%Data reported in Yang et al. Toxicology in Vitro 26 (2012) 1047–1055

%CDF Lung Summary
%Time  150 ppm          50 ppm 10 ppm
%Headspace Conc. (mg/ml)
FratFLung=[0    11.438  5.107    2.051
0.2       10.93   4.611    1.914
0.4       10.256  4.452    1.829
0.6       9.786   4.156    1.755
0.8       9.44    4.131    1.682
1         8.88    3.774    1.641 ];


b = size(FratFLung);

data = FratFLung(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = FratFLung(:, 1);

AA=data(1,:)*(VAIR+P1*VMED);
CCC = [AA];
data = log(data);

function preds = getpreds(Vmax, A10)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
```

```
  lossR = ControlData(tmp);

  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("A10", A10);
  setmdl("RLOSS", exp(lossR));


 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\invitromcmc_satfrlng.m"

chains = runmcmc();


save @file=fratlung1.dat @format=ascii @separator=tab chains
```

Female Rat Kidney

FRatLungMCMCvmax.m

```
% Female Rat Kidney In Vitro MCMC simulation file


load @file=invitro.dll @format=model

prepare @clear
prepare @all

WESITG=0;
WEDITG =0;

TSTOP = 1.0 ;
 CINT = 0.2 ;
 MAXT = 0.001;
TF=0.0; TI=0.2; %Sample Collection start; interval


%Volumes (L) - simulation specific
VVIAL = 0.01163 ;
VMED = 0.001 ;
VINJ = 0.0002 ;
VAIR = VVIAL-VMED ;

%Simulation specific protein concentration
PROT = 3.0 ;   %Protein mg/ml


%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;       %L/hr
VMAX1 = 0.0 ;      %umol/hr/mg protein
KM1 = 1.0 ;        %umol/L
P1 = 0.69 ;        %Liquid:air PC



seedrnd(45526)

use ControlData.m

global _ca1
global _time
global data
global tindex
```

```
global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

CDF female rat kidney incubation data

```
%Data reported in Yang et al. Toxicology in Vitro 26 (2012) 1047–1055

%Time  540 ppm 270 ppm        150 ppm         50 ppm 10 ppm 2 ppm
%Headspace Conc. (mg/ml)
FRatKid = [0.     22.705  11.003  5.381   1.985   0.436   0.090   ;
           0.2    21.864  10.443  5.102   1.785   0.366   0.078   ;
           0.4    21.230  10.065  4.888   1.636   0.311   0.056   ;
           0.6    20.674  9.656   4.624   1.497   0.266   0.046   ;
           0.8    19.735  9.259   4.387   1.393   0.237   0.044   ;
           1.     18.879  8.792   4.216   1.297   0.222   0.043   ];


b = size(FRatKid);

data = FRatKid(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = FRatKid(:, 1);

AA=data(1,:)*(VAIR+P1*VMED);
CCC = [AA];
data = log(data);

function preds = getpreds(Vmax, Km, A10)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);

  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
```

```
   setmdl("A10", A10);
   setmdl("RLOSS", exp(lossR));


  data @clear
  data("SAMPTIMES", ["T"], tindex);

  start @nocallback

  preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
    idx = find(_time == tindex(i));
    if(idx ~= [])
      preds(i) = max(0.0, _ca1(idx));
    end
end

preds = log(preds);

end

use ".\MCMCscripts\invitromcmc_sat.m"

chains = runmcmc();


save @file=fratkidney1.dat @format=ascii @separator=tab chains
```

Male Mouse Liver

MmouseLiverMCMCa0.m

```
% Male Mouse Liver In Vitro MCMC simulation file
% Includes estimation of the initial amount of CP in vial
% due to multiple vials making up a single concentration


load @file=invitro.dll @format=model

prepare @clear
prepare @all

WESITG=0;
WEDITG =0;

CINT = 0.2 ;
MAXT = 0.001 ;
TSTOP = 1.0 ;
TF=0.0; TI=0.2; %Sample Collection start; interval


%Volumes (L) - simulation specific
VVIAL = 0.0119573;
VMED = 0.001;
VINJ = 0.0003858 ;
VAIR = VVIAL-VMED;

%Simulation specific protein concentration
PROT=1.0 ;   %Protein mg/ml

%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;        %L/hr
VMAX1 = 0.0 ;       %umol/hr/mg protein
KM1 = 1.0 ;           %umol/L
P1 = 0.69;          %Liquid:air PC



seedrnd(45526)

use ControlData.m

global _ca1
global _time
global data
```

```
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

B6 male mouse liver incubation data

```
%Data reported in Himmelstein et al. TOXICOLOGICAL SCIENCES 79, 18–27 (2004)

% B6 male mouse liver
%[Time 529 ppm          264 ppm          132 ppm          50 ppm 10 ppm]
%Headspace Conc. (mg/ml)
B6MmiceLiver = [0       21.202  10.747  5.282   1.987   0.465
0.025   20.770  NaN     5.069   1.869   0.395
0.05    19.932  9.898   4.434   1.530   0.278
0.1     NaN     8.394   2.848   0.890   NaN
0.15    17.435  7.343   2.342   NaN     NaN
0.2     15.409  5.619   1.444   0.315   0.052
0.225   14.788  NaN     1.302   0.271   0.040
0.25    14.338  4.668   1.082   0.199   0.021
0.3     NaN     3.684   0.495   0.126   NaN
0.35    12.542  3.125   0.480   NaN     NaN
0.4     10.747  2.179   0.270   0.036   0.005
0.425   NaN     NaN     0.241   NaN     NaN
0.45    NaN     1.685   0.190   0.020   NaN
0.5     NaN     1.224   0.062   NaN     NaN
0.55    8.481   1.049   0.079   NaN     NaN
0.6     6.975   0.640   0.041   0.005   NaN
0.625   NaN     NaN     0.038   NaN     NaN
0.65    NaN     0.460   0.029   NaN     NaN
0.7     NaN     0.318   0.0077  NaN     NaN
0.75    5.561   0.270   0.013   NaN     NaN
0.8     4.299   0.159   0.0069  NaN     NaN
0.825   NaN     NaN     0.0059  NaN     NaN
0.85    NaN     0.109   0.0050  NaN     NaN
0.9     NaN     0.073   NaN     NaN     NaN
0.95    3.393   0.066   NaN     NaN     NaN];


b = size(B6MmiceLiver);

data = B6MmiceLiver(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
```

```
firstD = [1]
lastD  = [b(:,2)-1]
tindex = B6MmiceLiver(:, 1);

%AA=data(1,:)*(VAIR+P1*VMED);
%CCC = [AA];
data = log(data);

function preds = getpreds(Vmax, Km, A0)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);

  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
  setmdl("A10", exp(A0));
  setmdl("RLOSS", exp(lossR));


 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\mmouselivinvitroa0.m"

chains = runmcmc();

save @file=mmouseliver1a0.dat @format=ascii @separator=tab chains
```

Male Mouse Lung

MmouseLungMCMCa0.m

```
% Male Mouse Lung In Vitro MCMC simulation file
% Includes estimation of the initial amount of CP in vial
% due to multiple vials making up a single concentration



load @file=invitro.dll @format=model

prepare @clear
prepare @all



WESITG=0;
WEDITG =0;

 TSTOP = 1.0 ;
 CINT = 0.2 ;
 MAXT = 0.001;
 TF=0.0; TI=0.2; %Sample Collection start; interval

%Volumes (L) - simulation specific
VVIAL = 0.0119573;
VMED = 0.001;
VINJ = 0.0003858 ;
VAIR = VVIAL-VMED;

%Simulation specific protein concentration
PROT=1.0 ;   %Protein mg/ml

%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;        %L/hr
VMAX1 = 0.0 ;       %umol/hr/mg protein
KM1 = 1.0 ;           %umol/L
P1 = 0.69;         %Liquid:air PC


seedrnd(45526)

use ControlData.m

global _ca1
```

```
global _time
global data
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

B6 male mouse lung incubation data

```
%Data reported in Himmelstein et al. TOXICOLOGICAL SCIENCES 79, 18–27 (2004)

%time   10012.1 ppm     1000 ppm        250 ppm         50ppm  10 ppm
%Headspace Conc. (mg/ml)
B6MmiceLung = [0        90.6864         36.9563         8.5541  1.9319  0.3288
0.025   92.9729         34.7636         8.5569  1.6268  0.3629
0.05    NaN     32.7002         NaN     1.7520  0.2736
0.1     NaN     31.5351         7.1220  1.2424  NaN
0.15    NaN     NaN     NaN     1.1129  NaN
0.2     86.5465         29.6910         5.3084  0.8496  0.1464
0.225   87.6422         28.9789         5.4422  0.7895  0.1311
0.25    NaN     27.5967         NaN     0.7335  0.1104
0.3     NaN     27.2762         4.5859  0.6029  NaN
0.35    NaN     NaN     NaN     0.5335  NaN
0.4     NaN     25.3669         3.7714  0.4226  0.0652
0.425   83.6431         26.0065         3.9627  0.3831  0.0598
0.45    NaN     24.1644         NaN     0.3516  0.0523
0.5     NaN     NaN     3.1788  0.3228  NaN
0.55    NaN     NaN     NaN     0.2821  NaN
0.6     NaN     23.9130         2.9203  0.2348  0.0309
0.625   79.8247         23.9206         3.1837  0.1977  0.0290
0.65    NaN     21.9544         NaN     0.1906  0.0270
0.7     NaN     NaN     NaN     0.1861  NaN
0.8     NaN     22.2670         2.3174  0.1337  0.0163
0.825   75.2488         NaN     2.5895  0.1154  0.0148
0.85    NaN     20.9520         NaN     0.1072  0.0148
0.9     NaN     NaN     2.1820  0.1155  NaN];


b = size(B6MmiceLung);

data = B6MmiceLung(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
```

```
lastD  = [b(:,2)-1]
tindex = B6MmiceLung(:, 1);

%AA=data(1,:)*(VAIR+P1*VMED);
%CCC = [AA];
data = log(data);

function preds = getpreds(Vmax, Km, A0)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);

  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
  setmdl("A10", exp(A0));
  setmdl("RLOSS", exp(lossR));


 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\mmouselnginvitroa0.m"

chains = runmcmc();

save @file=mmouselung1a0.dat @format=ascii @separator=tab chains
```

