
Biokmod: A *Mathematica* toolbox for modeling Biokinetic Systems

Author: Guillermo Sanchez: guillermo@usal.es, <http://web.usal.es/guillermo/>

| *Last update:* 2010-04-15

This notebook is an updated version of Sánchez G; Biokmod: A Mathematica toolbox for modeling Biokinetic Systems". *Mathematica in Education and Research*: 10 (2) 2005. ISSN/ISBN: 1096-3324

```
In[1]:= $Version
```

```
Out[1]= 7.0 for Microsoft Windows (32-bit) (February 18, 2009)
```

Biokmod is a Mathematica toolbox for modeling biokinetic systems. It includes the SysModel package to solve system of ordinary linear differential equations (SOLDE) with special application to compartmental and physiological models. It can also be applied to fit the transfer rates using experimental data.

Inicalization

1. Starting with the package

Loading the package

We will need

```
In[2]:= Needs["PlotLegends`"]
```

and the package SysModel included in BIOKMOD (<http://web.usal.es/~guillermo/biokmod.htm>).

```
In[3]:= Needs["Biokmod`SysModel`"]
```

```
SysModel, version 1.4.1 2006-12-19
```

A compartmental analysis overview

Compartmental analysis has applications in clinical medicine, pharmacokinetics, internal dosimetry, nuclear medicine, ecosystem studies and chemical reaction kinetics [1] [2] [3]. It can be described as the analysis of a system in which the system is separated into a finite number of component parts which are called compartments. Compartments interact through the exchange of species. Species may be a chemical substance, hormone, individuals in a population and so on. A compartmental system is usually represented by a flow diagram or a block diagram.

We adopt the convention of representing compartments with circles or rectangles. The flow into or out of the compartments is represented by arrows. The i th compartment of a system of n compartments is labelled i and the size (amount or content) of the component in compartment i as $x_i(t)$. The exchange between compartments, or between a compartment and the environment is labeled k_{ij} , where ij represents the flow from i to j

(if there is not ambiguity k_i will be used in place of k_{ij} , also, if $i < 10$ and $j < 10$ k_{ij} will be written in place of $k_{i j}$). The environment represents the processes that are outside the system and is usually represented by zero, so k_{i0} is the fractional excretion coefficient from the i th compartment to the outside environment. If we suppose that the substance introduced into the system is a radioactive isotope, we must consider the radioactive decay, which is given by a constant rate represented by λ (this constant is specific for each isotope). The decay constant can be interpreted by an equal flow going out of the system in each compartment. The input from the environment into the j th compartment is called $b_j(t)$. With regards to the environment, we only need to know the flow, $b_j(t)$, into the system from the outside. The $k_{i j}$ are called fractional transfer rate coefficients, they are usually assumed constants but in some cases they can be function of the time (that is $k_{i j}(t)$).

So in the two compartment model of Fig. 0, $b_1(t)$ is the input from environment into compartment 1, k_{12} is the transfer rate coefficient from compartment 1 to compartment 2 and k_{21} from compartment 2 to 1, k_{20} the transfer rate coefficient from compartment 1 to the environment (output), and $x_1(t)$ and $x_2(t)$ represents the quantities in compartment 1 and 2 at time t .

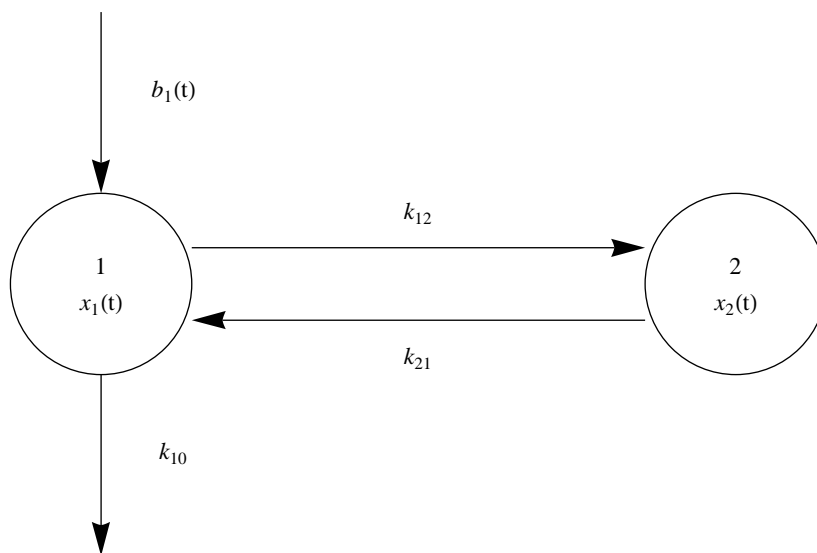


Fig1

Two compartment model with input and output from compartment 1.

The variables $x_1(t)$ and $x_2(t)$ are called the state variables of the system and their evolution in time is described by a system of ordinary differential equations (SODE). In this case the SODE is

eqn1

$$\begin{aligned}\dot{x}_1(t) &= -k_{10} x_1(t) - k_{12} x_1(t) + k_{21} x_2(t) + b_1(t) \\ \dot{x}_2(t) &= k_{12} x_1(t) - k_{21} x_2(t)\end{aligned}\quad (1)$$

Figure 2 shows a more complex compartmental model that is represented by eqn (2) where we have assumed also a radioactive decay with a disintegration constant λ .

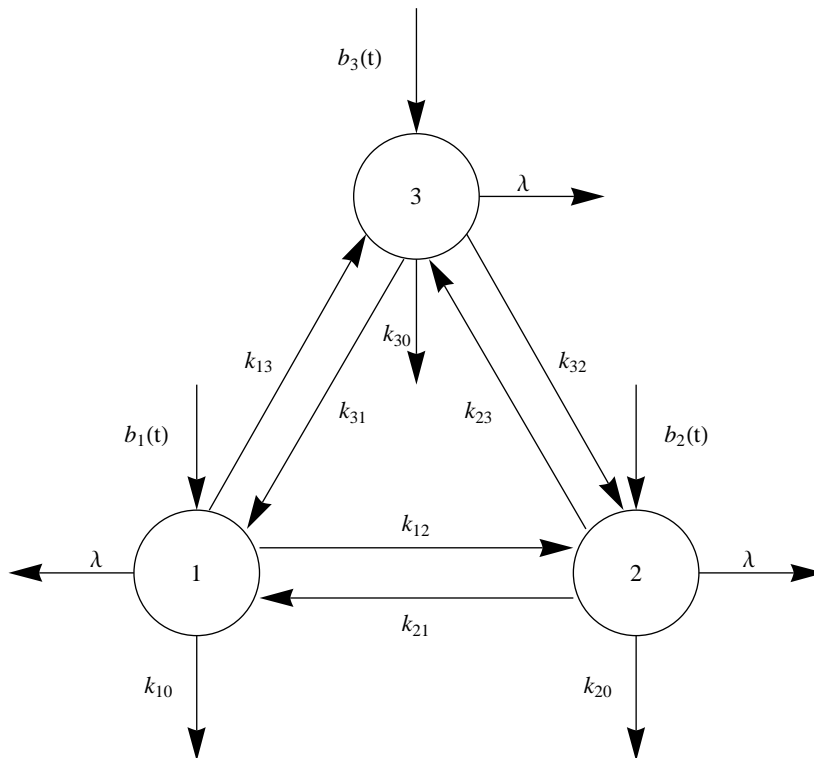


Fig2

The general multi-input multi-output (MIMO) three-compartment model.

eqn2

$$\begin{aligned}\dot{x}_1(t) &= -(\lambda + k_{10} + k_{12} + k_{13}) x_1(t) + \\ &\quad k_{21} x_2(t) + k_{31} x_3(t) + b_1(t) \\ \dot{x}_2(t) &= k_{12} x_1(t) - (\lambda + k_{20} + k_{21} + k_{23}) x_2(t) + \\ &\quad k_{32} x_3(t) + b_2(t) \\ \dot{x}_3(t) &= k_{13} x_1(t) + k_{23} x_2(t) - \\ &\quad (\lambda + k_{30} + k_{31} + k_{32}) x_3(t) + b_3(t)\end{aligned}$$

$$\dot{x}(t) = A \cdot x(t) + b(t) \text{ for } t \geq 0 \quad (2)$$

where $x(t)$ is a column vector representing the compartment content,

$$x(t) = \{x_1(t), x_2(t), x_3(t)\}^T$$

$b(t)$ is a column vector representing the inputs to compartment 1, 2, and 3,

$$b(t) = \{b_1(t), b_2(t), b_3(t)\}^T$$

and A is called the compartmental matrix and it can be obtained using the package function **CompartmentMatrix**.

