**Proença et al.:**

**Insights into In Vitro Biokinetics using Virtual Cell Based Assay Simulations**

**Supplementary Data**

---

**VCBA code in R language**

```r
# DESCRIPTION OF xdot function=DIFFERENTIAL EQUATIONS

xdot <- function(t, state, parameters) {
  with(as.list(c(state, parameters)), {
    #x1 total concentration in the medium
    #x2 concentration in the headspace
    #x3 concentration inside the cells

    rmax <- 0
    Ksat <- 0
    Cdcomp <- xx1/(1 + (Ks * St) + (Kl * Lt) + (Kp * SP/V))  # Concentration of dissolved compound in mol.l^-1. Eq. 4.

    #CONDITION 1) there are viable cells
    # 1) i) there is no compound entering then: xdot3 <- -kmet*xx3 - weight_change*xx3
    # 1) ii) there is compound entering and so xdot3 is as represented in equation
    if(ncells>1) {
      W <- mcells/ncells  # Wet Weight (g)
      Vcell <- Vcells*1E6/ncells  # cm^3
      DeltaC <- Cdcomp - (xx3/(MWcomp*BCF))
      if(DeltaC==0.0) {
        rexchange <- 0.0
      } else {
        rexchange <- MWcomp*((Vcell ^ (2/3))*rda*DeltaC + rmax*DeltaC/(DeltaC+Ksat))/W
      }
      xdot3 <- rexchange - kmet*xx3 - weight_change*xx3  # Function for concentration inside the cell (g.gww^-1) ###
      # compound uptake or send it back to the medium by cells in mol.l^-1.s^-1
      cells_up <- (rexchange - kmet*xx3)*mcells/MWcomp/(1e3)*V
    } else {
      #CONDITION 2) cells are all dead
      xdot3 <- 0
      cells_up <- chemdead
    }

    # kgcomp mass transfer coefficient on the air (m.s^-1)
    # kicomp mass transfer coefficient on the water film (m.s^-1)
    # KGLcomp dimensionless gas-liquid distribution coefficient.
    Fawcomp <- kaw*(-Cdcomp*xs2/KGLcomp)  # diffusive air-water exchange (mol.m^-2.s^-1)
    Fdecomp <- kdeccomp*Cdcomp  # medium decomposition in mol.l^-1.s^-1
    Fdecompa <- kdeccomp*xs2  # headspace decomposition in mol.l^-1.s^-1
    Flosses <- Fexch*xs2  # headspace losses. For Fexch is set to 0, this is 0, but can be used when modeling cross contamination

    # Equation for total concentration the medium (mol.l^-1.s^-1)
    xdot1 <- (P*Fawcomp - Fdecomp - cells_up)

    # Equation for concentration in the headspace (mol.l^-1.s^-1)
    xdot2 <- (-Ph*Fawcomp - Fdecomp - Flosses)
    list(c(xdot1, xdot2, xdot3))
  })
}
```

---

**doi:**10.14573/altex.1812101s1

ALTEX 36(x), SUPPLEMENTARY DATA
END OF DIFFERENTIAL EQUATIONS
******************************************************************************
library(deSolve) #Program to solve differential equations
******************************************************************************

*************** DESCRIPTION OF CONSTANTS ***********************

# To return AssayVol, P, Ph and SP

Pi<pi
#Well Description in meters
diaTop<-0.00685 #Diameter of the top of the well
diaBot<-0.00635 #Diameter of the bottom of the well
Depth<-0.01076 #Depth of the well
V<-1E-7 #Volume of medium in the well

**WELL GEOMETRY**

#To return AssayVol, P, Ph and SP

vt<-(V+(Pi*H*((diaTop*0.5)^2)-(1/3)*Pi*h*((diaBot+0.5)^2))^2)/3
hh<-((V+vt)*12*h*h/(PI*diaBot*diaBot))^(1/3)
xx<-diaBot*hh*0.5/h
lli<-((h^2+(diaBot*0.5)^2)^(1/2))
SP<-Pi*xx*lli
SPi<-PI*diaBot*0.5*lli
#surface of the plastic in contact with the medium
SP<-SP+SPi+PI*((diaBot*0.5)^2)