Male Mouse Kidney

MMouseKidneyMCMC.m

```
% Male Mouse Kidney In Vitro MCMC simulation file


load @file=invitro.dll @format=model

prepare @clear
prepare @all



WESITG=0;
WEDITG =0;

TSTOP = 1.0 ;
 CINT = 0.2 ;
 MAXT = 0.01;
TF=0.0; TI=0.2; %Sample Collection start; interval


%Volumes (L) - simulation specific
VVIAL = 0.01163 ;
VMED = 0.001 ;
VINJ = 0.0002 ;
VAIR = VVIAL-VMED ;

%Simulation specific protein concentration
PROT = 2.0 ;   %Protein mg/ml


%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;       %L/hr
VMAX1 = 0.0 ;      %umol/hr/mg protein
KM1 = 1.0 ;        %umol/L
P1 = 0.69 ;        %Liquid:air PC


seedrnd(45526)

use ControlData.m

global _ca1
global _time
global data
```

```
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

B6 male mouse kidney incubation data

```
%Data reported in Yang et al. Toxicology in Vitro 26 (2012) 1047–1055

% B6C3F1 Male mouse kidney CD metabolism
% Time 540 ppm        270 ppm        150 ppm        50 ppm 10 ppm 2 ppm
% protein  = 2mg/ml
%Headspace Conc. (mg/ml)
MMiceKid = [    0.      19.674  10.591  5.341   1.764   0.358   0.063
                0.2     17.887  10.001  4.742   1.303   0.207   0.038
                0.4     16.760  9.268   4.202   1.030   0.123   0.025
                0.6     15.391  8.640   3.853   0.822   0.077   0.018
                0.8     14.566  8.100   3.505   0.692   0.055   0.014
                1.      13.509  7.809   3.327   0.617   0.043   0.011   ];


b = size(MMiceKid);

data = MMiceKid(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = MMiceKid(:, 1);

AA=data(1,:)*(VAIR+P1*VMED);
CCC = [AA];
data = log(data);

function preds = getpreds(Vmax, Km, A10)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);
```

```
  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
  setmdl("A10", A10);
  setmdl("RLOSS", exp(lossR));


 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\invitromcmc_sat.m"

chains = runmcmc();


save @file=mmousekidney1.dat @format=ascii @separator=tab chains
```

Male Rat Liver

MratLiverMCMCa0.m

```
% Male Rat Liver In Vitro MCMC simulation file
% Includes estimation of the initial amount of CP in vial
% due to multiple vials making up a single concentration



load @file=invitro.dll @format=model

prepare @clear
prepare @all


WESITG=0;
WEDITG =0;

CINT = 0.2 ;
MAXT = 0.001 ;
TSTOP = 1.0 ;
TF=0.0; TI=0.2; %Sample Collection start; interval


%Volumes (L) - simulation specific
VVIAL = 0.0119573;
VMED = 0.001;
VINJ = 0.0003858 ;
VAIR = VVIAL-VMED;

%Simulation specific protein concentration
PROT=1.0 ;   %Protein mg/ml

%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;        %L/hr
VMAX1 = 0.0 ;       %umol/hr/mg protein
KM1 = 1.0 ;           %umol/L
P1 = 0.69;         %Liquid:air PC


seedrnd(45526)

use ControlData.m

global _ca1
global _time
```

```
global data
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

CDF male rat liver incubation data

```
%Data reported in Himmelstein et al. TOXICOLOGICAL SCIENCES 79, 18–27 (2004)


%male rat liver data
%[Time 264 ppm 132 ppm 50 ppm]
%Headspace Conc. (mg/ml)
MratMLiver=[0 9.824   4.6755  2.0125
0.025   9.454   4.503   2.18
0.05    8.939   4.318   1.634
0.1     9.767   3.918   1.354
0.15    9.603   3.708   1.113
0.2     7.856   3.217   0.893
0.225   7.581   3.007   0.931
0.25    7.02    2.885   0.706
0.3     7.925   2.559   0.545
0.35    7.679   2.478   0.419
0.4     6.097   2.0245  0.291
0.425   5.974   1.841   0.308
0.45    5.568   1.786   0.237
0.5     6.201   1.547   0.175
0.55    NaN     1.558   0.125
0.6     4.637   1.1375  0.077
0.625   4.584   1.01    0.082
0.65    4.231   0.995   0.067
0.7     NaN     0.837   0.048
0.75    NaN     0.708   0.034
0.8     3.482   0.5715  0.0195
0.825   3.428   0.483   0.02
0.85    3.18    0.489   0.018
0.9     NaN     0.397   NaN
0.95    NaN     NaN     0.009];


b = size(MratMLiver);

data = MratMLiver(:,2:b(:,2));
```

```
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = MratMLiver(:, 1);

%AA=data(1,:)*(VAIR+P1*VMED);
%CCC = [AA];
data = log(data);

function preds = getpreds(Vmax, Km, A0)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);

  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
  setmdl("A10", exp(A0));
  setmdl("RLOSS", exp(lossR));


 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\mratlivinvitroa0.m"

chains = runmcmc();
```

save @file=mratliver1a0.dat @format=ascii @separator=tab chains

Male Rat Kidney

MRatKidneyMCMC.m

```
% Male Rat Kidney In Vitro MCMC simulation file


load @file=invitro.dll @format=model

prepare @clear
prepare @all

WESITG=0;
WEDITG =0;

TSTOP = 1.0 ;
 CINT = 0.2 ;
 MAXT = 0.01;
TF=0.0; TI=0.2; %Sample Collection start; interval


%Volumes (L) - simulation specific
VVIAL = 0.01163 ;
VMED = 0.001 ;
VINJ = 0.0002 ;
VAIR = VVIAL-VMED ;

%Simulation specific protein concentration
PROT = 3.0 ;   %Protein mg/ml


%Initial values
RLOSS = 0.001424 ;  %L/hr
KG1 = 0.22 ;       %L/hr
VMAX1 = 0.0 ;      %umol/hr/mg protein
KM1 = 1.0 ;        %umol/L
P1 = 0.69 ;        %Liquid:air PC



seedrnd(45526)

use ControlData.m

global _ca1
global _time
global data
global tindex
```

```
global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

CDF male rat kidney incubation data

```
%Data reported in Yang et al. Toxicology in Vitro 26 (2012) 1047–1055

% CDF male rat kidney CD metabolism ; (3 mg protein/mL)
% w/NADP+      Time    540 ppm         270 ppm         150 ppm         50 ppm 10 ppm 2 ppm
%Headspace Conc. (mg/ml)
MRatKid = [0.   22.510  12.125  5.405   1.810   0.418   0.096
                0.2     21.626  11.497  5.084   1.631   0.350   0.080
                0.4     21.101  11.072  4.773   1.486   0.298   0.068
                0.6     20.388  10.582  4.515   1.380   0.265   0.059
                0.8     19.658  10.032  4.305   1.284   0.247   0.054
                1.      18.978  9.587   4.058   1.206   0.228   0.051];


b = size(MRatKid);

data = MRatKid(:,2:b(:,2));
firstT = [1]
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = MRatKid(:, 1);

AA=data(1,:)*(VAIR+P1*VMED);
CCC = [AA];
data = log(data);


function preds = getpreds(Vmax, Km, A10)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate from ControlData matrix
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);
```

```
  setmdl("VMAX1", exp(Vmax));
  setmdl("KM1", exp(Km));
  setmdl("A10", A10);
  setmdl("RLOSS", exp(lossR));



 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\invitromcmc_sat.m"

chains = runmcmc();


save @file=mratkidney1.dat @format=ascii @separator=tab chains
```

Mixed Human Liver

HumanLiverMCMCa0.m

```
% Human Liver In Vitro MCMC simulation file
% Includes estimation of the initial amount of CP in vial
% due to multiple vials making up a single concentration



set @format=shorte
%load @file=invitro.dll @format=model

%prepare @clear
%prepare @all


WESITG = 0;
WEDITG = 0;

CINT = 0.2 ;
MAXT = 0.001 ;
TSTOP = 1.0 ;
TF=0.0; TI=0.2; %Sample Collection start; interval

%Volumes (L) - simulation specific
VVIAL = 0.0119573;
VMED = 0.001;
VINJ = 0.0004 ;
VAIR=VVIAL-VMED;

%Simulation specific protein concentration
PROT=1.0 ;   %Protein mg/ml


%Initial values
RLOSS = 0.001424 ;  %L/hr
%KG1 = 0.44 ;       %L/hr
VMAX1 = 0.0 ;      %umol/hr/mg protein
KM1 = 1.0 ;        %umol/L
P1 = 0.69;         %Liquid:air PC



seedrnd(45526)

use ControlData.m
```

```
global _ca1
global _time
global data
global tindex


global CCC
global firstT
global lastT
global firstD
global lastD
global ControlData
```

Mixed human liver incubation data

```
%Data reported in Himmelstein et al. TOXICOLOGICAL SCIENCES 79, 18–27 (2004)

%Time  264 ppm          132 ppm          50 ppm
%Headspace Conc. (mg/ml)
humanliver = [0 9.443   4.749   1.663
0.025   9.345   4.638   1.683
0.05    8.807   4.644   1.586
0.1     9.093   4.797   1.358
0.15    8.551   4.44    1.175
0.2     7.941   3.617   0.941
0.225   7.787   3.548   0.881
0.25    7.308   3.506   0.804
0.3     7.808   3.477   0.666
0.35    7.042   3.361   0.556
0.4     6.606   2.533   0.39
0.425   6.45    2.534   0.359
0.45    NaN     2.458   0.319
0.5     NaN     2.421   NaN
0.55    5.768   2.463   NaN
0.6     5.444   1.623   0.133
0.625   NaN     1.683   0.117
0.65    NaN     1.734   0.102
0.7     NaN     1.525   NaN
0.75    4.624   1.439   NaN
0.8     4.387   0.92    0.04
0.825   NaN     1.013   0.035
0.85    NaN     0.97    0.031
0.9     NaN     0.867   NaN
0.95    3.632   0.822   NaN];


b = size(humanliver);
data = humanliver(:,2:b(:,2));
firstT = [1]
```

```
lastT  = [b(:,1)]
firstD = [1]
lastD  = [b(:,2)-1]
tindex = humanliver(:, 1);