<p>CompartmentMatrix[n, {trans-coeffs}, λ]</p>	<p>gives the matrix of coefficients of a compartmental system where n is the number of compartments of the system, <i>trans-coeffs</i> are the transfer rates and λ is the radioactive decay constant (by default $\lambda = 0$, which means that it is not a radioactive substance.). The transfer rates $\{k_{ij}\}$ from compartment i to compartment j are written: $\{\{1, 2, k_{12}\}, \dots, \{i, j, k_{ij}\}, \dots\}$ (by default $k_{ij} = 0$).</p>
-------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

So, the compartmental matrix of eqn (2) can be obtained as follows:

```
In[4]:= CompartmentMatrix[3, {{1, 0, k10}, {1, 2, k12},
    {1, 3, k13}, {2, 0, k20}, {2, 1, k21}, {2, 3, k23},
    {3, 0, k30}, {3, 1, k31}, {3, 2, k32}}, λ] // MatrixForm
```

```
Out[4]//MatrixForm=
```

$$\begin{pmatrix} -\lambda - k_{10} - k_{12} - k_{13} & k_{21} & k_{31} \\ k_{12} & -\lambda - k_{20} - k_{21} - k_{23} & k_{32} \\ k_{13} & k_{23} & -\lambda - k_{30} - k_{31} - k_{32} \end{pmatrix}$$

In those circumstances where the coefficients, a_{ij} , of a SODE are associated with physiologically meaningful values that correspond to the measured physiological parameter, or may be a function of them, a physiological model [4-5] rather than compartmental model is required. For these cases we will directly give the values a_{ij} to the A matrix, instead of k_{ij} , using the command **CoefMatrix**.

CoefMatrix[n, {coeffs}] gives the matrix of coefficients for n retention variables where coeffs are coefficients of the matrix: $\{ \{1, 2, a_{12}\}, \dots, \{i, j, a_{ij}\}, \dots \}$ for the ith row and jth column (by default $a_{ij}=0$).

Equation (3) represents a physiological model [4] where the drug is transported from V_p (vascular volume), by perfusate flow Q , to V_{Tu} (tissue water space) across a permeability barrier (permeability-surface product PS) and its binding is described by binding/unbinding constants k_{on}/k_{off} .

eqn3

$$\begin{aligned} \dot{x}_1(t) &= -\frac{Q+PS}{V_p} x_1(t) + \frac{PS}{V_p} x_2(t) + \frac{Q}{V_p} b_1(t) \\ \dot{x}_2(t) &= \frac{PS}{V_{Tu}} x_1(t) - \left(\frac{PS}{V_{Tu}} + k_{on} \right) x_2(t) + k_{off} \frac{V_{Tb}}{V_{Tu}} x_3(t) \\ \dot{x}_3(t) &= k_{on} \frac{V_{Tu}}{V_{Tb}} x_2(t) - k_{off} x_3(t) \end{aligned} \quad (3)$$

The matrix A for this model, which will be used later, is

```
In[5]:= physiomodel =
  CoefMatrix[3, {{1, 1, - (Q/Vp + PS/Vp)}, {1, 2, PS/Vp},
    {2, 1, PS/Vtu}, {2, 2, - (PS/Vtu + kon)}, {2, 3, koff Vtb/Vtu},
    {3, 2, kon Vtu/Vtb}, {3, 3, -koff}}];
```

The patterns that we have seen can be expanded to systems of n compartments or n state variables (in the case of physiological models). The equation for any compartment i is given by:

eqn4

$$\begin{aligned} \dot{x}(t) &= A \cdot x(t) + b(t) \quad t \geq 0 \\ x(0) &= x_0 \end{aligned} \quad (4)$$

where $x(0)$ is the vector of the initial conditions:

$$x(0) = \{x_1(0), x_2(0), \dots, x_n(0)\}^T$$

The package has the function **ShowODE** for representing the differential equations.

ShowODE[Matrix-Coeffs, incond, input, t, x]	gives the SODE for a compartmental or physiological model in classical mathematical notation; Matrix-Coeffs is the coefficients matrix; incond are the initial conditions $\{x_1(0), x_2(0), \dots, x_n(0)\}$; input are the inputs $\{b_1(t), b_2(t), \dots, b_n(t)\}$ in compartment $\{1, \dots, n\}$; x is the symbol that represents the retention variables.
----------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

So, the eqn (3), after taking into account that the input function is $\left\{\frac{Q}{V_p} b_1(t), 0, 0\right\}$ and the initial condition is $\{0, 0, 0\}$, can be written as follows:

```
In[6]:= ShowODE[physiomodel, {0, 0, 0}, { $\frac{Q}{V_p}$  b1[t], 0, 0}, t, x] //
```

```
TableForm // TraditionalForm
```

```
Out[6]//TraditionalForm=
```

$$\begin{aligned}(x_1)'(t) &= \frac{Q b_1(t)}{V_p} + x_1(t) \left(-\frac{PS}{V_p} - \frac{Q}{V_p} \right) + \frac{PS x_2(t)}{V_p} \\(x_2)'(t) &= \frac{k_{\text{off}} x_3(t) V_{\text{Tb}}}{V_{\text{Tu}}} + x_2(t) \left(-k_{\text{on}} - \frac{PS}{V_{\text{Tu}}} \right) + \frac{PS x_1(t)}{V_{\text{Tu}}} \\(x_3)'(t) &= \frac{k_{\text{on}} x_2(t) V_{\text{Tu}}}{V_{\text{Tb}}} - k_{\text{off}} x_3(t) \\x_1(0) &= 0 \\x_2(0) &= 0 \\x_3(0) &= 0\end{aligned}$$

```
In[7]:= ClearAll[physiomodel];
```

The iodine model

In some examples of this paper we will use the iodine biokinetic model represented in the Fig. 03 where compartment 1 is the blood, compartment 2 is the thyroid, compartment 3 is the rest of the body, compartment 4 is the bladder, $3 \rightarrow 0$, i.e. a transfer from compartment 3 to the environment, represents the output to the gastro intestinal tract (GIT) and $4 \rightarrow 0$ represents the output, via urine excretion, to the environment.

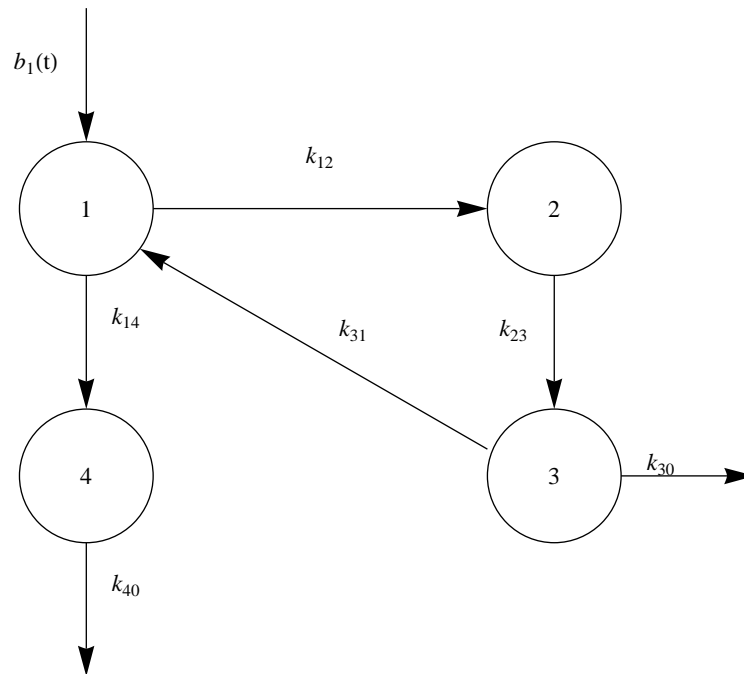


Fig3

Fig. 03 Iodine model from ICRP 78.

According to the ICRP 78, of the iodine entering the blood, 30 % goes to the thyroid, where it is cleared with a biological half-life of 80 days (The half-life, denoted by $T_{1/2}$, and transfer rate, k , are related by $k = \text{Log}(2)/T_{1/2}$ being Log the natural logarithm). The remaining iodine in the blood (70 %) is cleared to the bladder. The half-life of the iodine in the blood is 0.25 days. The iodine leaving the thyroid is assumed to be uniformly distributed in the "Rest of body" (referred in the diagram as compartment 3) where it is cleared with a half-life of 12 days. Of the iodine leaving compartment 3, the 80 % returns to the blood and the remainder 20 % is excreted to feces. The iodine from bladder is excreted as urine with a the transfer rate of 12 day^{-1} . Therefore, the transfer rates, in days^{-1} , are $k_{12} = 0.3 \text{Log}(2)/0.25$, $k_{14} = 0.7 \text{Log}(2)/0.25$, $k_{23} = \text{Log}(2)/80$, $k_{30} = 0.2 \text{Log}(2)/12$, $k_{31} = 0.8 \text{Log}(2)/12$ and $k_{40} = 12$.