#cell assay surface:
AS<-(Vx'2)
P<-(AS/V) #element to simplify differential equation xdot1

#head space volume (m3)
Vh=vVessel-V
Ph<-(AS/Vh) #element to simplify differential equation xdot2

**AIR-WATER EXCHANGE**

# To return kdeccomp, kdecacomp, kgcomp,KGLcomp, Klcomp and kaw
# Constant values introduced knime
kdeccomp<-2.14E-07
kdecacomp<-9.21E-07
Te<-37+273.15 #Air temperature from Celsius to K. Experiment temperature constant at 37 C.
MWair<-28.8 #Air molecular weight
MWcomp<-78.1 #Compound Molecular Weight
Press<-1 #Air Pressure in atm.
Svair<-20.1 #atomic diffusion air
Svcomp<-90.96 #atomic diffusion
RR<-8.3144 #Universal gas constant kJ/(mol.K)-1
H37<-5.62E+02 #Henry’s constant at 37 C
vb<-89.60 #molar volume at its normal boiling point (cm^3/g mol)
kgH2O<-3.0e-3 #mass transfer coefficient for water m*s-1
kLCO2<-4.1e-2 #mass transfer coefficient of CO2 in the Water side
fi<-2.6 #association factor or organic solutes diffusing into water

#Parameter for the exchange of compound between wells. Null for now.
Fexch<0

#Evaporation and Degradation equations
#Organic compound gas phase diffusion coefficient (m2/s). From Fuller 1966
valgas<-1E-7*((MWair+MWcomp)/(MWair*MWcomp))^-0.5
valgas<-(valgas/(Press*(Svair^0.33)+(Svcomp^0.33))^-2)
DGcomp<-valgas*Te(-1.75)
DGw<-1.23655e-9*Te(-1.75)
kgcomp<-kgH2O*(DGcomp/DGw)^0.67
KGLcomp<-H37/(RR*Te) #Gas-Liquid partition Coefficient (dimensionless)
muw <- 0.6913  # simplified water viscosity in cP at 37°C.
denw <- 0.933   # simplified water density in g/cm³ at 37°C.
Fix <- 7.4E-12 * (k*MWcomp) * 0.5 / (vb)^0.6
 DLcomp <- F*Te/muw  # liquid phase diffusion coefficients (m²/s).
SCAcomp <- muw * 1e-3 / (1e3 * denw * DLcomp)  # Schimdt number.
kcomp <- kCO2 * (SCAcomp / 600) ^ (-0.5)  # mass transfer coefficient on the water film.
kaw <- (kgcomp*KGLcomp*kcomp) / (kcomp+kgcomp*KGLcomp)  # mass transfer coefficient.

######### BCF AND PARTITION #########
# To return mass fractions and densities for each component Kp, Ks, KL, KP, BCF and rad

logkow <- -2.31  # logarithm of partition octanol-water

# mass fractions in % weight
faq <- 0.614  # 3T3 cell line aqueous fraction
fl <- 0.142  # 3T3 cell line lipidic fraction
fp <- 0.244  # 3T3 cell line proteic fraction

# densities in g/L
rhoaq <- 1000  # aqueous phase density
rhoL <- 900  # lipid phase density
rhoP <- 1350  # proteic phase density

# Lipid and protein in medium serum. Mind the % of serum supplementing the medium
S0 <- 0.0234  # protein content for 5% serum
L0 <- 0.08  # lipid content for 5% serum

# Equations for partition

# plastic well partition
Kp <- 10^(0.97*logkow - 6.94)

# serum well partition
if (logkow < 1.09) {
  vals <- 1.31
} else if (logkow >= 1.09 && logkow <= 4.6) {
  vals <- 0.57*logkow + 0.69
} else if (logkow > 4.6) {
  vals <- logkow - 1.3
}

Ks <- 10^(-vals - 1.178)