AA=data(1,:)*(VAIR+P1*VMED);
CCC = [AA];
data = log(data);

function preds = getpreds(Vmax, Km, A0)
  global _ca1
  global _time
  global tindex
  global ControlData

  % draw back ground loss rate
  tmp = ceil(rand*500);
  lossR = ControlData(tmp);

  setmdl("VMAX1", exp(Vmax));  % reset model parameter as global variables
  setmdl("KM1", exp(Km));
  setmdl("A10", exp(A0));
  setmdl("RLOSS", exp(lossR));

 data @clear
 data("SAMPTIMES", ["T"], tindex);

 start @nocallback

 preds = NaN*ones(length(tindex), 1);

for i = 1:length(tindex)
   idx = find(_time == tindex(i));
   if(idx ~= [])
     preds(i) = max(0.0, _ca1(idx));
   end
end

preds = log(preds);

end

use ".\MCMCscripts\invitrohumlivsata0.m"

chains = runmcmc();

save @format=ascii @file=humanlivera0.dat @separator=tab chains
```

MCMC Control Scripts (establish priors and likelihood, called in MCMC Run Scripts)

invitromcmc_sat.m (used for female mouse liver and lung, female rat liver and kidney, male mouse and rat kidney)

```
function tchains = runmcmc(pchains = [])
   % Driver code for MCMC analysis
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds

   numParms = 3
   numChains = 3
   numIts = 50000
   funcNames = ["mcInit", "mcEvalLikelihoods", "mcEvalPriors", "mcSamplePriors", "mcEvalProposal",
"mcSampleProposal"]
   updateMode = 4
   chains = mcmc(numParms, numIts, numChains, updateMode, funcNames, pchains);
   save @format=ascii @file=mcmc_results.dat chains
   tchains = chains([1:2:50000],:);
end

function mcInit()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   global OpMcmcPriorBounds
   OpMcmcPriorBounds = [...
   0.01, 100
   -10, 5
   -10, 5
```

```
    ];
    global OpMcmcAdaptive
    OpMcmcAdaptive = 1;
    global OpMcmcDelayedRejection
    OpMcmcDelayedRejection = 0;
    global OpMcmcAdaptPeriod
    OpMcmcAdaptPeriod = 30;
    global OpMcmcAdaptCovarScale
    OpMcmcAdaptCovarScale = 1;
    global OpMcmcLoggingPeriod
    OpMcmcLoggingPeriod = 50;
    global OpMcmcAdaptLowerThresh
    OpMcmcAdaptLowerThresh = 0.25;
    global OpMcmcAdaptUpperThresh
    OpMcmcAdaptUpperThresh = 0.45;
    global OpMcmcAdaptLowerThreshDR
    OpMcmcAdaptLowerThreshDR = 0.45;
    global OpMcmcAdaptUpperThreshDR
    OpMcmcAdaptUpperThreshDR = 0.65;
    global OpMcmcSigmaDecreaseFact
    OpMcmcSigmaDecreaseFact = 0.9;
    global OpMcmcSigmaIncreaseFact
    OpMcmcSigmaIncreaseFact = 1.1;
    global OpMcmcDRSigmaReduceFact
    OpMcmcDRSigmaReduceFact = 0.2;
    global OpMcmcDRSigmaReduceFactAM
    OpMcmcDRSigmaReduceFactAM = 0.1;
    global OpMcmcAdaptLowerThreshAM
    OpMcmcAdaptLowerThreshAM = 0.15;
    global OpMcmcAdaptUpperThreshAM
    OpMcmcAdaptUpperThreshAM = 0.3;
    global OpMcmcCovarScaleDecreaseFact
    OpMcmcCovarScaleDecreaseFact = 20;
    global OpMcmcCovarScaleIncreaseFact
    OpMcmcCovarScaleIncreaseFact = 20;
    global OpDemcSnookerFraction
    OpDemcSnookerFraction = 0.1;
    global OpDemcThinningFactor
    OpDemcThinningFactor = 10;
    global OpDemcB
    OpDemcB = 0.0001;
end

function samp = mcSampleProposal(prevsamp)
    global data
    global firstT
    global lastT
    global firstD
```

```
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   samp = [];
   % This function is a stub...
   % Code for a user-defined proposal function can be inserted here.
end

function val = mcEvalProposal(samp, prevsamp)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   val = 0;
   % This function is a stub...
   % Code for a user-defined proposal function can be inserted here.
end

function mcDumpSamples()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   LI
   Vmax
   Km
end

function names = mcSampNames()
   names = "LI";
   names = [names, "Vmax"];
   names = [names, "Km"];
   names
```

```
end

function parms = mcPackSamples()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   parms = [];
   parms = [parms LI];
   parms = [parms Vmax];
   parms = [parms Km];
end

function mcUnpackSamples(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   idx = 1;
   LI = parms(idx); idx = idx + 1;
   Vmax = parms(idx); idx = idx + 1;
   Km = parms(idx); idx = idx + 1;
end

function parms = mcSamplePriors()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   LI = normrnd(1, 1, 0.01, 100);
```

```
   Vmax = unifrnd(-10, 5);
   Km = unifrnd(-10, 5);
   parms = mcPackSamples();
end

function val = mcEvalPriors(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   mcUnpackSamples(parms);
   val = 0.0;
   val = val + normlpdf(LI, 1, 1, 0.01, 100);
   val = val + uniflpdf(Vmax, -10, 5);
   val = val + uniflpdf(Km, -10, 5);
end

function val = mcEvalLikelihoods(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global Km
   global preds
   mcUnpackSamples(parms);
   val = 0.0;
   for i = firstD : lastD
      preds = getpreds(Vmax, Km, CCC(i));
      for j = firstT : lastT
         if(~isnan(data(j, i)))
            val = val + normlpdf(data(j, i), preds(j), LI);
         end
      end
   end
end
```

invitromcmc_satfrlng.m (used for female rat lung with Km fixed to female mouse Km)

```
function tchains = runmcmc(pchains = [])
   % Driver code for MCMC analysis
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global preds

   numParms = 2
   numChains = 3
   numIts = 40000
   funcNames = ["mcInit", "mcEvalLikelihoods", "mcEvalPriors", "mcSamplePriors", "mcEvalProposal",
"mcSampleProposal"]
   updateMode = 4
   chains = mcmc(numParms, numIts, numChains, updateMode, funcNames, pchains);
   save @format=ascii @file=mcmc_results.dat chains
   tchains = chains([1:2:40000],:);
end

function mcInit()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global preds
   global OpMcmcPriorBounds
   OpMcmcPriorBounds = [...
   0.01, 100
   -10, 5
   ];
   global OpMcmcAdaptive
   OpMcmcAdaptive = 1;
   global OpMcmcDelayedRejection
   OpMcmcDelayedRejection = 0;
   global OpMcmcAdaptPeriod
   OpMcmcAdaptPeriod = 30;
   global OpMcmcAdaptCovarScale
```

```
    OpMcmcAdaptCovarScale = 1;
    global OpMcmcLoggingPeriod
    OpMcmcLoggingPeriod = 50;
    global OpMcmcAdaptLowerThresh
    OpMcmcAdaptLowerThresh = 0.25;
    global OpMcmcAdaptUpperThresh
    OpMcmcAdaptUpperThresh = 0.45;
    global OpMcmcAdaptLowerThreshDR
    OpMcmcAdaptLowerThreshDR = 0.45;
    global OpMcmcAdaptUpperThreshDR
    OpMcmcAdaptUpperThreshDR = 0.65;
    global OpMcmcSigmaDecreaseFact
    OpMcmcSigmaDecreaseFact = 0.9;
    global OpMcmcSigmaIncreaseFact
    OpMcmcSigmaIncreaseFact = 1.1;
    global OpMcmcDRSigmaReduceFact
    OpMcmcDRSigmaReduceFact = 0.2;
    global OpMcmcDRSigmaReduceFactAM
    OpMcmcDRSigmaReduceFactAM = 0.1;
    global OpMcmcAdaptLowerThreshAM
    OpMcmcAdaptLowerThreshAM = 0.15;
    global OpMcmcAdaptUpperThreshAM
    OpMcmcAdaptUpperThreshAM = 0.3;
    global OpMcmcCovarScaleDecreaseFact
    OpMcmcCovarScaleDecreaseFact = 20;
    global OpMcmcCovarScaleIncreaseFact
    OpMcmcCovarScaleIncreaseFact = 20;
    global OpDemcSnookerFraction
    OpDemcSnookerFraction = 0.1;
    global OpDemcThinningFactor
    OpDemcThinningFactor = 10;
    global OpDemcB
    OpDemcB = 0.0001;
end

function samp = mcSampleProposal(prevsamp)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global preds
    samp = [];
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
```

```
end

function val = mcEvalProposal(samp, prevsamp)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global preds
   val = 0;
   % This function is a stub...
   % Code for a user-defined proposal function can be inserted here.
end

function mcDumpSamples()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global preds
   LI
   Vmax
end

function names = mcSampNames()
   names = "LI";
   names = [names, "Vmax"];
   names
end

function parms = mcPackSamples()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global preds
   parms = [];
```

```
    parms = [parms LI];
    parms = [parms Vmax];
end

function mcUnpackSamples(parms)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global preds
    idx = 1;
    LI = parms(idx); idx = idx + 1;
    Vmax = parms(idx); idx = idx + 1;
end

function parms = mcSamplePriors()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global preds
    LI = normrnd(1, 1, 0.01, 100);
    Vmax = unifrnd(-10, 5);
    parms = mcPackSamples();
end

function val = mcEvalPriors(parms)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global preds
    mcUnpackSamples(parms);
    val = 0.0;
    val = val + normlpdf(LI, 1, 1, 0.01, 100);
    val = val + uniflpdf(Vmax, -10, 5);
```

```
end

function val = mcEvalLikelihoods(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global LI
   global Vmax
   global preds
   mcUnpackSamples(parms);
   val = 0.0;
   for i = firstD : lastD
      preds = getpreds(Vmax, CCC(i));
      for j = firstT : lastT
         if(~isnan(data(j, i)))
            val = val + normlpdf(data(j, i), preds(j), LI);
         end
      end
   end
end
```

mmouselivinvitroa0.m (used for male mouse liver with initial amount in vial included)