Note Log means natural logarithm

In the example that follows we will refer to iodine 131 which has a radioactive half-life of 8.02 days. This means that radioactive decay constant λ is $\text{Log}(2)/8.02$ in day^{-1} . Using

these values we obtain the iodine 131 compartmental matrix.

```
In[8]:= iodine131matrix = CompartMatrix[4,
  {{1, 2, 0.3 Log[2] / 0.25}, {1, 4, 0.7 Log[2] / 0.25},
  {4, 0, 12.}, {2, 3, Log[2] / 80.}, {3, 0, 0.2 Log[2] / 12},
  {3, 1, 0.8 Log[2] / 12}}, Log[2] / 8.02];
```

2. Solving biokinetic models

Main package functions

The solution of eqn (4) when k_{ij} are constants is given by eqn (5). It can also be solved using the Laplace transform [2]. If some k_{ij} are functions of the time, that is $k_{ij}(t)$, then eqn (0) must be solved using a numeric method.

eqn5

$$x(t) = x_0 e^{At} + \int_0^t e^{A(t-\tau)} b(\tau) d\tau \quad (5)$$

The package has the **SystemDSolve** function to compute eqn (0) analytically where k_{ij} are constants. It gives the option to chose the method for solving this equation.

SystemDSolve Matrix-Coeffs, incond, input, t, t₁, x, options]	gives the analytical solution $x(t)=\{x_1(t), \dots, x_n(t)\}$ at time t_1 of a SODE $\dot{x}(t) = A \cdot x(t) + b(t)$, with initial conditions $x(0) = \text{incond} = \{c_1, \dots, c_n\}$, coefficients matrix $A = \text{Matrix-Coeffs} = \{\{a_{11}, \dots, a_{1n}\}, \dots, \{a_{n1}, \dots, a_{nn}\}\}$; and $b(t) = \{b_1(t), \dots, b_n(t)\}$. The method to solve the equation can be chosen with options.
-----------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The default method, **Method**→**Automatic**, is usually the fastest. The default method solves eqn (4) by applying eqn (5) except if b is a constant, eqn (6) is then used.

eqn6

$$x(t) = x_0 e^{At} + b \int_0^t e^{A\tau} d\tau \quad (6)$$

If $b(t) = \{\sum a_i \exp(b_i t), \sum a_j \exp(b_j t), \dots, \sum a_p \exp(b_p t)\}^T$, where some elements may be zero, the rule given below by eqn (7) is applied to solve the integral term of eqn (0). This method computes the matrix-exponential $e^{A t}$, given by $e^{A t} = S e^{\Lambda t} S^{-1}$, where Λ is the diagonal matrix of eigenvalues and S is the eigenvectors matrix.

eqn7

$$\int_0^t e^{b(t-\tau)} (a e^{c\tau}) d\tau = a \int_0^t e^{b t + (c-b)\tau} d\tau = \frac{a}{b-c} (e^{bt} - e^{ct}) \text{ if } b \neq c \quad (7)$$

If Method→EigenValues is entered eigenvalues-eigensystem method is applied to evaluate the matrix-exponential with the *Mathematica* function MatrixExp. If Method→LaplaceTransform is entered Laplace Transform method is used [2] [6]. Applying Laplace transforms to eqn (7) gives

eqn8

$$X(s) = (sI - A)^{-1} x_0 + (sI - A)^{-1} B(s) \quad (8)$$

where $X(s)$ and $B(s)$ are the Laplace transforms of $x(t)$ and $b(t)$. Then eqn (8) can be solved by evaluating the inverse transformation:

eqn9

$$x(t) = \mathcal{L}^{-1}\left((sI - A)^{-1} x_0\right) + \mathcal{L}^{-1}\left((sI - A)^{-1} B(s)\right) \quad (9)$$

Comparing eqn (8) and eqn (9) and applying the Laplace transform convolution property $\mathcal{L}^{-1}\left((sI - A)^{-1} B(s)\right) = \mathcal{L}^{-1}\left((sI - A)^{-1}\right) \mathcal{L}^{-1}(B(s))$ it may be seen that the matrix-exponential is given by,

eqn10

$$e^{A t} = \mathcal{L}^{-1}\left(\left(sI - A\right)^{-1}\right) \quad (10)$$

Acute inputs. If the inputs are acute (also called single or impulsive inputs) $b(t) = x_0$ at $t = 0$ and $b(t) = 0$ for $t \geq 0$ then we have an SODE given by eqn (11). Note that is equivalent to eqn (6) taking x_0 as initial condition and $b(t) = 0$ for $t \geq 0$.

eqn11

$$\begin{aligned} \dot{x}(t) &= A \cdot x(t), \quad t \geq 0 \\ x(0) &= x_0 \end{aligned} \quad (11)$$

The solution is

eqn12

$$x(t) = x_0 e^{At}$$

For this case the package provides the function **AcuteInput**.

<p>AcuteInput[Matrix-Coeffs, incond, t₁, x]</p>	<p>gives the analytical solution $x(t) = \{x_1(t), \dots, x_n(t)\}$ at time t_1 of a SODE $\dot{x}(t) = A \cdot x(t)$, with initial conditions, incond, $x(0) = \{c_1, \dots, c_n\}$, coefficient matrix, Matrix-Coeffs, $A = \{\{a_{11}, \dots, a_{1n}\}, \dots, \{a_{n1}, \dots, a_{nn}\}\}$.</p>
------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sometimes we need to solve models given by eqn (0) where the transfer rates are function of the time or the inputs of the model are too complicate. In these cases we should use the command **SystemNDSolve**. This function applies the *Mathematica* function **NDSolve**.

<p>SystemNDSolve[Matrix-Coeffs, incond, input, {t, t_{min}, t_{max}}, t₁, x, options]</p>	<p>gives the numerical solution $x(t) = \{x_1(t), \dots, x_n(t)\}$, in the interval t_{\min}, t_{\max}, at time t_1, of $\dot{x}(t) = A \cdot x(t) + b(t)$, with the initial conditions, incond, $x(0) = \{c_1, \dots, c_n\}$, coefficient matrix, Matrix-Coeffs, $A = \{\{a_{11}, \dots, a_{1n}\}, \dots, \{a_{n1}, \dots, a_{nn}\}\}$ and $b(t) = \{b_1(t), \dots, b_n(t)\}$; options are the options of NDSolve.</p>
--------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Models with constant transfer rates

In this subsection we will show a few examples of the iodine 131 model given by the compartmental matrix `iodine131matrix` where the transfer rate coefficients k_{ij} are constants.

Single input. Let's suppose an impulsive-injection of 1200 Bq of ^{131}I (Bq is a unit of radioactivity) into the blood (compartment 1) in $t = 0$. It is equivalent to setting initial

conditions as {1200, 0, 0, 0}. The content in each compartments can be obtained using AcuteInput. It can also be obtained using SystemDSolve with $b(t) = \{0, 0, 0, 0\}$.

```
In[9]:= AcuteInput[iodine131matrix, {1200, 0, 0, 0}, t1, x]
```

```
Out[9]= {x1[t1] → 1200 (-0.210331 (0. + 0. e-12.0864 t1) +
          1.06531 (0. + 0.938726 e-2.85897 t1) -
          0.0512079 (0. + 0.0164231 e-0.146574 t1) +
          0.291417 (0. + 0.00277467 e-0.0927512 t1)),
          x2[t1] → 1200 (-0.210331 (0. + 0. e-12.0864 t1) +
          1.06531 (0. - 0.282505 e-2.85897 t1) -
          0.0512079 (0. - 0.265339 e-0.146574 t1) +
          0.291417 (0. + 0.986105 e-0.0927512 t1)),
          x3[t1] → 1200 (-0.210331 (0. + 0. e-12.0864 t1) +
          1.06531 (0. + 0.000901627 e-2.85897 t1) -
          0.0512079 (0. + 0.964012 e-0.146574 t1) +
          0.291417 (0. + 0.166101 e-0.0927512 t1)),
          x4[t1] → 1200 (-0.210331 (0. + 1. e-12.0864 t1) +
          1.06531 (0. + 0.197442 e-2.85897 t1) -
          0.0512079 (0. + 0.00266957 e-0.146574 t1) +
          0.291417 (0. + 0.000448997 e-0.0927512 t1)) }
```