# lipid partition
KL <- 170.4
SC <- 4.4
KL<KL*KI
KP <- SC*Ks

# Bioconcentration factor
# Does not contain lipid and protein concentration as these parameters are already contained in KL and KP
BCF <- (faq/rhoaq) + (fl/rhoL) + (fp/rhoP)

# Cell Permeability Equations
logp <- 1.1711 + 0.98*logkow - 0.0011*MWcomp  # QSAR for log cell permeability (cm·h⁻¹)
Per <- 10^logp  # Cell permeability still in cm·h⁻¹
rda <- Per/36000  # Rate of uptake=Cell permeability from cm·h⁻¹ to L·cm⁻²·s⁻¹

# Herein is considered rate of uptake is the same of elimination. If this assumption change parameters have to be revised
## Metabolism

kmet<-0 #3T3 are considered metabolic incompetent cells

### Constants Description

```r
coreModel< function(ci,cell_i,numCells,nec,kt) {

# Parameters for Leslie matrix. Already structured in arrays of 1 row and 4 columns
di<-c(9.63,3.65,3.45,2.26) # duration at each stage in min
zi<-c(0.005,0.005,0.04,0.04) # mortality at each stage

di<-array(di,dim=c(1,4))
zi<-array(zi,dim=c(1,4))

# Initial values for the time loop. Once the program runs once the loop, these values stop being used and actualised for the time run(kl)
numCells<-1680 # initial cell number
ncells<-numCells
N<-c(50.7,19.2,18.1,9.2)*ncells/100 # fraction of cells in each cell cycle phase
N<-array(N,dim=c(4))
mcells<-mcells_old
Lt<-L0
St<-S0
weight_change<-0
chemdead<-0
x1<-ci
x2<-0
x3<-cell_i

# Definition of the Duration
totalTime<-48 # Time in h
kl_fin<-totalTime # in h
val<-array(kl_fin) # array for cell growth
dead_cells<-array(kl_fin) # array for cell death
```

### Time Cycle

```r
for (kl in 1:kl_fin) {

t<-seq((kl-1)*3600,kl*3600,by=1) # time in seconds but making steps of 60 s

# Differential equations solving
parameters<-c(Ks=Ks,St=St,Kl=Kl,Kp=Kp,Lt=Lt,SP=SP,V=V,kgcomp=kcomp,KGLcomp=KGLcomp,kcomp=kcomp,ncells=ncells,mcells=mcells,ccells=vcells,faq=faq,rhoa=rhoa,phi=phi,kdecacomp=kdecacomp,kdeccomp=kdeccomp,Ph=Ph)
state<-c(xx1=x1,xx2=x2,xx3=x3)
out<-ode(y=state,times=t,func=xdot,parms=parameters,method="radau",atol=1e-4,rtol=1e-4,hmax=1)

out_row<-length(out[,1])
out_col<-length(out[1,])
out<-matrix(data=out,nrow=out_row,ncol=out_col)
```

### Mortality NEC and Kt

```r
cq<-mean(out[,4])
val2<-cq - nec
val1<-max(0,val2)
zak<-x+kt=val1
pi<-exp(-zak)
gamma<--(1-pi)*pi*(di-1)/(1-pi*di)
PS<-pi*(1-gamma)
GS<-pi*gamma
```

### Density dependence HepG2

```r
if(cellType =="HepG2") {
    #Fii<-3.8062
    #dvol<-55.0676e-5
```
#F<-exp(-ncells/dvol)
# else {
# F<-1.026 #optimised fecundity
#
# Leslie Matrix#
numSteps<-4 #how many cell phases the cell line has; Ex:3T3 has 4 and HepaRG 1.