```
function tchains = runmcmc(pchains = [])
   % Driver code for MCMC analysis
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   A0_i = zeros(5, 1);

   numParms = 8
   numChains = 3
   numIts = 50000
   funcNames = ["mcInit", "mcEvalLikelihoods", "mcEvalPriors", "mcSamplePriors", "mcEvalProposal",
"mcSampleProposal"]
   updateMode = 4
   chains = mcmc(numParms, numIts, numChains, updateMode, funcNames, pchains);
   save @format=ascii @file=mcmc_results.dat chains
   tchains = chains([1:2:50000],:);
end

function mcInit()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   global OpMcmcPriorBounds
   OpMcmcPriorBounds = [...
   -10, 0
   -10, 0
   -10, 0
   -10, 0
   -10, 0
```

```
    0.01, 100
    -15, 5
    -15, 5
    ];
    global OpMcmcAdaptive
    OpMcmcAdaptive = 1;
    global OpMcmcDelayedRejection
    OpMcmcDelayedRejection = 0;
    global OpMcmcAdaptPeriod
    OpMcmcAdaptPeriod = 30;
    global OpMcmcAdaptCovarScale
    OpMcmcAdaptCovarScale = 1;
    global OpMcmcLoggingPeriod
    OpMcmcLoggingPeriod = 200;
    global OpMcmcAdaptLowerThresh
    OpMcmcAdaptLowerThresh = 0.25;
    global OpMcmcAdaptUpperThresh
    OpMcmcAdaptUpperThresh = 0.45;
    global OpMcmcAdaptLowerThreshDR
    OpMcmcAdaptLowerThreshDR = 0.45;
    global OpMcmcAdaptUpperThreshDR
    OpMcmcAdaptUpperThreshDR = 0.65;
    global OpMcmcSigmaDecreaseFact
    OpMcmcSigmaDecreaseFact = 0.9;
    global OpMcmcSigmaIncreaseFact
    OpMcmcSigmaIncreaseFact = 1.1;
    global OpMcmcDRSigmaReduceFact
    OpMcmcDRSigmaReduceFact = 0.2;
    global OpMcmcDRSigmaReduceFactAM
    OpMcmcDRSigmaReduceFactAM = 0.1;
    global OpMcmcAdaptLowerThreshAM
    OpMcmcAdaptLowerThreshAM = 0.15;
    global OpMcmcAdaptUpperThreshAM
    OpMcmcAdaptUpperThreshAM = 0.3;
    global OpMcmcCovarScaleDecreaseFact
    OpMcmcCovarScaleDecreaseFact = 20;
    global OpMcmcCovarScaleIncreaseFact
    OpMcmcCovarScaleIncreaseFact = 20;
    global OpDemcSnookerFraction
    OpDemcSnookerFraction = 0.1;
    global OpDemcThinningFactor
    OpDemcThinningFactor = 10;
    global OpDemcB
    OpDemcB = 0.0001;
end

function samp = mcSampleProposal(prevsamp)
    global data
```

```
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    samp = [];
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function val = mcEvalProposal(samp, prevsamp)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    val = 0;
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function mcDumpSamples()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    A0_i
    LI
    Vmax
    Km
```

```
end

function names = mcSampNames()
   names = "A0_i(1)";
   names = [names, "A0_i(2)"];
   names = [names, "A0_i(3)"];
   names = [names, "A0_i(4)"];
   names = [names, "A0_i(5)"];
   names = [names, "LI"];
   names = [names, "Vmax"];
   names = [names, "Km"];
   names
end

function parms = mcPackSamples()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   parms = [];
   parms = [parms reshape(A0_i, 1, 5)];
   parms = [parms LI];
   parms = [parms Vmax];
   parms = [parms Km];
end

function mcUnpackSamples(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   idx = 1;
   A0_i = reshape(parms(idx:idx+4), 5, 1); idx = idx + 5;
   LI = parms(idx); idx = idx + 1;
```

```
    Vmax = parms(idx); idx = idx + 1;
    Km = parms(idx); idx = idx + 1;
end

function parms = mcSamplePriors()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    LI = normrnd(1, 1, 0.01, 100);
    Vmax = unifrnd(-15, 5);
    Km = unifrnd(-15, 5);
    for i = firstD : lastD
        A0_i(i) = unifrnd(-10, 0);
    end
    parms = mcPackSamples();
end

function val = mcEvalPriors(parms)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    mcUnpackSamples(parms);
    val = 0.0;
    val = val + normlpdf(LI, 1, 1, 0.01, 100);
    val = val + uniflpdf(Vmax, -15, 5);
    val = val + uniflpdf(Km, -15, 5);
    for i = firstD : lastD
        val = val + uniflpdf(A0_i(i), -10, 0);
    end
end

function val = mcEvalLikelihoods(parms)
```

```
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    mcUnpackSamples(parms);
    val = 0.0;
    for i = firstD : lastD
        preds = getpreds(Vmax, Km, A0_i(i));
        for j = firstT : lastT
            if(~isnan(data(j, i)))
                val = val + normlpdf(data(j, i), preds(j), LI);
            end
        end
    end
end
```

mmouselnginvitroa0.m (used for male mouse lung with initial amount in vial included)

```
function tchains = runmcmc(pchains = [])
   % Driver code for MCMC analysis
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   A0_i = zeros(5, 1);

   numParms = 8
   numChains = 3
   numIts = 50000
   funcNames = ["mcInit", "mcEvalLikelihoods", "mcEvalPriors", "mcSamplePriors", "mcEvalProposal",
"mcSampleProposal"]
   updateMode = 4
   chains = mcmc(numParms, numIts, numChains, updateMode, funcNames, pchains);
   save @format=ascii @file=mcmc_results.dat chains
   tchains = chains([1:2:50000],:);
end

function mcInit()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   global OpMcmcPriorBounds
   OpMcmcPriorBounds = [...
   -10, 0
   -10, 0
   -10, 0
   -10, 0
   -10, 0
```

```
    0.01, 100
    -15, 5
    -15, 5
    ];
    global OpMcmcAdaptive
    OpMcmcAdaptive = 1;
    global OpMcmcDelayedRejection
    OpMcmcDelayedRejection = 0;
    global OpMcmcAdaptPeriod
    OpMcmcAdaptPeriod = 30;
    global OpMcmcAdaptCovarScale
    OpMcmcAdaptCovarScale = 1;
    global OpMcmcLoggingPeriod
    OpMcmcLoggingPeriod = 200;
    global OpMcmcAdaptLowerThresh
    OpMcmcAdaptLowerThresh = 0.25;
    global OpMcmcAdaptUpperThresh
    OpMcmcAdaptUpperThresh = 0.45;
    global OpMcmcAdaptLowerThreshDR
    OpMcmcAdaptLowerThreshDR = 0.45;
    global OpMcmcAdaptUpperThreshDR
    OpMcmcAdaptUpperThreshDR = 0.65;
    global OpMcmcSigmaDecreaseFact
    OpMcmcSigmaDecreaseFact = 0.9;
    global OpMcmcSigmaIncreaseFact
    OpMcmcSigmaIncreaseFact = 1.1;
    global OpMcmcDRSigmaReduceFact
    OpMcmcDRSigmaReduceFact = 0.2;
    global OpMcmcDRSigmaReduceFactAM
    OpMcmcDRSigmaReduceFactAM = 0.1;
    global OpMcmcAdaptLowerThreshAM
    OpMcmcAdaptLowerThreshAM = 0.15;
    global OpMcmcAdaptUpperThreshAM
    OpMcmcAdaptUpperThreshAM = 0.3;
    global OpMcmcCovarScaleDecreaseFact
    OpMcmcCovarScaleDecreaseFact = 20;
    global OpMcmcCovarScaleIncreaseFact
    OpMcmcCovarScaleIncreaseFact = 20;
    global OpDemcSnookerFraction
    OpDemcSnookerFraction = 0.1;
    global OpDemcThinningFactor
    OpDemcThinningFactor = 10;
    global OpDemcB
    OpDemcB = 0.0001;
end

function samp = mcSampleProposal(prevsamp)
    global data
```

```
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    samp = [];
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function val = mcEvalProposal(samp, prevsamp)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    val = 0;
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function mcDumpSamples()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    A0_i
    LI
    Vmax
    Km
```

```
end

function names = mcSampNames()
   names = "A0_i(1)";
   names = [names, "A0_i(2)"];
   names = [names, "A0_i(3)"];
   names = [names, "A0_i(4)"];
   names = [names, "A0_i(5)"];
   names = [names, "LI"];
   names = [names, "Vmax"];
   names = [names, "Km"];
   names
end

function parms = mcPackSamples()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   parms = [];
   parms = [parms reshape(A0_i, 1, 5)];
   parms = [parms LI];
   parms = [parms Vmax];
   parms = [parms Km];
end

function mcUnpackSamples(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   idx = 1;
   A0_i = reshape(parms(idx:idx+4), 5, 1); idx = idx + 5;
   LI = parms(idx); idx = idx + 1;
```

```
    Vmax = parms(idx); idx = idx + 1;
    Km = parms(idx); idx = idx + 1;
end

function parms = mcSamplePriors()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    LI = normrnd(1, 1, 0.01, 100);
    Vmax = unifrnd(-15, 5);
    Km = unifrnd(-15, 5);
    for i = firstD : lastD
        A0_i(i) = unifrnd(-10, 0);
    end
    parms = mcPackSamples();
end

function val = mcEvalPriors(parms)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    mcUnpackSamples(parms);
    val = 0.0;
    val = val + normlpdf(LI, 1, 1, 0.01, 100);
    val = val + uniflpdf(Vmax, -15, 5);
    val = val + uniflpdf(Km, -15, 5);
    for i = firstD : lastD
        val = val + uniflpdf(A0_i(i), -10, 0);
    end
end

function val = mcEvalLikelihoods(parms)
```

```
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    mcUnpackSamples(parms);
    val = 0.0;
    for i = firstD : lastD
        preds = getpreds(Vmax, Km, A0_i(i));
        for j = firstT : lastT
            if(~isnan(data(j, i)))
                val = val + normlpdf(data(j, i), preds(j), LI);
            end
        end
    end
end
```

mratlivinvitroa0.m (used for male rat liver with initial amount in vial included)