Multiexponential input. In this example it is assumed an input into compartment 1 is given by $b_1(t) = -27.13 e^{-24.08 t} + 27.13 e^{-2.86 t} - 0.020 e^{-0.147 t} + 0.0194 e^{-0.093 t}$ (This kind of input happens in real situations when there is an input from the GIT to the blood, for instance if the iodine is intaken by orally). The initial condition is {1, 0, 0, 0}. The content in each compartments is computed using SystemDSolve.

```

In[10]:= SystemDSolve[iodine131matrix,
  {1, 0, 0, 0}, {-27.13 e-24.08 t + 27.13 e-2.86 t -
    0.020 e-0.147 t + 0.0194 e-0.093 t, 0, 0, 0}, t, t, x]

Out[10]= {x1[t] → 1.27845 e-24.08 t - 26 383.4 e-2.86 t +
  26 383.1 e-2.85897 t - 0.046598 e-0.147 t + 0.0389847 e-0.146574 t -
  0.00840863 e-0.146574 t + 0.000953319 e-0.146574 t -
  0.0563495 e-0.093 t + 0.00792735 e-0.0927512 t +
  0.0635695 e-0.0927512 t - 0.000914526 e-0.0927512 t,
  x2[t] → -0.0443356 e-24.08 t + 7937. e-2.86 t - 7939.87 e-2.85897 t +
  0.746684 e-0.147 t - 0.629853 e-0.146574 t + 0.135853 e-0.146574 t -
  0.0154022 e-0.146574 t - 22.408 e-0.093 t + 2.81734 e-0.0927512 t +
  22.5923 e-0.0927512 t - 0.325018 e-0.0927512 t,
  x3[t] → 0.0000160487 e-24.08 t - 25.3217 e-2.86 t +
  25.3404 e-2.85897 t - 2.30199 e-0.147 t + 2.28834 e-0.146574 t -
  0.493573 e-0.146574 t + 0.0559583 e-0.146574 t -
  3.79278 e-0.093 t + 0.474556 e-0.0927512 t +
  3.80547 e-0.0927512 t - 0.0547464 e-0.0927512 t,
  x4[t] → -0.20688 e-24.08 t + 0.883907 e-12.0864 t -
  5549.84 e-2.86 t + 5549.16 e-2.85897 t -
  0.00757473 e-0.147 t + 0.00633692 e-0.146574 t -
  0.00136682 e-0.146574 t + 0.000154961 e-0.146574 t -
  0.00911864 e-0.093 t + 0.0012828 e-0.0927512 t +
  0.0102868 e-0.0927512 t - 0.000147988 e-0.0927512 t}

```

Periodic input. Here the input is given by $\{1 + 0.5 \cos(0.3 t), 0, 0, 0\}$ and the initial condition is $\{1, 0, 0, 0\}$. When the inputs are trigonometric functions it is a good idea to use `SystemDSolve` choosing the Laplace transform method. The evolution of the iodine retention in the blood (compartment 1) and in the thyroid (compartment 2) is plotted below.

```

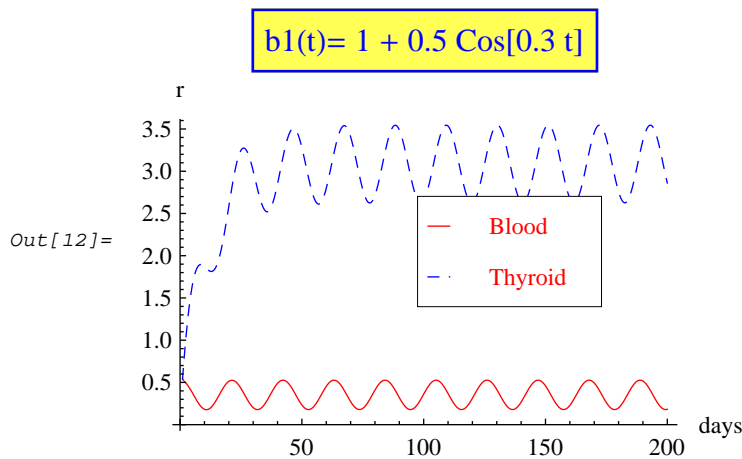
In[11]:= solIodine = SystemDSolve[iodine131matrix,
  {1, 0, 0, 0}, {1 + 0.5 Cos[0.3 t], 0, 0, 0},
  t, t, x, Method → "LaplaceTransform"];

```

```

In[12]:= Plot[Evaluate[{x1[t], x2[t]} /. solIodine],
  {t, 1, 200}, PlotStyle -> {{RGBColor[1, 0, 0]},
  {AbsoluteDashing[{4, 4}], RGBColor[0, 0, 1]}},
  AxesLabel -> {"days", "r"}, PlotLegend ->
  {"Blood", "Thyroid"}, LegendShadow -> None,
  ShadowForeground -> Directive[White, FontColor -> Red],
  LegendPosition -> {-0.25, -0.2}, LegendSize -> {0.5, 0.3},
  PlotLabel -> Style[Framed["b1(t) = 1 + 0.5 Cos[0.3 t]"],
  12, Blue, Background -> Lighter[Yellow]]]

```



Constant input. Let's consider a constant input of 12 kBq/day into compartment 1 during a time t , the initial content ($t = 0$) in each compartment is 0. After a time T , in this case 2 days, input ceases. We wish to know $x(t)$ for $0 \leq t \leq T$ and for $t > T$. The first period ($0 \leq t_1 \leq T$) is evaluated as follows:

```

In[13]:= constinput[t1_] = SystemDSolve[iodine131matrix,
  {0, 0, 0, 0}, {12, 0, 0, 0}, t, t1, x];

```

The second period starts in $t_2 = t - T$, then compartments content can be evaluated assuming a single input in $t_2 = 0$ where the initial conditions are the compartment content at the end of the first period (that is, constinput[T]).

```

In[14]:= singleinput[t2_, T_] = AcuteInput[
  iodine131matrix, Map[Last, constinput[T]], t2, x];

```

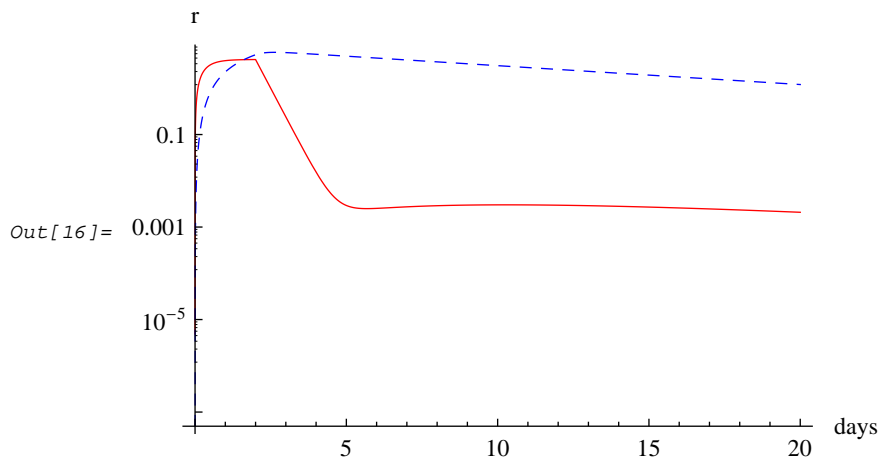
A combined function that represents the retention for both intervals (one for $0 < t \leq T$,

when the input is constant, and the other for $t > T$, when there isn't any input) can be built as follows.

```
In[15]:= constsingle[t_, T_] :=
         If[ t ≤ T, constinput[t], singleinput[t - T, T]]
```

This function is applied to T and the content in compartments 1 and 2 as function of t is plotted.

```
In[16]:= LogPlot[{constsingle[t, 2][[1, 2]], constsingle[t, 2][[2, 2]]},
                {t, 0, 20}, PlotStyle → {{RGBColor[1, 0, 0]},
                {AbsoluteDashing[{4, 4}], RGBColor[0, 0, 1]}},
                AxesLabel → {"days", "r"}]
```



The reader can apply DSolve to solve the previous examples. It will be observed that **sysModel** is clearly faster, it even gives the solution in some occasions where DSolve does not obtain it.

```
In[17]:= ClearAll[solIodine, constinput,
            singleinput, constsingle, constsingle]
```

A model with variable transfer rates

We will continue using the model of iodine 131 drawn in Fig. 0 but some transfer rates are function of the time, t .