if(numSteps==1) {
  dead_cells[k]<-N[1]*(1-PS[1]+GS[1])
  LE<-c(PS[1])
  LE<-array(LE,dim=c(1,1))
  #The PS and GS for other cell phases are set to 0 so in valdead it will not count with these phases
  PS[2:4]<-0
  GS[1:4]<-0
}
  #Because the cell we are using here is 3T3 and it has 4 phases, then it will run this option
if(numSteps==4) {
  LE<-c(PS[1],GS[1],0,0,0,PS[2],GS[2],0,0,0,PS[3],GS[3],F,0,0,PS[4])
  LE<-array(LE,dim=c(4,4))
  N<-LE%*%N
}
  #Number of cells at each moment#
  val[k]<-sum(N)
  ncells<-val[k]
  #When all cells die
  if(ncells<1) {
    ncells<-0
    out_[,4]<-0
    Wcells<-0
    mcells<-0
    Vcells<-0
  } else {
    Wcell<-mcells/ncells
    weight_change<-(mcells-mcells_old)/mcells/ncells/3600
    mcells_old<-mcells
    #So for the new cycle the herein calculated mcells will be the mcells(ki-1)=mcells_old
    #Return of chemical, lipid and protein to medium from cells death#
    chemdead<-dead_cells[k]*out_[out_row,4]*Wcell*1E-3/MWcomp/V
    mcells_old<-mcells
    #So for the new cycle the herein calculated mcells will be the mcells(ki-1)=mcells_old
    #Description of the procedures for different Modes, can only run one mode at a time.
    #Description experimental viability curve

L<-(Lt*$valdead)**1E-3/V
#albumin MW= 66400 g/mol
St<-(St*$valdead)/66400/V
#output of the differential equations and for cycle (concentration in air, medium and intracellularly)
x0<out_[out_row,1]
x1<out_[out_row,2]
x2<out_[out_row,3]
x3<out_[out_row,4]
# Start new cycle
return(c(x1,x2,x3,ncells,Lt,St))

######################################################################## END LOOP ##################################################
# # # Description of the procedures for different Modes, can only run one mode at a time.

#Description experimental viability curve
# Mode: OPTIMIZATION

valerror <- function(initialValues) {
  sol <- array(0, dim=c(NumberRow,6))
  viaT <- array(0, dim=c(NumberRow,1))
  NEC <- initialValues[1]
  kt <- initialValues[2]
  valerror <- 0.0 # Initial value of error value
  # Loop for each experimental concentration-viability comparison with predicted
  for(i in 1:NumberRow) {
    sol[i,1:6]=coreModel(ci=ExpCon[i],cell_i=0,numCells=numCells,nec=NEC,kt=kt)
    viaT[i] <- sol[i,4]/sol[1,4]*100 # Final viability in %
    valerror <- valerror + (viaT[i] - ExpVia[i])^2
  }
  print(c(error=valerror,nec=NEC,kt=kt))
  return(valerror)
}

# optim is a defined R function for optimization.
# First input is the initial values to be optimised.
# Second is the function to be minimised, function that has discriminated the values to be optimised. Then the method of optimization and the lower values possible.
# This optimization can take days. An initial testing of approximated values can be made manually and ranges adjusted so the optimization does not drift to non suitable values
output <- optim(c(0.03,0.135),vallerror,method='L-BFGS-B',lower=c(0.007,0.07),upper=c(0.09,0.25))

### SELECTION OF DIFFERENT MODES ###

modeSel <- "Single Exposure"
if(modeSel=="Single Exposure") {
  final <- singleExposure()
  all_diss <- alldiss(Lt=final[,5],St=final[,6])
  cdiss <- as.double(all_diss[1])
  disscss <- as.double(all_diss[2])
  disscK <- as.double(all_diss[3])
  disscP <- as.double(all_diss[4])
  caq <- final[3]/MWcomp/BCF
  f <- data.frame(name="Benzene",Dissolved=(final[1,1]/cdiss),protein=(final[1,1]/disscss),lipid=(final[1,1]/disscK),plastic=(final[1,1]/disscP),SCF=BCF,ct=final[1,1],head space=final[1,2],conInside=final[1,3])
}

#### MODE: Description of plots ####

## Chart viability vs concentration (experimental & predicted) ##
conc <- seq(0.1, 1.3, by=0.3E) # Concentrations to be simulated