```
function tchains = runmcmc(pchains = [])
   % Driver code for MCMC analysis
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   A0_i = zeros(3, 1);

   numParms = 6
   numChains = 3
   numIts = 50000
   funcNames = ["mcInit", "mcEvalLikelihoods", "mcEvalPriors", "mcSamplePriors", "mcEvalProposal",
"mcSampleProposal"]
   updateMode = 4
   chains = mcmc(numParms, numIts, numChains, updateMode, funcNames, pchains);
   save @format=ascii @file=mcmc_results.dat chains
   tchains = chains([1:2:50000],:);
end

function mcInit()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   global OpMcmcPriorBounds
   OpMcmcPriorBounds = [...
   -10, 0
   -10, 0
   -10, 0
   0.01, 100
   -15, 5
```

```
    -15, 5
    ];
    global OpMcmcAdaptive
    OpMcmcAdaptive = 1;
    global OpMcmcDelayedRejection
    OpMcmcDelayedRejection = 0;
    global OpMcmcAdaptPeriod
    OpMcmcAdaptPeriod = 30;
    global OpMcmcAdaptCovarScale
    OpMcmcAdaptCovarScale = 1;
    global OpMcmcLoggingPeriod
    OpMcmcLoggingPeriod = 200;
    global OpMcmcAdaptLowerThresh
    OpMcmcAdaptLowerThresh = 0.25;
    global OpMcmcAdaptUpperThresh
    OpMcmcAdaptUpperThresh = 0.45;
    global OpMcmcAdaptLowerThreshDR
    OpMcmcAdaptLowerThreshDR = 0.45;
    global OpMcmcAdaptUpperThreshDR
    OpMcmcAdaptUpperThreshDR = 0.65;
    global OpMcmcSigmaDecreaseFact
    OpMcmcSigmaDecreaseFact = 0.9;
    global OpMcmcSigmaIncreaseFact
    OpMcmcSigmaIncreaseFact = 1.1;
    global OpMcmcDRSigmaReduceFact
    OpMcmcDRSigmaReduceFact = 0.2;
    global OpMcmcDRSigmaReduceFactAM
    OpMcmcDRSigmaReduceFactAM = 0.1;
    global OpMcmcAdaptLowerThreshAM
    OpMcmcAdaptLowerThreshAM = 0.15;
    global OpMcmcAdaptUpperThreshAM
    OpMcmcAdaptUpperThreshAM = 0.3;
    global OpMcmcCovarScaleDecreaseFact
    OpMcmcCovarScaleDecreaseFact = 20;
    global OpMcmcCovarScaleIncreaseFact
    OpMcmcCovarScaleIncreaseFact = 20;
    global OpDemcSnookerFraction
    OpDemcSnookerFraction = 0.1;
    global OpDemcThinningFactor
    OpDemcThinningFactor = 10;
    global OpDemcB
    OpDemcB = 0.0001;
end

function samp = mcSampleProposal(prevsamp)
    global data
    global firstT
    global lastT
```

```
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    samp = [];
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function val = mcEvalProposal(samp, prevsamp)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    val = 0;
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function mcDumpSamples()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    A0_i
    LI
    Vmax
    Km
end
```

```
function names = mcSampNames()
   names = "A0_i(1)";
   names = [names, "A0_i(2)"];
   names = [names, "A0_i(3)"];
   names = [names, "LI"];
   names = [names, "Vmax"];
   names = [names, "Km"];
   names
end

function parms = mcPackSamples()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   parms = [];
   parms = [parms reshape(A0_i, 1, 3)];
   parms = [parms LI];
   parms = [parms Vmax];
   parms = [parms Km];
end

function mcUnpackSamples(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   idx = 1;
   A0_i = reshape(parms(idx:idx+2), 3, 1); idx = idx + 3;
   LI = parms(idx); idx = idx + 1;
   Vmax = parms(idx); idx = idx + 1;
   Km = parms(idx); idx = idx + 1;
end
```

```
function parms = mcSamplePriors()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   LI = normrnd(1, 1, 0.01, 100);
   Vmax = unifrnd(-15, 5);
   Km = unifrnd(-15, 5);
   for i = firstD : lastD
      A0_i(i) = unifrnd(-10, 0);
   end
   parms = mcPackSamples();
end

function val = mcEvalPriors(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   mcUnpackSamples(parms);
   val = 0.0;
   val = val + normlpdf(LI, 1, 1, 0.01, 100);
   val = val + uniflpdf(Vmax, -15, 5);
   val = val + uniflpdf(Km, -15, 5);
   for i = firstD : lastD
      val = val + uniflpdf(A0_i(i), -10, 0);
   end
end

function val = mcEvalLikelihoods(parms)
   global data
   global firstT
   global lastT
   global firstD
```

```
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    mcUnpackSamples(parms);
    val = 0.0;
    for i = firstD : lastD
        preds = getpreds(Vmax, Km, A0_i(i));
        for j = firstT : lastT
            if(~isnan(data(j, i)))
                val = val + normlpdf(data(j, i), preds(j), LI);
            end
        end
    end
end
```

invitrohumlivsata0.m (used for mixed human liver with initial amount in vial included)

```
function tchains = runmcmc(pchains = [])
   % Driver code for MCMC analysis
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   A0_i = zeros(3, 1);

   numParms = 6
   numChains = 3
   numIts = 50000
   funcNames = ["mcInit", "mcEvalLikelihoods", "mcEvalPriors", "mcSamplePriors", "mcEvalProposal",
"mcSampleProposal"]
   updateMode = 4
   chains = mcmc(numParms, numIts, numChains, updateMode, funcNames, pchains);
   save @format=ascii @file=mcmc_results.dat chains
   tchains = chains([1:2:50000],:);
end

function mcInit()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   global OpMcmcPriorBounds
   OpMcmcPriorBounds = [...
   -10, 0
   -10, 0
   -10, 0
   0.01, 100
   -15, 5
```

```
    -15, 5
    ];
    global OpMcmcAdaptive
    OpMcmcAdaptive = 1;
    global OpMcmcDelayedRejection
    OpMcmcDelayedRejection = 0;
    global OpMcmcAdaptPeriod
    OpMcmcAdaptPeriod = 30;
    global OpMcmcAdaptCovarScale
    OpMcmcAdaptCovarScale = 1;
    global OpMcmcLoggingPeriod
    OpMcmcLoggingPeriod = 200;
    global OpMcmcAdaptLowerThresh
    OpMcmcAdaptLowerThresh = 0.25;
    global OpMcmcAdaptUpperThresh
    OpMcmcAdaptUpperThresh = 0.45;
    global OpMcmcAdaptLowerThreshDR
    OpMcmcAdaptLowerThreshDR = 0.45;
    global OpMcmcAdaptUpperThreshDR
    OpMcmcAdaptUpperThreshDR = 0.65;
    global OpMcmcSigmaDecreaseFact
    OpMcmcSigmaDecreaseFact = 0.9;
    global OpMcmcSigmaIncreaseFact
    OpMcmcSigmaIncreaseFact = 1.1;
    global OpMcmcDRSigmaReduceFact
    OpMcmcDRSigmaReduceFact = 0.2;
    global OpMcmcDRSigmaReduceFactAM
    OpMcmcDRSigmaReduceFactAM = 0.1;
    global OpMcmcAdaptLowerThreshAM
    OpMcmcAdaptLowerThreshAM = 0.15;
    global OpMcmcAdaptUpperThreshAM
    OpMcmcAdaptUpperThreshAM = 0.3;
    global OpMcmcCovarScaleDecreaseFact
    OpMcmcCovarScaleDecreaseFact = 20;
    global OpMcmcCovarScaleIncreaseFact
    OpMcmcCovarScaleIncreaseFact = 20;
    global OpDemcSnookerFraction
    OpDemcSnookerFraction = 0.1;
    global OpDemcThinningFactor
    OpDemcThinningFactor = 10;
    global OpDemcB
    OpDemcB = 0.0001;
end

function samp = mcSampleProposal(prevsamp)
    global data
    global firstT
    global lastT
```

```
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    samp = [];
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function val = mcEvalProposal(samp, prevsamp)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    val = 0;
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function mcDumpSamples()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    A0_i
    LI
    Vmax
    Km
end
```

```
function names = mcSampNames()
   names = "A0_i(1)";
   names = [names, "A0_i(2)"];
   names = [names, "A0_i(3)"];
   names = [names, "LI"];
   names = [names, "Vmax"];
   names = [names, "Km"];
   names
end

function parms = mcPackSamples()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   parms = [];
   parms = [parms reshape(A0_i, 1, 3)];
   parms = [parms LI];
   parms = [parms Vmax];
   parms = [parms Km];
end

function mcUnpackSamples(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   idx = 1;
   A0_i = reshape(parms(idx:idx+2), 3, 1); idx = idx + 3;
   LI = parms(idx); idx = idx + 1;
   Vmax = parms(idx); idx = idx + 1;
   Km = parms(idx); idx = idx + 1;
end
```

```
function parms = mcSamplePriors()
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   LI = normrnd(1, 1, 0.01, 100);
   Vmax = unifrnd(-15, 5);
   Km = unifrnd(-15, 5);
   for i = firstD : lastD
      A0_i(i) = unifrnd(-10, 0);
   end
   parms = mcPackSamples();
end

function val = mcEvalPriors(parms)
   global data
   global firstT
   global lastT
   global firstD
   global lastD
   global CCC
   global A0_i
   global LI
   global Vmax
   global Km
   global preds
   mcUnpackSamples(parms);
   val = 0.0;
   val = val + normlpdf(LI, 1, 1, 0.01, 100);
   val = val + uniflpdf(Vmax, -15, 5);
   val = val + uniflpdf(Km, -15, 5);
   for i = firstD : lastD
      val = val + uniflpdf(A0_i(i), -10, 0);
   end
end

function val = mcEvalLikelihoods(parms)
   global data
   global firstT
   global lastT
   global firstD
```