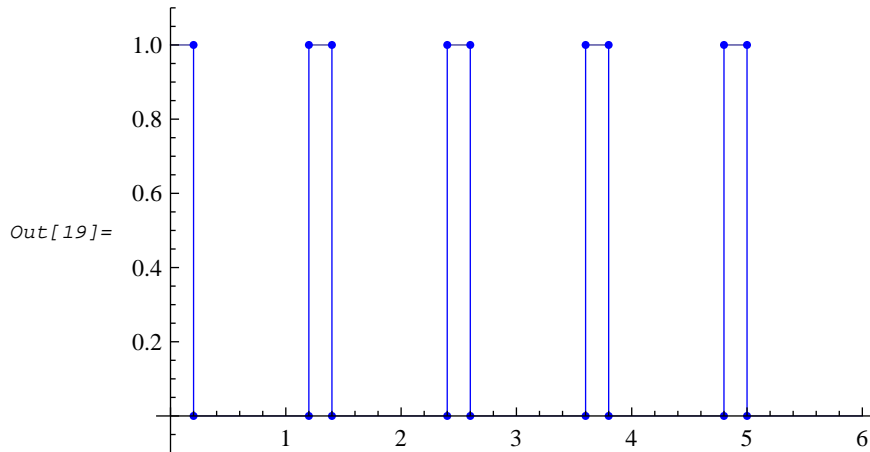
A model with multi-inputs and variable transfer rates. Let's consider the iodine model with the following transfer rate: $k_{14} = 1.9404$, $k_{12} = 0.8316 (1 + \text{Cos}(0.2 t))$, $k_{23} = 0.0086625 (1 + \text{Cos}(0.3 t))$, $k_{30} = 0.01155$ and $k_{31} = 0.0462 (1 + \text{Cos}(1.2 t))$. Also, the transfer from bladder to outside of the system (via urine excretion) happens periodically, it has a constant rate $k_{40} = 12 \text{ day}^{-1}$ during the first 0.02 hours, and $k_{40} = 0$ for the following 4 hours. An input into compartment 1 given by $b_1(t) = 1.7 t^{0.3} e^{-24 t}$, in $\text{Bq} \cdot \text{day}^{-1}$, happens every 8 hours, so during 48 hours when the input cease. We wish to obtain the iodine content in each compartment as function of the time. The initial condition is $x(t) = \{0, 0, 0, 0\}$.

The first step is to build a function $k_{40}(t)$ that represents the periodic transfer rate from bladder to outside. The process can be described by $k_{40}(t) = k h(t)$ where $h(t) = 0$ if $(n - 1)T < t \leq nT - \Delta T$, with $n = \{1, 2, \dots\}$, and $h(t) = 1$ in other case. It can be done using the function UnitStep.

Here $h(t, T, \Delta T)$ is plotted taking as example $T = 1$ and $\Delta T = 0.2$.

```
In[18]:= h[t_, a_, d_] := 1 - UnitStep[Mod[t, a + d] - d];
```

```
In[19]:= Plot[h[t, 1, 0.2`], {t, 0, 6}, PlotPoints -> 1000,
PlotRange -> {-0.1, 1.1}, ExclusionsStyle -> {Blue, Blue}]
```



Then this function is used with $k_{40} = 12 h(t, 4/24, 0.02/24) \text{ day}^{-1}$ for building the iodine 131 coefficient matrix.

```
In[20]:= iodineNonLinear =
CompartmentMatrix[4, {{1, 2, 0.8316 (1 + Cos[0.2 t])},
{1, 4, 1.9404}, {2, 3, 0.0086625 (1 + Cos[0.3 t])},
{3, 0, 0.01155}, {3, 1, 0.0462 (1 + Cos[1.2 t])},
{4, 0, h[t, 4 / 24, 0.02 / 24] }}, Log[2] / 8.02];
```

The following function represents a periodic input $b_1(t) = 1.7 t^{0.3} e^{-24t}$, in $\text{Bq} \cdot \text{day}^{-1}$, with a period T .

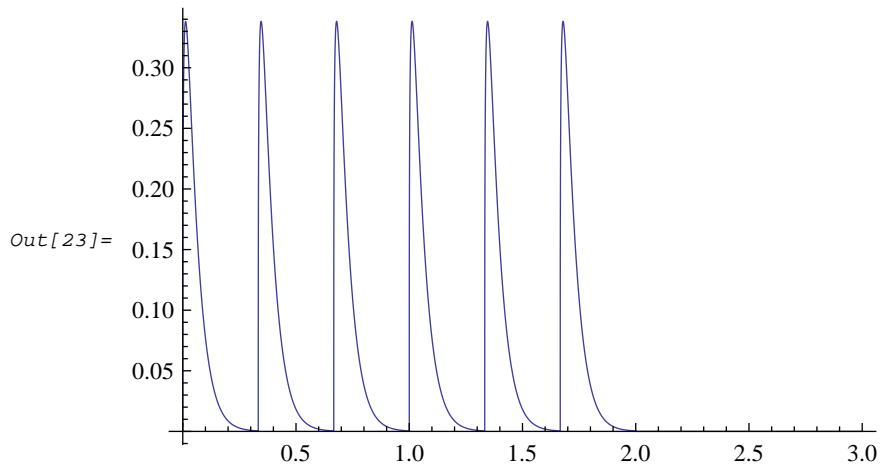
```
In[21]:= inputB[t_, T_] :=
1.7 (t - Floor[t/T] T)^0.3 Exp[-24 (t - Floor[t/T] T)];
```

Now, it builds the complete form of the inputs. It is assumed that input ceased after 2 days an $T = 1/3$ day.

```
In[22]:= b1[t_] := {If[t < 2, inputB[t, 1 / 3], 0], 0, 0, 0};
```

Here is plotted $b_1(t)$. This form of $b_1(t)$ is physiologically more representative of an injection than an impulsive input.

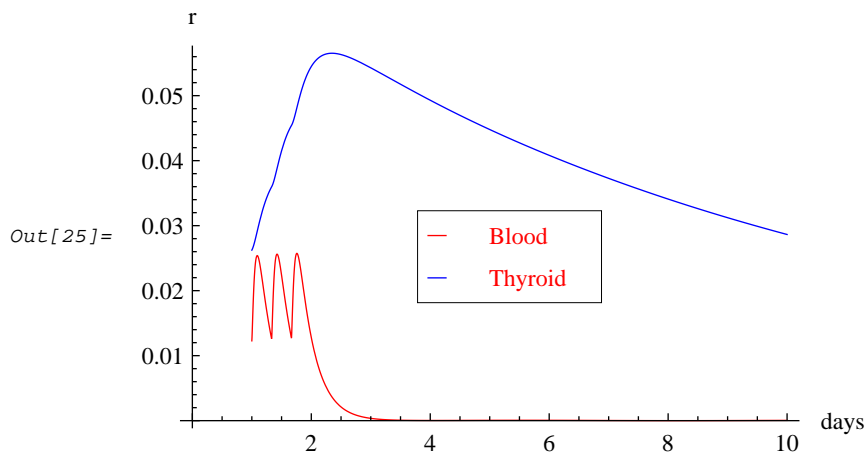
```
In[23]:= Plot[b1[t][[1]], {t, 0, 3}, PlotPoints -> 1000, PlotRange -> All]
```



Finally, the model is solved and the retention in compartment 1 (blood) and 2 (thyroid) is plotted.

```
In[24]:= {x1[t_], x2[t_], x3[t_], x4[t_]} =
  {x1[t], x2[t], x3[t], x4[t]} /.
  SystemNDSolve[iodineNonLinear, {0, 0, 0, 0},
    b1[t], {t, 0, 200}, t, x, MaxSteps -> 100 000];
```

```
In[25]:= Plot[{x1[t], x2[t]}, {t, 1, 10}, AxesOrigin -> {0, 0},
  PlotStyle -> {{RGBColor[1, 0, 0]}, {RGBColor[0, 0, 1]}},
  AxesLabel -> {"days", "r"},
  PlotLegend -> {"Blood", "Thyroid"}, LegendShadow -> None,
  LegendPosition -> {-0.25, -0.2}, LegendSize -> {0.5, 0.25},
  ShadowForeground -> Directive[White, FontColor -> Red]]
```



In bioassay with radioactive isotopes it must usually be evaluated the total disintegrations in a compartment i during a time t . In the following example it is calculated the disintegrations in thyroid during the following 200 days after starting the intake, we must multiply by 24×3600 to obtain the solution in Bq.

```
In[26]:= NIntegrate[24 * 3600 x2[t], {t, 0, 200}]
```

NIntegrate::ncvb :

NIntegrate failed to converge to prescribed accuracy after 9
 recursive bisections in t near {t} = {1.61489}.
 NIntegrate obtained 60103.80014790265`
 and 0.21766289350687973` for
 the integral and error estimates. >>

Out[26]= 60103.8

The message given by NIntegrate does not necessarily indicate an error. In fact, it can be

avoided by changing the limits of integration.

```
In[27]:= NIntegrate[24 * 3600 x2[t] , {t, 0, 1}] +
          NIntegrate[24 * 3600 x2[t] , {t, 1, 10}] +
          NIntegrate[24 * 3600 x2[t] , {t, 10, 200}]
```

```
Out[27]= 60 103.8
```

```
In[28]:= Clear[h, iodineNonLinear, inputB, b1, x1, x2, x3, x4]
```

3. The Laplace transforms and the identifiability analysis

The transfer rates k_{ij} are usually estimated using experimental data as it will shown in the next section. The problem that often happens is that there is not only one value for k_{ij} but also a number finite of values that satisfy the solution of the model. This is known as the identifiability analysis [2]. The Laplace transforms is very useful in identifiability analysis

The package function `SystemLTSolve` gives the Laplace transform $X(s)$ of $x(t)$ applying the eqn (9).

```
SystemLTSolve[Matrix-Coeffs, gives the Laplace transforms X (s)
incond, input, t, s, x]      = {X1 (s), ..., Xn (s)},
                             of a SODE  $\dot{x}(t) = A \cdot x(t) + b(t)$ ,
                             with the initial conditions,
                             incond,  $x(0) = \{c_1, \dots, c_n\}$ ,
                             coefficient matrix, Matrix-Coeffs,
                              $A = \{a_{11}, \dots, a_{1n}\}, \dots, \{a_{n1}, \dots, a_{nn}\}$  and  $b(t)$ 
                              $= \{b_1(t), \dots, b_n(t)\}$ .
```

A model with identifiability problems. In the model of the Fig. 04 (Example 6.5. Godfrey 1983 [2]) the response in the central compartment 1 to a single input of 1 unit at $t = 0$ has been fitted (no noise is assumed) for the function $x_{1 \text{ exp}}(t) = 0.7 \exp(-5 t) + 0.2 \exp(-t) + 0.1 \exp(-0.1 t)$. The problem consists of determining the unknown transfer rates, $\{k_{10}, k_{12}, k_{13}, k_{21}, k_{31}\}$, that will be assumed constants.