# Description of function for plots#
Graphs <- function(ci=c)(
  FData <- array(0, dim=c(48,6))
  FData[,] <- coreModel(ci=ci,cell_i=0,numCells=1680,nec=0.00063,kt=35.001) # Adjust to the optimised nec and kt
  ViaCells <- FData[,4]
  #% Viability along C'
```r
RelNCells = FData[,4]/1680 # Relative Nr cells
all_diss = all_diss(FData[,5],St=FData[,6])
DissC = (FData[,1]/all_diss[1]) # Dissolved concentration
Intra = FData[,3]
LipC = (FData[,1]/all_diss[3])
ProteC = (FData[,1]/all_diss[2]) # Intracellular Concentration
Fin = cbind(ViaCells, RelNCells, DissC, Intra, LipC, ProteC)
return(Fin)
```

```r
# Defining cell viability in %#
PredV <- function(CI = CI) {
  Pred = array(0, dim = c(1, length(conc)))
  Pred = Graphs(ci = CI)[48, 1]/Graphs(ci = conc[1])[48, 1]*100
  return(Pred)
}
```

```r
## Charts frame##
par(mfrow = c(3, 2))

# # Charts frame ##
par(mfrow = c(3, 2))

# # plot cell viability vs concentrations ##
plot(ExpCon, ExpVia, ylim = c(0, 100), type = "p", pch = 19, col = 2, cex = 1.3, xlab = "Concentration (M)", ylab = "% Viability", cex.lab = 1.25)
arrows(ExpCon, ExpVia - StDExp, ExpCon, ExpVia + StDExp, col = 2, length = 0.05, angle = 90, code = 3, cex = 1) # Error bars
PredVia = array(0, dim = c(1, length(conc)))
for (con in 1:length(conc)) {
  PredVia[1, con] = PredV(CI = conc[con])
}
lines(conc, PredVia, lwd = 2)

# # Definition of Color Gradient##
colvector <- colorRampPalette(c("lightskyblue", "dodgerblue4"))
Colvector <- array(colvector(length(conc)), dim = c(1, length(conc)))

## plot N cells vs Time##
plot(1:48, Graphs(ci = conc[1])[, 2], type = "l", ylim = c(0, 6), xlab = "time (s)", ylab = "Relative n°Cells ", cex.lab = 1.25)
for (con in 1:length(conc)) {
  lines(1:48, Graphs(ci = conc[con])[, 2], col = Colvector[con], lwd = 1.5)
}

## plot Dissolved concentration vs time##
plot(1:48, Graphs(ci = conc[1])[, 3], type = "l", xlab = "time (s)", ylab = "Dissolved Concentration (M) ", cex.lab = 1.25)
for (con in 1:length(conc)) {
  lines(1:48, Graphs(ci = conc[con])[, 3], col = Colvector[con], lwd = 1.5)
}

## plot Intracellular concentration vs time##
plot(1:48, Graphs(ci = conc[1])[, 4], type = "l", xlab = "time (s)", ylab = "Intracellular Concentration (g/g ww) ", cex.lab = 1.25)
for (con in 1:length(conc)) {
  lines(1:48, Graphs(ci = conc[con])[, 4], col = Colvector[con], lwd = 1.5)
}

## plot Lipid concentration vs time##
plot(1:48, Graphs(ci = conc[1])[, 5], type = "l", xlab = "time (s)", ylab = "[Chemical] bound to Lipid (M) ", cex.lab = 1.25)
for (con in 1:length(conc)) {
  lines(1:48, Graphs(ci = conc[con])[, 5], col = Colvector[con], lwd = 1.5)
}

## plot Protein concentration vs time##
plot(1:48, Graphs(ci = conc[1])[, 6], type = "l", xlab = "time (s)", ylab = "[Chemical] bound to Protein (M) ", cex.lab = 1.25)
for (con in 1:length(conc)) {
  lines(1:48, Graphs(ci = conc[con])[, 6], col = Colvector[con], lwd = 1.5)
}
```