```
    global lastD
    global CCC
    global A0_i
    global LI
    global Vmax
    global Km
    global preds
    mcUnpackSamples(parms);
    val = 0.0;
    for i = firstD : lastD
        preds = getpreds(Vmax, Km, A0_i(i));
        for j = firstT : lastT
            if(~isnan(data(j, i)))
                val = val + normlpdf(data(j, i), preds(j), LI);
            end
        end
    end
end
```

In Vivo Model Files

chloroprene.model (R model code)

```
#Chloroprene PBPK Model
#Translated from the acslX model presented in Yang et al. 2012
#By Jerry Campbell 2019

States  = {
AI        ,
AX        ,
AM        ,
AMLU    ,
AMK     ,
ALU       ,
AL        ,
AK        ,
AS        ,
AR        ,
AF
};

Outputs = {
MASBAL ,
CLU       ,
CL        ,
CK        ,
CS        ,
CR        ,
CF        ,
CV  ,
CVLUM ,
ppm ,
AMP     ,
AMPLU ,
AMPK    ,
cvl ,
qcbal ,
vbal
};



Inputs = {EXPPULSE};
```

```
#BODY WEIGHT (kg)
    BW = 0.03 ;    # Body weight (kg)


#SPECIAL FLOW RATES
QPC = 29.1 ;     # Unscaled Alveolar Vent (L/h/kg^0.75)
QCC = 20.1 ;     # Unscaled Cardiac Output (L/h/kg^0.75)

#FRACTIONAL BLOOD FLOWS TO TISSUES
QLC = 0.161 ;   # Flow to Liver as % Cardiac Output (unitless)
QFC = 0.07  ;   # Flow to Fat as % Cardiac Output (unitless)
QSC = 0.159 ;   # Flow to Slow as % Cardiac Output (unitless)
QKC = 0.09  ;   # Flow to Kidney as % Cardiac Output (unitless)

#FRACTIONAL VOLUMES OF TISSUES
VLC = 0.055 ;      # Volume Liver as % Body Weight (unitless)
VLUC = 0.0073 ;   # Volume Lung as % Body Weight (unitless)
VFC = 0.1  ;         # Volume Fat as % Body Weight (unitless)
VRC = 0.08098 ;      # Volume Rapid Perfused as % Body Weight (unitless)
VSC = 0.384  ;       # Volume Slow Perfused as % Body Weight (unitless)
VKC = 0.0167 ;      # Volume Kidney as % Body Weight  (unitless)

#PARTITION COEFFICIENTS PARENT
PL = 1.26 ;    # Liver/Blood Partition Coefficient (unitless)
PLU = 2.38 ;    # Lung/Blood Partition Coefficient (unitless)
PF = 17.35 ;    # Fat/Blood Partition Coefficient (unitless)
PS = 0.59  ;   # Slow/Blood Partition Coefficient (unitless)
PR = 1.76 ;    # Rapid/Blood Partition Coefficient (unitless)
PB = 7.83 ;    # Blood/Air Partition Coefficient (unitless)
PK = 1.76 ;    # Kidney/Blood Partition Coefficient (unitless)

#KINETIC CONSTANTS
MW = 88.5  ;   # Molecular weight (g/mol)

# Metabolism in Liver
VMAXC = 7.95  ;   # Scaled VMax for Oxidative Pathway:Liver (mg/h/BW^0.75)
KM = 0.041  ;   # Km for Oxidative Pathway:Liver (mg/L)

# Metabolism in Lung
VMAXCLU = 0.18;     # Scaled VMax for Oxidative Pathway:Lung (mg/h/BW^0.75)
KMLU = 0.26;        # Km for Oxidative Pathway:Lung (mg/L)

# Metabolism in Kidney
VMAXCKid = 0.0 ;    # Scaled VMax for Oxidative Pathway:Kidney (mg/h/BW^0.75)
KMKD = 1.0  ;   # Km for Oxidative Pathway :Kidney

#DOSING INFORMATION
TSTOP = 7.0 ;     # Dosing stop time
```

```
CONC = 13.0  ;   # Initial concentration (ppm)



Dynamics {

# Scaled parameters

      QC = QCC*pow(BW,0.75)  ;      #Cardiac output
      QP = QPC*pow(BW,0.75)  ;      #Alveolar ventilation
      QL = QLC*QC    ;             #Liver blood flow
      QF = QFC*QC    ;             #Fat blood flow
      QS = QSC*QC    ;             #Slowly-perf tissue blood flow
      QK = QKC*QC    ;             #Kidney tissue blood flow

     QRC = 1-QLC-QKC-QFC-QSC  ;     #Rapily Perfused tissues
      QR = QRC*QC           ;                    #Rapily-perf tissue blood flow

      VL = VLC*BW      ;           #Liver volume
     VLU = VLUC*BW     ;             #Lung volume
      VF = VFC*BW      ;           #Fat tissue volume
      VS = VSC*BW      ;           #Slowly-perfused tissue volume
      VR = VRC*BW      ;           #Richly-perfised tissue volume
      VK = VKC*BW      ;           #kidney tissue volume

    ROBC = 1 - VLC - VLUC - VFC - VSC - VRC - VKC ;  #Rest of body un-perfused tissued for Monte Carlo
sims

# METABOLISM
    VMAX = VMAXC*pow(BW,0.75)  ;      #Maximum rate of metabolism-Liver (mg/hr/kg-BW)
  VMAXLU = VMAXCLU*pow(BW,0.75) ;     #Maximum rate of metabolism-Lung (mg/hr/kg-BW)
  VMAXKD = VMAXCKid*pow(BW,0.75)  ;   #Maximum rate of metabolism-Kidney (mg/hr/kg-BW)

# Exposure Control (mg/L)
     CIX = CONC*MW/24450 ;
      CI = CIX *EXPPULSE  ;

# Tissue Venous Concentrations (mg/L)
    CVLU = ALU/(VLU*PLU) ;
     CVL = AL/(VL*PL) ;
     CVK = AK/(VK*PK) ;
     CVS = AS/(VS*PS) ;
     CVR = AR/(VR*PR) ;
     CVF = AF/(VF*PF) ;

# Concentration in Pulmonary/Arterial and venous blood Compartments (mg/L)
     CPU = (QP*CI+(QF*CVF + QL*CVL + QS*CVS + QR*CVR + QK*CVK))/(QP/PB+QC)  ;
```

```
      CX = CPU/PB ;
      CV = (QF*CVF + QL*CVL + QS*CVS + QR*CVR + QK*CVK)/QC ;
     CPUM = CPU*1000/MW ;
      RAI = QP*CI ;
      dt(AI) = RAI ;
      RAX = QP*CX ;
      dt(AX) = RAX ;

# Amount metabolized in Liver (mg)
      RAM = VMAX*CVL/(KM+CVL) ;
      dt(AM) = RAM ;

# Amount metabolized in Lung (mg)
      RAMLU = VMAXLU*CVLU/(KMLU+CVLU) ;
      dt(AMLU) = RAMLU ;

# Amount metabolized in Kidney (mg)
      RAMK = VMAXKD*CVK/(KMKD + CVK)  ;
      dt(AMK) = RAMK ;

# Amount in Lung Compartment (mg)
      RALU = QC*(CPU-CVLU) - RAMLU ;
      dt(ALU) = RALU ;

# Amount in Liver Compartment (mg)
      RAL = QL*(CVLU-CVL) - RAM ;
      dt(AL) = RAL ;

# Amount in Kidney Compartment (mg)
      RAK = QK*(CVLU-CVK) - RAMK ;
      dt(AK) = RAK ;

# Amount in Slowly Perfused Tissues (mg)
      RAS = QS*(CVLU - CVS) ;
      dt(AS) = RAS ;

# Amount in Rapidly Perfused Tissues (mg)
      RAR = QR*(CVLU -CVR) ;
      dt(AR) = RAR ;

# Amount in Fat Compartment (mg)
      RAF = QF*(CVLU - CVF) ;
      dt(AF) = RAF ;

} # End of Dynamics

CalcOutputs {
# Mass-balance
```

```
        MASBAL = AI - AX - (AL+AM+AMLU+ALU+AK+AMK+AS+AR+AF) ;
        #Tissue Concentrations (mg/L)
          CLU = ALU/VLU ;
      CL = AL/VL ;
      CK = AK/VK ;
      CS = AS/VS ;
      CR = AR/VR ;
      CF = AF/VF ;

#Concentrations for plots
        CVLUM = CVLU*1000/MW ;  #(umol/L)
#Dose metrics
  ppm = CONC ;
  AMP = ((AM*1000/MW)/(VL*1000))/(TSTOP/24) ;
 AMPLU = ((AMLU*1000/MW)/(VLU*1000))/(TSTOP/24) ;
  AMPK = ((AMK*1000/MW)/(VK*1000))/(TSTOP/24) ;

  cvl = CVL ;

#Blood Flow balance
  qcbal = QC - QL - QF - QS - QK - QR ;

#Tissue Volume balance
   vbal = BW*(1-ROBC) - VL - VLU - VF - VS - VK - VR ;

} # End of CalcOutputs

End.
```

Mouse.R (Base model parameters for mouse)

```
#Female Mouse parameters (See Model Parameters Spreadsheet for Documentation)

parms <-c(
BW = 0.04 ,     # Body weight (kg)
QPC = 29.1 ,     # Unscaled Alveolar Vent (L/h/kg^0.75)
QCC = 20.1 ,     # Unscaled Cardiac Output (L/h/kg^0.75)

#FRACTIONAL BLOOD FLOWS TO TISSUES
QLC = 0.161 ,   # Flow to Liver as % Cardiac Output (unitless)
QFC = 0.07  ,   # Flow to Fat as % Cardiac Output (unitless)
QSC = 0.159  ,   # Flow to Slow as % Cardiac Output (unitless)
QKC = 0.09  ,   # Flow to Kidney as % Cardiac Output  (unitless)

#FRACTIONAL VOLUMES OF TISSUES
VLC = 0.055 ,   # Volume Liver as % Body Weight (unitless)
VLUC = 0.0073 ,  # Volume Lung as % Body Weight (unitless)
VFC = 0.1  ,    # Volume Fat as % Body Weight (unitless)
VRC = 0.08098 , # Volume Rapid Perfused as % Body Weight (unitless)
VSC = 0.384  ,  # Volume Slow Perfused as % Body Weight (unitless)
VKC = 0.0167 ,  # Volume Kidney as % Body Weight  (unitless)