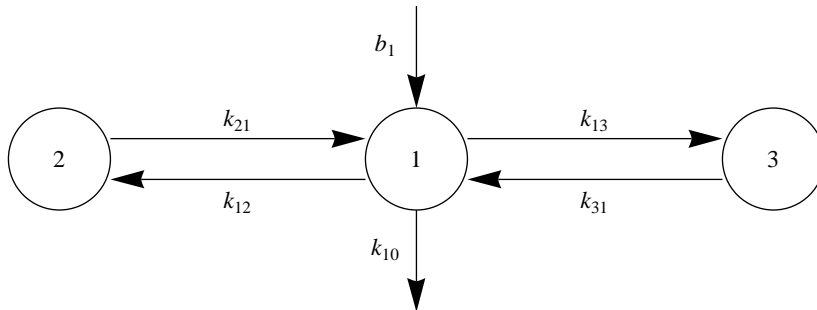


Fig4

Fig. 04 Tricompartimental model with input in the central compartment

The problem could be solved obtaining $\{k_{10}, k_{12}, k_{13}, k_{21}, k_{31}\}$ from $x_{1 \text{ exp}}(t) = x_1(t)$, however it can be considerably easier applying its Laplace transforms $X_{1 \text{ exp}}(s) = X_1(s)$.

The Laplace transform $X(s)$ of an impulsive input of 1 (It can be used any mass unit, so if 1 is in mg the solution will be in mg) in compartment 1 at $t=0$ can be made using SystemLTSolve with the initial condition $\{1, 0, 0\}$ and $b(t) = \{0, 0, 0\}$. We will need to define the compartmental matrix of the system.

```
In[29]:= tricompartment = CompartmentMatrix[3, {{1, 2, k12},
      {2, 1, k21}, {1, 3, k13}, {3, 1, k31}, {1, 0, k10}}];
```

```
In[30]:= {X1[s_], X2[s_], X3[s_]} =
      {X1[s], X2[s], X3[s]} /. SystemLTSolve[tricompartment,
      {1, 0, 0}, {0, 0, 0}, t, s, X] // Simplify
```

```
Out[30]= {
      (s + k21) (s + k31)
      -----,
      k10 (s + k21) (s + k31) + s (k12 (s + k31) + (s + k21) (s + k13 + k31))
      -----,
      k12 (s + k31)
      -----,
      k10 (s + k21) (s + k31) + s (k12 (s + k31) + (s + k21) (s + k13 + k31))
      -----,
      k13 (s + k21)
      -----,
      k10 (s + k21) (s + k31) + s (k12 (s + k31) + (s + k21) (s + k13 + k31))
      -----}

```

The Laplace transform $X_{1 \text{ exp}}(s)$ of $x_{1 \text{ exp}}(t) = 0.7 \exp(-5 t) + 0.2 \exp(-t) + 0.1 \exp(-0.1 t)$ is

```
In[31]:= Xlexp = LaplaceTransform[
          0.7 Exp[-5 t] + 0.2 Exp[-t] + 0.1 Exp[-0.1 t], t, s]
```

$$\text{Out}[31]= \frac{0.1}{0.1 + s} + \frac{0.2}{1 + s} + \frac{0.7}{5 + s}$$

Now, the transfer rate $\{k_{10}, k_{12}, k_{13}, k_{21}, k_{31}\}$ can be obtained solving $X_{1 \text{ exp}}(s) = X_1(s)$. The following procedure is applied.

```
In[32]:= SolveAlways[Xlexp == X1[s], s]
```

```
Out[32]= {{k10 -> 0.746269, k12 -> 1.25488, k13 -> 1.70885, k21 -> 0.324354,
           k31 -> 2.06565}, {k10 -> 0.746269, k12 -> 1.70885,
           k13 -> 1.25488, k21 -> 2.06565, k31 -> 0.324354}}
```

You realize that two sets of solutions are possible. Notice that neither set contains any negative rate constants, which would permit rejection of such a set.

```
In[33]:= ClearAll[tricompartament, X1, X2, X3, Xlexp];
```

4. Model Fitting

■ Basic fit

In this section we will describe how experimental data and biokinetic parameters can fitted. More detail about fitting in [8].

We want to estimate the values, k_{on} and k_{off} for the physiological model given by eqn (0) where $V_{\text{Tu}} = 6.411$, $V_p = 0.973$, $V_{\text{Tb}} = 1$, $PS = 2.714$ and $Q = 3$. We build the coefficients matrix (note that we use `CoefMatrix` and not `CompartMatrix`) as function k_{off} and k_{on} replacing V_{Tu} , V_p , V_{Tb} , PS , Q for their values.

```
In[34]:= physiomodel[kon_, koff_] =
  CoefMatrix[3, {{1, 1, - (Q/Vp + PS/Vp)}, {1, 2, PS/Vp},
    {2, 1, PS/Vtu}, {2, 2, - (PS/Vtu + kon)}, {2, 3, koff Vtb/Vtu},
    {3, 2, kon Vtu/Vtb}, {3, 3, -koff}}] /.
  {Vtu -> 6.411, Vp -> 0.973, PS -> 2.714, Q -> 3,
   Vtb -> 1, kon -> kon, koff -> koff};
```

```
In[35]:= ShowODE[physiomodel[kon, koff], {0, 0, 0},
  {Q/Vp b1[t] /. {Vp -> 0.973, Q -> 3}, 0, 0}, t, x] //
  TableForm // TraditionalForm
```

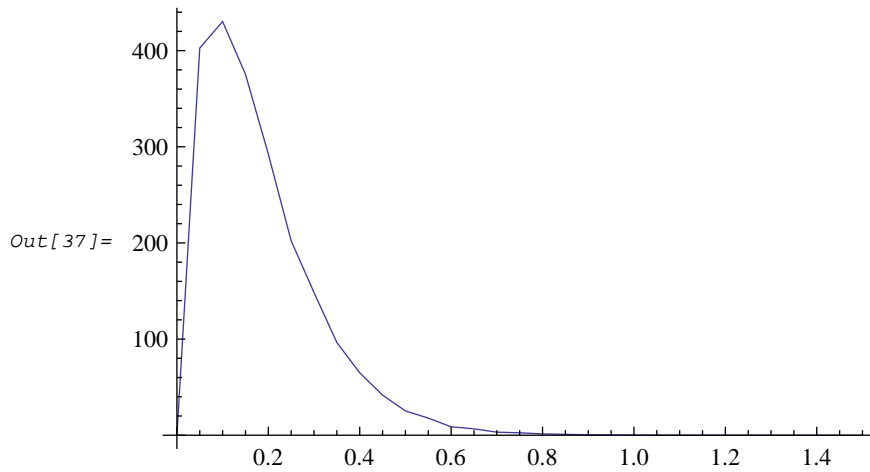
```
Out[35]//TraditionalForm=
```

$$\begin{aligned} (x_1)'(t) &= 3.08325 b_1(t) - 5.87256 x_1(t) + 2.78931 x_2(t) \\ (x_2)'(t) &= 0.155982 k_{\text{off}} x_3(t) + (-k_{\text{on}} - 0.423335) x_2(t) + 0.423335 x_1(t) \\ (x_3)'(t) &= 6.411 k_{\text{on}} x_2(t) - k_{\text{off}} x_3(t) \\ x_1(0) &= 0 \\ x_2(0) &= 0 \\ x_3(0) &= 0 \end{aligned}$$