#PARTITION COEFFICIENTS PARENT
PL = 1.26 ,     # Liver/Blood Partition Coefficient (unitless)
PLU = 2.38 ,    # Lung/Blood Partition Coefficient (unitless)
PF = 17.35 ,    # Fat/Blood Partition Coefficient (unitless)
PS = 0.59  ,    # Slow/Blood Partition Coefficient (unitless)
PR = 1.76 ,     # Rapid/Blood Partition Coefficient (unitless)
PB = 7.8 ,     # Blood/Air Partition Coefficient (unitless)
PK = 1.76 ,     # Kidney/Blood Partition Coefficient (unitless)

#KINETIC CONSTANTS
MW = 88.5  ,    # Molecular weight (g/mol)

#Revised Metabolism Constants based on Yoon report
# Metabolism in Liver
VMAXC = 99.0  , # Scaled VMax for Oxidative Pathway:Liver (mg/h/BW^0.75)
KM = 1.0  ,    # Km for Oxidative Pathway:Liver (mg/L)

# Metabolism in Lung
VMAXCLU = 99.0 ,  # Scaled VMax for Oxidative Pathway:Lung (mg/h/BW^0.75)
KMLU = 1.0  ,    # Km for Oxidative Pathway:Lung (mg/L)


# Metabolism in Kidney
VMAXCKid = 00.0 , # Scaled VMax for Oxidative Pathway:Kidney (mg/h/BW^0.75)
```

```
KMKD = 1.0  ,    # Km for Oxidative Pathway :Kidney (mg/L)


#DOSING INFORMATION
TSTOP = 7.0 ,
CONC = 0.0     # Initial concentration (ppm)
)
```

Fmouse.R (Female mouse specific model parameters)

```
#Female Mouse parameters

source('./params/mouse.R')

#Revised Metabolism Constants based on Yoon report
#QIVIE for VMAXC and VMAXCLU were based on a 40 g mouse
#which was the rounded average female weight in the 2-year bioassay

#Female mouse average body weight (rounded to g)
parms["BW"] <- 0.040      #Weighted average of the female mouse control group NTP chloroprene
bioassay

# Metabolism in Liver
parms["VMAXC"] <- 7.99   # Scaled VMax for Oxidative Pathway:Liver (mg/h/BW^0.75)
parms["KM"] <- 0.040      # Km for Oxidative Pathway:Liver (mg/L)

# Metabolism in Lung
parms["VMAXCLU"] <- 0.12  # Scaled VMax for Oxidative Pathway:Lung (mg/h/BW^0.75)
parms["KMLU"] <- 0.21      # Km for Oxidative Pathway:Lung (mg/L)

# Metabolism in Kidney
parms["VMAXCKid"] <- 00.0  # Scaled VMax for Oxidative Pathway:Kidney (mg/h/BW^0.75)
parms["KMKD"] <- 1.0      # Km for Oxidative Pathway :Kidney (mg/L)
```

Human.R (Base human model parameters)

```
#Human parameters (See Model Parameters Spreadsheet for Documentation)

parms <-c(
BW = 70.0 ,      # Body weight (kg)
QPC = 24.0 ,     # Unscaled Alveolar Vent (L/h/kg^0.75)
QCC = 16.5 ,     # Unscaled Cardiac Output (L/h/kg^0.75)

#FRACTIONAL BLOOD FLOWS TO TISSUES
QLC = 0.227 ,     # Flow to Liver as % Cardiac Output (unitless)
QFC = 0.052  ,    # Flow to Fat as % Cardiac Output (unitless)
QSC = 0.191  ,    # Flow to Slow as % Cardiac Output (unitless)
QKC = 0.175  ,    # Flow to Kidney as % Cardiac Output  (unitless)

#FRACTIONAL VOLUMES OF TISSUES
VLC = 0.0257 ,    # Volume Liver as % Body Weight (unitless)
VLUC = 0.0076 ,   # Volume Lung as % Body Weight (unitless)
VFC = 0.27  ,     # Volume Fat as % Body Weight (unitless)
VRC = 0.0533 ,    # Volume Rapid Perfused as % Body Weight (unitless)
VSC = 0.4  ,      # Volume Slow Perfused as % Body Weight (unitless)
VKC = 0.0044 ,    # Volume Kidney as % Body Weight  (unitless)

#PARTITION COEFFICIENT PARENT
PL = 2.37 ,       # Liver/Blood Partition Coefficient (unitless)
PLU = 2.94 ,      # Lung/Blood Partition Coefficient (unitless)
PF = 28.65 ,      # Fat/Blood Partition Coefficient (unitless)
PS = 1.00  ,      # Slow/Blood Partition Coefficient (unitless)
PR = 2.67 ,       # Rapid/Blood Partition Coefficient (unitless)
PB = 4.5 ,       # Blood/Air Partition Coefficient (unitless)
PK = 2.67 ,       # Kidney/Blood Partition Coefficient (unitless)

#KINETIC CONSTANTS
MW = 88.5  ,    # Molecular weight (g/mol)

#Revised Metabolism Constants based on Yoon report
# Metabolism in Liver
VMAXC = 99.0  , # Scaled VMax for Oxidative Pathway:Liver (mg/h/BW^0.75)
KM = 1.0 ,    # Km for Oxidative Pathway:Liver (mg/L)

# Metabolism in Lung
VMAXCLU = 99.0 ,  # Scaled VMax for Oxidative Pathway:Lung (mg/h/BW^0.75)
KMLU = 1.0  ,    # Km for Oxidative Pathway:Lung (mg/L)


# Metabolism in Kidney
VMAXCKid = 0.0 , # Scaled VMax for Oxidative Pathway:Kidney (mg/h/BW^0.75)
```

```
KMKD = 1.0  ,    # Km for Oxidative Pathway :Kidney (mg/L)


#DOSING INFORMATION
TSTOP = 7.0 ,
CONC = 0.0     # Initial concentration (ppm)
)
```

Mhuman.R (mixed human specific model parameters)

```
#Mixed Human Parameters

source('./params/Human.R')

#Revised Metabolism Constants based on Yoon report
# Metabolism in Liver
parms["VMAXC"] <- 14.51   # Scaled VMax for Oxidative Pathway:Liver (mg/h/BW^0.75)
parms["KM"] <- 0.031      # Km for Oxidative Pathway:Liver (mg/L)

# Metabolism in Lung
parms["VMAXCLU"] <- 0.0031  # Scaled VMax for Oxidative Pathway:Lung (mg/h/BW^0.75)
parms["KMLU"] <- 0.031      # Km for Oxidative Pathway:Lung (mg/L)


# Metabolism in Kidney (no renal metabolism)
parms["VMAXCKid"] <- 00.0  # Scaled VMax for Oxidative Pathway:Kidney (mg/h/BW^0.75)
parms["KMKD"] <- 1.0      # Km for Oxidative Pathway :Kidney (mg/L)
```

Fmouse_InVivo.R (simulates 15-day female mouse kinetic study with plots generated for the time-course blood data)

```
# Simulates the 15 day mouse exposure study
# Data collected during and after exposure on 1st day
# and at end of exposure on day 5 and 15 (1 day nose-only)
#VMAXC and VMAXCLU were scaled from the in vitro values to the
#rounded average BW reported in the study which was 20 g
#for all three concentrations

#set the working directory to where you downloaded the scripts
setwd(dirname(parent.frame(2)$ofile))


# load libraries needed to run scenario
library(deSolve)

# Model path and name
mName <- "chloroprene.model"

#load model inits file for the ode solver
source(paste0(mName,"_inits.R"))

#load the states files
#source(paste0(mPath,"states.R"))

#load the model dll
dyn.load(paste0(mName,.Platform$dynlib.ext))

#Scenario specific values
tstart <- 0.0
tstop <- 443.0
times <- seq(tstart, tstop , by=0.02)

# Physiolgical parameters path

#load the parameters
source('./params/Fmouse.R')
source('./states.R')

# timing variables for forcing functions
dstart <- tstart
dlength <- 6     #hours per day to expose
ddaysperwk <- 5  #days of week to expose
dexpend <- 19    #days of exposure
parms["TSTOP"] <- tstop
```

```
# Source forcing functions
# this loads the function forcing() in the namespace
source("forfunc.R")


#Scenario Specific Parameters
parms["BW"]<-  0.020   #measured in the study
parms["QPC"]<-  32.8   #measured in the study
parms["QCC"]<-  parms["QPC"]/1.45   #V/Q Ratio Marino et al. 2006


parms["CONC"]<-  12.3


# Run ODE
print(system.time(
  out <- ode(Y, times, func = "derivs", parms = parms, method="vode",  atol=1.0e-10, rtol=1.0e-8,
          dllname = mName, initforc="initforc", forcings=forcings,
          initfunc = "initmod", nout = length(Outputs),
          outnames = Outputs)
))

out1 <- as.data.frame(out,stringsAsFactors = F)

#Scenario Specific Exposure
parms["BW"]<-  0.020   #measured in the study
parms["QPC"]<-  49.0   #measured in the study
parms["QCC"]<-  parms["QPC"]/1.45   #V/Q Ratio Marino et al. 2006

parms["CONC"]<-  32.0



# Run ODE
print(system.time(
  out <- ode(Y, times, func = "derivs", parms = parms, method="vode",  atol=1.0e-10, rtol=1.0e-8,
          dllname = mName, initforc="initforc", forcings=forcings,
          initfunc = "initmod", nout = length(Outputs),
          outnames = Outputs)
))

out2 <- as.data.frame(out,stringsAsFactors = F)

#Scenario Specific Exposure
parms["BW"]<-  0.020   #measured in the study
parms["QPC"]<-  38.7   #measured in the study
parms["QCC"]<-  parms["QPC"]/1.45   #V/Q Ratio Marino et al. 2006
```

```
parms["CONC"]<-  90.0



# Run ODE
print(system.time(
  out <- ode(Y, times, func = "derivs", parms = parms, method="vode", atol=1.0e-10, rtol=1.0e-8,
          dllname = mName, initforc="initforc", forcings=forcings,
          initfunc = "initmod", nout = length(Outputs),
          outnames = Outputs)
))

out3 <- as.data.frame(out,stringsAsFactors = F)