The mathematical expression of $b_1(t)$ was unknown but an experiment was made [4] where $b_1(t)$ was given by the best fit of the input function to experimental data $\{\{t_1, b_1\}, \dots, \{t_n, b_n\}\}$ obtained via sampling from an arterial catheter. Here are the experimental data:

```
In[36]:= dataCatheter = {{0., 0.}, {0.05, 402.7}, {0.1, 430.3},
  {0.15, 375.4}, {0.2, 292.4}, {0.25, 202.2},
  {0.3, 148.4}, {0.35, 96.4}, {0.4, 64.9}, {0.45, 41.7},
  {0.5, 25.3}, {0.55, 17.8}, {0.6, 8.8}, {0.65, 6.6},
  {0.7, 3.2}, {0.75, 2.5}, {0.8, 1.4}, {0.85, 0.9},
  {0.9, 0.5}, {1., 0.2}, {1.1, 0.07}, {1.2, 0.03},
  {1.3, 0.01}, {1.4, 0.003}, {1.45, 0.001}, {1.5, 0.001}};
```

```
In[37]:= ListPlot[dataCateter, Joined → True, PlotRange → All]
```



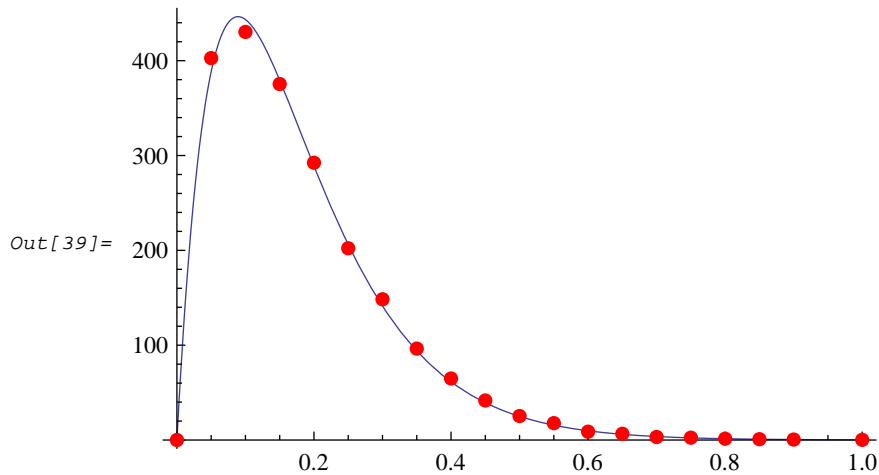
The graphic form suggests that the below function could be appropriated to fit the data

```
FindFit
In[38]:= b[t_] =
          a t Exp[c t] /. FindFit[dataCateter, a t Exp[c t], {a, c}, t]
```

```
FindFit
Out[38]= 13610.1 e-11.216 t t
```

Here are shown experimental data and the fitted function

```
In[39]:= Plot[b[t], {t, 0, 1},
  Epilog -> {Hue[0], PointSize[0.02`], Point /@ dataCateter}]
```



Now, we have the input function in the form required by SysModel.

```
In[40]:= inputb[t_] = { $\frac{Q}{v_p} b[t]$ , 0, 0} /. {vp -> 0.973, Q -> 3}
```

```
Out[40]= {41963.3 e-11.216 t t, 0, 0}
```

It was also measured $x_1(t)$ by sampling and it was obtained the following experimental data $\{\{t_1, x_1(t_1)\}, \dots, \{t_n, x_1(t_n)\}\}$.

```
In[41]:= dataPhysio = {{0.03, 14.50}, {0.08, 61.06},
  {0.28, 120.93}, {0.33, 109.01}, {0.38, 98.08},
  {0.48, 69.66}, {0.55, 51.51}, {0.65, 34.77},
  {0.75, 23.21}, {0.85, 15.59}, {0.95, 12.16},
  {1.05, 9.598}, {1.15, 8.278}, {1.25, 6.842},
  {1.35, 5.871}, {1.45, 5.297}, {1.6, 4.886},
  {1.8, 3.846}, {2., 3.317}, {2.2, 2.899}, {2.4, 2.627},
  {2.6, 2.289}, {2.8, 1.998}, {3., 1.930}, {3.4, 1.589},
  {3.75, 1.308}, {4.25, 1.112}, {4.75, 1.064},
  {5.25, 0.938}, {6.75, 0.842}, {7.25, 0.831},
  {7.75, 0.778}, {8.25, 0.818}, {11., 0.739}};
```

We wish to obtain the numeric values of k_{on} and k_{off} fitting $x_1(t)$ experimental data. We

applied to the model the Laplace transforms obtaining $X_1(s, k_{on}, k_{off})$

```
In[42]:= X1[kon_, koff_] = X1[s] /. SystemLTSolve[
  physiomodel[kon, koff], {0, 0, 0}, inputb[t], t, s, X];
```

$x_1(t)$ can be obtained applying the InverseLaplaceTransform as function of k_{on} and k_{off} . Notice that $x_1(t)$ only has analytical solution when k_{on} and k_{off} take a numeric values.

```
In[43]:= x1fit[kon_?NumericQ, koff_?NumericQ] := Block[{x1},
  x1[t_] = InverseLaplaceTransform[X1[kon, koff], s, t];
  Plus @@ Apply[(x1[#1] - #2) ^ 2 &, dataPhysio, {1}]];
```

We can now use FindMinimum to find k_{on} and k_{off} .

```
In[44]:= FindMinimum[x1fit[kon, koff], {kon, 0.1, 1}, {koff, 0.1, 1}]
```

```
Out[44]= {72.3264, {kon -> 0.713783, koff -> 0.11072}}
```

They can also be found using NMinimize[{x1fit[kon, koff], 0 < kon, 0 < koff}, {kon, koff}], it has the advantage that restrictions for parameters can be used. However, it will usually take more time of computation.

The same solution can obtained applying SystemNDSolve instead of Laplace Method to solve the differentialequation

```
x1fit[kon_?NumericQ, koff_?NumericQ] :=
  Block[{x1}, x1[t1_] = x1[t1] /. SystemNDSolve[
    physiomodel[kon, koff], {0, 0, 0},
    inputb[t], {t, 0, 12}, t1, x]; Plus @@
    Apply[(x1[#1] - #2) ^ 2 &, dataPhysio, {1}]];
```

■ Statistical analysis

Sometimes we need not only the values of fitted parameters but also additional statistic information such as confidence intervals, ANOVA table and so on. We can apply `NonlinearModelFit` but if the model to be fitted it doesn't have an analytic expression `NonlinearRegress` can not find the derivatives. In this case, the gradient method can be used but the derivatives for each parameter to be fitted must be supplied to *Mathematica* (It was suggested in the mathgroup by Carl Woll, from Washington University). It can be complex when the number of variables and coefficients is big. We have found that the problem can be made easier using the numeric derivatives applying the package

```
In[45]:= Needs["NumericalCalculus`"];
```

The first step consists in defining a *Mathematica* function that gives the solution of the model as function of the parameters to be fitted. We need the solution of the model for $x_1(t)$ because experimental data has been measured for $x_1(t)$. Notice that the model has only solution if the parameters take a numeric values.

```
In[46]:= model[t1_?NumericQ, kon_?NumericQ, koff_?NumericQ] :=
  x1[t1] /. SystemNDSolve[physiomodel[kon, koff],
    {0, 0, 0}, inputb[t], {t, 0, 12}, t1, x]
```

It could also be used: `model[t1_?NumericQ, kon_?NumericQ, koff_?NumericQ] := InverseLaplaceTransform[X1[kon, koff], s, t1]`;

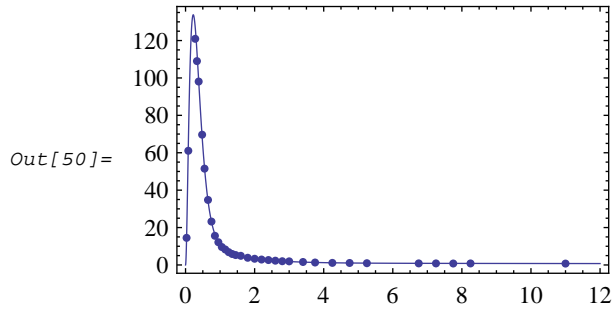
The derivatives of the model for k_{on} and k_{off} are supplied to *Mathematica*.

```
In[47]:= Derivative[0, 1, 0][model][a1_, n_, a3_] :=
  ND[model[a1, a2, a3], a2, n];
Derivative[0, 0, 1][model][a1_, a2_, n_] :=
  ND[model[a1, a2, a3], a3, n];
```

Finally we can apply `NonlinearModelFit` including the statistical reports (the list of available reports can be check `nlm["Properties"]`). It will usually take a long time.

```
In[49]:= nlm = NonlinearModelFit[dataPhysio, model[t, kon, koff],
  {{kon, 0.71}, {koff, 0.11}}, {t}, Method -> Gradient];
```

```
In[50]:= Show[ListPlot[dataPhysio],
  Plot[nlm[t], {t, 0, 12}, PlotRange -> {0, 500}],
  Frame -> True]
```