#unload the model dll
dyn.unload(paste0(mName,.Platform$dynlib.ext))

#displays the plots of the observed versus predicted data

## Read the dataset to be plotted
#12.8 ppm
Dataset1 <- data.frame(cbind((c(0.5, 0.5, 0.5, 0.5, 3, 3, 3, 3, 3, 6, 6, 6, 6, 6, 6.083, 6.083, 6.083, 6.17,
6.17, 102, 102, 102, 102, 438, 438, 438, 438)),
                  (c(0.97, 0.82, 1.12, 1.22, 0.6, 2.7, 2.03, 2.1, 2.24, 2.08, 1.75, 1.53, 1.37, 1.16, 0.08, 0.09,
0.16, 0.1, 0.25, 0.17, 0.23, 0.2, 0.18, 0.28, 0.33, 0.24, 0.31))))
colnames(dataset1) <- c("time","cart")

#30 ppm
dataset2  <- data.frame(cbind((c(0.5, 0.5, 0.5, 0.5, 0.5, 3, 3, 3, 3, 3, 3, 6, 6, 6, 6, 6.083, 6.083, 6.083,
6.083, 6.083, 6.17, 6.17, 6.17, 6.17, 6.17, 6.25, 6.25, 6.25, 6.25, 6.25, 102, 102, 102, 102, 102, 438, 438,
438, 438, 438)),
                  (c(3, 2.27, 1.66, 2.08, 0.69, 3.94, 3.9, 1.52, 2.48, 1.68, 3.87, 2.26, 1.26, 4.18, 2.06, 0.46, 0.41,
0.92, 0.52, 0.77, 0.28, 0.26, 0.1, 0.12, 0.13, 0.18, 0.31, 0.69, 0.16, 0.13, 2.32, 2.26, 1.15, 1.32, 0.88, 0.75,
2.08, 1.6, 1.12, 1.45))))
colnames(dataset2) <- c("time","cart")

#90 ppm
dataset3  <- data.frame(cbind((c(0.5, 0.5, 0.5, 0.5, 0.5, 3, 3, 3, 3, 3, 6, 6, 6, 6, 6, 6.083, 6.083, 6.083,
6.083, 6.083, 6.17, 6.17, 6.17, 6.17, 6.17, 6.25, 6.25, 6.25, 6.25, 6.25, 102, 102, 102, 102, 102, 438, 438,
438, 438, 438)),
                  (c(5.92, 4.86, 4.82, 8.26, 7.69, 7.42, 12.95, 7.18, 3.46, 5.62, 9, 6.46, 7.63, 8.79, 8.12, 1.39,
3.01, 1.62, 0.92, 1.59, 0.66, 1.46, 0.67, 0.88, 0.93, 0.94, 0.63, 0.57, 0.64, 0.58, 3.73, 5.48, 4.09, 3, 6.43,
4.44, 3.64, 2.76, 3.41, 1.96))))
colnames(dataset3) <- c("time","cart")
```

```
par(mfrow=c(1,1))
plot(out1$time,out1$CV*1000/parms["MW"], type = 'l', col='red',lwd = 2,
    xlab="TIME",ylab = expression(mu*"M"), main='Mouse Study 3 Week - Day 1',
    xlim=c(0.0,7.0), ylim=c(0.0,20.0))
points(out2$CV*1000/parms["MW"]~out2$time,type = 'l',col='blue', lwd=2)
points(out3$CV*1000/parms["MW"]~out3$time,type = 'l',col='orange', lwd=2)
points(dataset1$time,dataset1$cart,type = 'p',col='red', pch=21, bg='red')
points(dataset2$time,dataset2$cart,type = 'p',col='blue', pch=21, bg='blue')
points(dataset3$time,dataset3$cart,type = 'p',col='orange', pch=21, bg='orange')


plot(out1$time,out1$CV*1000/parms["MW"], type = 'l', col='red',lwd = 2,
    xlab="TIME",ylab = expression(mu*"M"), main='Mouse Study 3 Week - Day 5',
    xlim=c(96.0,106.0), ylim=c(0.0,20.0))
points(out2$CV*1000/parms["MW"]~out2$time,type = 'l',col='blue', lwd=2)
points(out3$CV*1000/parms["MW"]~out3$time,type = 'l',col='orange', lwd=2)
points(dataset1$time,dataset1$cart,type = 'p',col='red', pch=21, bg='red')
points(dataset2$time,dataset2$cart,type = 'p',col='blue', pch=21, bg='blue')
points(dataset3$time,dataset3$cart,type = 'p',col='orange', pch=21, bg='orange')


plot(out1$time,out1$CV*1000/parms["MW"], type = 'l', col='red',lwd = 2,
    xlab="TIME",ylab = expression(mu*"M"), main='Mouse Study 3 Week - Day 19',
    xlim=c(432.0,442.0), ylim=c(0.0,20.0))
points(out2$CV*1000/parms["MW"]~out2$time,type = 'l',col='blue', lwd=2)
points(out3$CV*1000/parms["MW"]~out3$time,type = 'l',col='orange', lwd=2)
points(dataset1$time,dataset1$cart,type = 'p',col='red', pch=21, bg='red')
points(dataset2$time,dataset2$cart,type = 'p',col='blue', pch=21, bg='blue')
points(dataset3$time,dataset3$cart,type = 'p',col='orange', pch=21, bg='orange')
```

Fmouse_metric.R (simulates the bioassay study for two weeks to accumulate daily average dose metrics)

```
# Simulates female mouse for 2 weeks using mouse study protocol (6 hr/day 5 days/week)
# Uses metabolism constants from Redo of the MCMC in vitro with flux

#Set the working directory to where you downloaded the scripts
setwd(dirname(parent.frame(2)$ofile))


# Load libraries needed to run scenario
library(deSolve)

# Model path and name
mName <- "chloroprene.model"

#Load model inits file for the ode solver
source(paste0(mName,"_inits.R"))

#Load the model dll
dyn.load(paste0(mName,.Platform$dynlib.ext))

#Scenario specific values
tstart <- 0.0
tstop <- 336.0
times <- seq(tstart, tstop , by=0.01)
nend <- length(times)

#Physiolgical parameters path
#Load the parameters
source('./params/Fmouse.R')
source('./states.R')



#Timing variables for forcing functions
dstart <- tstart
dlength <- 6      #hours per day to expose
ddaysperwk <- 5   #days of week to expose
dexpend <- 12     #days of exposure
parms["TSTOP"] <- tstop

#Source forcing functions
#This loads the function forcing() in the namespace
source("forfunc.R")
```

```
#Scenario Specific Exposure
ppm <- c(12.8, 32.0, 80.0)
bw <- c(0.040, 0.040, 0.036) #wt avg bw for each exposure level rounded to g

cinh1 <- data.frame(ppm)
cinh <- lapply(cinh1, as.numeric)
outlist <- list()
ppm2 <- list()



for(i in 1:nrow(cinh1)){

  parms["CONC"] <-  ppm[i]
  parms["BW"] <-  bw[i]
  {

    out <-ode(Y, times, func = "derivs", parms = parms, method="vode", atol=1.0e-10, rtol=1.0e-8,
            dllname = mName, initforc="initforc", forcings=forcings, initfunc = "initmod", nout =
length(Outputs),
            fcontrol=list(method="linear"), outnames = Outputs)

  }
  outlist[[i]] <- out[nend,]
}
frout1 <- data.frame(outlist)
dout <- data.frame(t(frout1), row.names=paste(1:3))
rout <- cbind(dout[,c("ppm","AMP","AMPLU","AMPK")])

print("Female Mouse MCMC Redo")
print(rout)



#load the model dll
dyn.unload(paste0(mName,.Platform$dynlib.ext))
```

Human_Continuous.R (simulates continuous exposure – 7 days/week, 24 hrs/day and reports dose metrics on an average per day basis)

```
#Simulates human for 2 weeks using mouse study protocol (6 hr/day 5 days/week)
#Uses metabolism constants from Yang et al. 2012 Table 3

#Set the working directory to where you downloaded the scripts
setwd(dirname(parent.frame(2)$ofile))


#Load libraries needed to run scenario
library(deSolve)

#Model path and name
mName <- "chloroprene.model"

#Load model inits file for the ode solver
source(paste0(mName,"_inits.R"))

#Load the model dll
dyn.load(paste0(mName,.Platform$dynlib.ext))

#Scenario specific values
tstart <- 0.0
tstop <- 336
times <- seq(tstart, tstop , by=0.05)
nend <- length(times)

#Physiolgical parameters path
#Load the parameters
source('./params/Mhuman.R')   #Revised parameters from June 27 2018 update
source('./states.R')


#Timing variables for forcing functions
dstart <- tstart
dlength <- 24     #hours per day to expose
ddaysperwk <- 7  #days of week to expose
dexpend <- 100    #days of exposure
parms["TSTOP"] <- tstop

#Source forcing functions
#This loads the function forcing() in the namespace
source("forfunc.R")

#Scenario Specific Exposure
parms["CONC"]<-  12.8
```

```
#0.00028 ppm = 1.0 ug/m3
ppm <- c(0.00028, 0.001, 0.01, 0.1, 1.0, seq(2, 100, by=2))
lend <- length(ppm)
cinh1 <- data.frame(ppm)
cinh <- lapply(cinh1, as.numeric)
outlist <- list()
ppm2 <- list()



for(i in 1:nrow(cinh1)){

  parms["CONC"] <-  cinh1[i,]

  {

    out <-ode(Y, times, func = "derivs", parms = parms, method="vode",atol=1.0e-10, rtol=1.0e-8,
            dllname = mName, initforc="initforc", forcings=forcings, initfunc = "initmod", nout =
length(Outputs),
            fcontrol=list(method="linear"), outnames = Outputs)

  }
  outlist[[i]] <- out[nend,]
}
frout1 <- data.frame(outlist)
dout <- data.frame(t(frout1), row.names=paste(1:lend))
rout <- cbind(dout[,c("ppm","AMP","AMPLU","AMPK")])

#displays the output
print("Human Table 3")
print(rout)

#unload the model dll
dyn.unload(paste0(mName,.Platform$dynlib.ext))
```