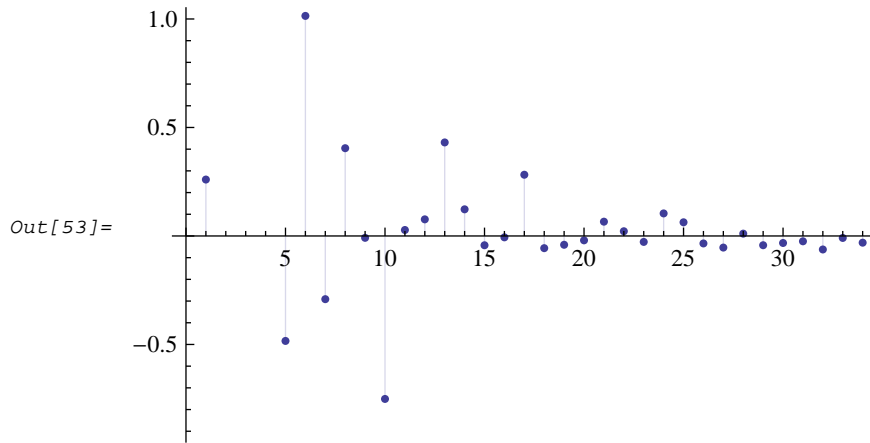
```
In[51]:= nlm["ParameterTable"]
```

	Estimate	Standard Error	t Statistic	P-Value
Out[51]= kon	0.713776	0.13596	5.24988	9.60469×10^{-6}
koff	0.110714	0.0871494	1.2704	0.213102

```
In[52]:= nlm["FitResiduals"]
```

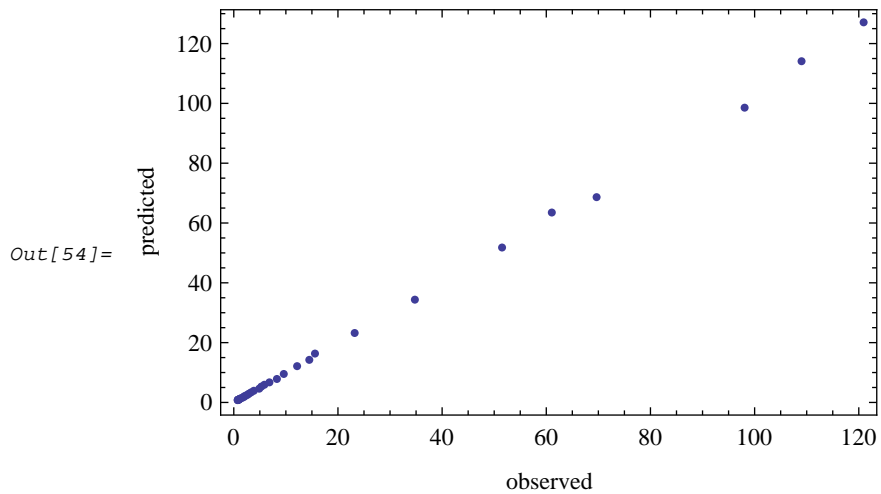
```
Out[52]= {0.259995, -2.44331, -6.16581, -5.08623,
  -0.483776, 1.01419, -0.291276, 0.40449, -0.00856568,
  -0.751105, 0.0275022, 0.07658, 0.431045, 0.122794,
  -0.0428312, -0.00633453, 0.282182, -0.0562251,
  -0.0405949, -0.0201386, 0.0658499, 0.0217456,
  -0.0274031, 0.103885, 0.0630154, -0.0344641,
  -0.0535101, 0.0107025, -0.0426817, -0.0323704,
  -0.0245828, -0.0622732, -0.00913405, -0.0309325}
```

```
In[53]:= ListPlot[%, Filling -> Axis]
```



Plot the predicted values against the observed values:

```
In[54]:= ListPlot[
  Transpose[nlm[{"Response", "PredictedResponse"}]],
  FrameLabel -> {"observed", "predicted"},
  Frame -> True, Axes -> False]
```



```
In[55]:= nlm[{"MeanPredictionConfidenceIntervalTable",
  "SinglePredictionConfidenceIntervalTable"}]
```

Observed	Predicted	Standard Error	Confidence Interval
14.5	14.24	1.13628×10^{-6}	{14.24, 14.24}
61.06	63.5033	0.000113428	{63.5031, 63.5035}
120.93	127.096	0.0190186	{127.057, 127.135}
109.01	114.096	0.0330181	{114.029, 114.163}
98.08	98.5638	0.0512763	{98.4593, 98.6682}
69.66	68.6458	0.0986937	{68.4448, 68.8468}
51.51	51.8013	0.137902	{51.5204, 52.0822}
34.77	34.3655	0.197338	{33.9635, 34.7675}
23.21	23.2186	0.25556	{22.698, 23.7391}
15.59	16.3411	0.308315	{15.7131, 16.9691}
12.16	12.1325	0.353109	{11.4132, 12.8518}
9.598	9.52142	0.388886	{8.72929, 10.3136}
8.278	7.84696	0.415604	{7.0004, 8.69351}
6.842	6.71921	0.433881	{5.83542, 7.60299}
5.871	5.91383	0.444673	{5.00806, 6.8196}
5.297	5.30333	0.449124	{4.3885, 6.21817}
4.886	4.60382	0.44646	{3.69441, 5.51323}
3.846	3.90223	0.431272	{3.02375, 4.7807}
3.317	3.35759	0.40981	{2.52284, 4.19235}
2.899	2.91914	0.388169	{2.12846, 3.70981}
Out[55]= { 2.627	2.56115	0.370687	{1.80609, 3.31621}
2.289	2.26725	0.359885	{1.53419, 3.00032}
1.998	2.0254	0.356518	{1.2992, 2.75161}
1.93	1.82612	0.359902	{1.09302, 2.55921}
1.589	1.52598	0.380497	{0.750937, 2.30103}
1.308	1.34246	0.405028	{0.51745, 2.16748}
1.112	1.16551	0.438516	{0.272282, 2.05871}

1.116	1.10551	0.430510	{0.616606, 2.05017}
1.064	1.0533	0.463904	{0.108356, 1.99824}
0.938	0.980682	0.48001	{0.00293393,
0.842	0.87437	0.487525	{-0.118686,
0.831	0.855583	0.481919	{-0.126055,
0.778	0.840273	0.474247	{-0.125735,
0.818	0.827134	0.465238	{-0.120525,
0.739	0.769932	0.407921	{-0.0609753,

You can realize that the previous patterns could be applied to other model with two or more parameters for fitting .

The pattern for fitting model is the following:

```

Derivative[1, 0, 0, 0, ..., 0][model][n_, a2_,
a3_, a4_, ...] := ND[model[a1, a2, a3, a4, ...,
ai], a1, n];
Derivative[0, 1, 0, 0, ..., 0][model][a1_, n_,
a3_, a4_, ...] := ND[model[a1, a2, a3, a4, ...,
ai], a2, n];
Derivative[0, 0, 1, 0, ..., 0][model][a1_, a2_,
n_, a4_, ...] := ND[model[a1, a2, a3, a4, ...,
ai], a3, n]; ....Derivative[0, 0, 0, ..., 0,
1][model][a1_, a2_, a3_, ..., n_] :=
ND[model[a1, a2, a3, a4, ..., ai], ai, n];
nlm= NonlinearModelFit[data, model, variables,
parameters, Method -> Gradient];

```

and then add

```
In[56]:= ClearAll[physiomodel,  
           dataCateter, b, inputb, X1, x1fit, model];
```

5. Additional information

Some of its features can be run directly in the *webMathematica* side: <http://www3.enusa.es/webMathematica/Public/biokmod.html>.

The author (guillermo@usal.es) will appreciate any suggestion.

6. References

[1]

1. J.A. Jacquez *Compartmental Analysis in Biology and Medicine*, The University of Michigan Press, 1985.

[2]

2. K. Godfrey, *Compartmental models and their application*. Academic Press, London, 1983.

[3]

3. G. Sanchez and J. Lopez-Fidalgo, 'Mathematical techniques for solving analytically large compartmental systems'. *Health Phys.*, 85 (2): 194-193 (2003).

[4]

4. A. Sánchez-Navarro, C. Casquero, and M. Weiss, 'Distribution of Ciprofloxacin and Ofloxacin in the Isolated Hindlimb of the Rat', *Pharmaceutical Research*, 16: 587-591 (1999).

[5]

5. Q. Zheng, 'Exploring Physiologically based pharmacokinetic model'. *Mathematica in Education and Research*, 6 (2): 22-28 (1997)

[6]

6. W. Harris and H. von Bremen, 'Using the Laplace Transform to compute the matrix exponential'. *Mathematica in Education and Research*, 7 (2): 35-36 (1998).

ICRP 78

7. International Commission on Radiological Protection. *Individual Monitoring for Internal Exposure of Workers*. ICRP Publication 78. Pergamon Press. Oxford, 1997.

[8]

8. D.C. Bates and D.G. Watts. *Nonlinear regression Analysis and its Applications*. John Wiley and Sons, New York, 1988.