LIXOFT

MLXTRAN

THE MODEL CODING LANGUAGE FOR

- MONOLIX[1]
- MLXPLORE
- SIMULIX

March 2014 .

[1] MLXTRAN is the model coding language used by MONOLIX, MLXPLORE and SIMULIX. This document presents MLXTRAN features for MONOLIX.

Contents

1 General structure of MLXTRAN **4**

2 The block INPUT: **4**

3 The block EQUATION: **5**

3.1 Intermediate variable . . . 5

3.2 Ordinary Differential Equations . . . 5

3.3 Delayed Differential Equations . . . 6

3.4 Mixtures . . . 7

3.5 Usual mathematical functions . . . 8

3.6 Probability distribution functions . . . 8

3.7 Operators . . . 9

3.8 Conditional statements . . . 11

3.9 Examples . . . 11

4 The block OBSERVATION: **17**

5 Pharmacokinetic model/Dynamical system **21**

5.1 The pkmodel function . . . 21

5.2 The PK macros . . . 24

5.3 Independent variables from doses . . . 29

5.4 Analytical solutions . . . 29

5.4.1 General description . . . 29

5.4.2 Miscellaneous . . . 32

5.5 Examples . . . 33

6 The block OUTPUT: **37**

6.1 Examples . . . 37

7 Miscallenous **38**

7.1 Macros 38
7.2 Main independent variable 39
7.3 Annotations 39
7.4 Reserved names 40

1 General structure of MLXTRAN

MLXTRAN is a declarative, human-readable language describing simple and complex structural models as well as hierarchical statistical models. Different components of the model are implemented in different sections.

A model script for MONOLIX focuses on the structural model and the computation tasks built on top of it. The structural model is a parametric model. It defines the computation of the predictions. Predictions form a deterministic function of usual and random variables. They are defined at the individual level.

Some keywords manage the complexity of the deterministic computations. An ODE initial value problem can be defined. The function `pkmodel` and the block **`PK:`** can be included to define pharmacokinetics with domain-specific keywords.

A MLXTRAN script for MONOLIX is composed of several blocks:

`INPUT:`	The MLXTRAN script can access variables from other scripts, data, or graphical user interface.
`PK:`	For PK models, the structural model can include a block `PK:` (optional).
`EQUATION:`	MLXTRAN supports flexible equation-based descriptions implemented in a block `EQUATION:`. For example, it can include ODEs or DDEs.
`OBSERVATION:`	Observations of the structural model can be defined, adding a statistical layer to its predictions.
`OUTPUT:`	The model from an MLXTRAN script can be exercised to provide computations or simulation results.

2 The block INPUT:

The block **`INPUT:`** declares some variables that were defined outside of the current script, and enables them. They come from the project or the data. This block also declares the types of these variables. The keyword `parameter` declares the input individual parameters. `regressor` declares the input regression values.

Example:

```
INPUT:
parameter = {ka, V, Cl}
regressor = {Weight, Age}
```

3 The block EQUATION:

The block **EQUATION:** contains mathematical equations. Several keywords allow to describe complex or domain-specific equations. Remember that the model description is a declarative language, not an imperative language. Therefore these equations are mathematical definitions rather than a series of instructions.

3.1 Intermediate variable

An intermediate variable can be defined as the result of a computational expression, based on other variables and usual mathematical operators and functions.

Example:

```
INPUT:
parameter = {a, b, c}

EQUATION:
d = -a^2 * (exp(b)+3)/c
```

Intermediate variables can also be defined in the block **PK:**.

3.2 Ordinary Differential Equations

A system of ODE (Ordinary Differential Equations) can be defined in the block `EQUATION:`.

The main independent variable `t` is the differentiation variable. The components of its solution can be referenced as any other variable. The problem is defined by the equations of the derivatives, the initial time point and the initial values of the components. The stiffness of the problem can be defined. The problem accepts input source terms, which allows to bring some discontinuity into the defined derivatives.

Derivatives The keywords prefixed with `ddt_` in front of a variable name, as `ddt_a` and `ddt_b`, define the derivatives of an ODE system. The variable names denote the components of the solution. These variables are defined at the whole section level through their derivatives. The derivatives themselves aren't variables but keywords, and cannot be referenced by other equations nor be defined under a conditional statement.

The structure of the ODE system is part of the structure of the whole model and cannot be conditional. Adding conditional intermediate variables for the values of the derivatives easily provides this flexibility.

The derivatives of the ODE system cannot be referenced by other equations, but they can be *extended* with some domain-specific keywords into a custom dynamic system. Such keywords are available for the pharmacokinetics, for example: in this case the ODE system includes a

derivative for the quantity whose dynamic is also ruled by a specific keyword, as a dose absorption. This results in a complete system where the final derivative of the ODE component is the derivative provided as an equation within the initial ODE system augmented by the input derivative translating the dynamics from the domain-specific keyword. The example on the custom elimination in section 5.5 illustrates this feature.

Initial values The keyword `t0` defines the initial point of an ODE problem. More precisely, the components are defined for $t \leq t0$ as the equations provided for the initial values, and for $t > t0$ they are defined as the solution of the ODE system with initial values fixed at $t = t0$. By default, `t0` is the first time point defined in the MlxPlore project.

The keyword `t0` defines the initial time point of an ODE system, while the variable names with suffix `_0` define the initial values of the system, i.e. the values of its components for $t \leq t0$. Then, for $t > t0$, the components of the system are defined as the solution of the ODE system. By default, an initial value is null.

Example:

```
t0 = 5
A_0 = r/k
B_0 = 0
ddt_A = r - c*A*B - k*A
ddt_B = c*A*B - d*B
```

Type of ODE system The keyword `odeType` defines the type of the ODE system. The problem can be indicated as being stiff, non-stiff, or linear. By default, the problem is viewed as non-stiff.

Example:

```
odeType = stiff
; Other possible values:
; odeType = nonStiff
; odeType = linear
```

Source terms An ODE problem accepts input source terms. They can be defined into the project, within a data set, or using the keyword `depot`. Each input source term defines an instantaneous shift of the value of its target ODE component by a given amount. A zero order, first order, or transit process can also be defined with `depot`.

3.3 Delayed Differential Equations

A system of DDE (Delayed Differential Equations) can be defined in the bloc `EQUATION:`.

A DDE system is defined as an ODE system, except that some derivatives rely on the value of a component at a previous time. These delayed components are defined using a specific keyword.

Also, the initialization of the system relies on an history function to define the trajectory in the past, instead of a vector of initial values.

Delayed components The keyword `delay` defines a delayed component. It takes two arguments: the name of the component and the value of the delay. The delay must be constant on occasions. It is available only directly within the equation of a derivative.

Example:

```
t0 = 5
A_0 = r/k
B_0 = 0
ddt_A = r - c*A*B - k*A
ddt_B = c*A*B - delay(A, d)
```

History function The components of the history function use the same syntax as the initial values of an ODE system.

Example:

```
t0 = 5
A_0 = r/k
B_0 = 0
ddt_A = r - c*A*B - k*A
ddt_B = c*A*B - d*B
```

Type of DDE system The same method is used for all DDE systems. The keyword `odeType` is only relevant for ODE systems.

Limitations An observation for repeated time to event cannot be combined with a DDE system.

3.4 Mixtures

A mixture of continuous observations can be defined within a block **EQUATION:**. It can be a mixture within or between subjects. Both are functions taking as arguments the weighted predictions to mix.

Example:

```
EQUATION:
f = bsmm(f1, p, f2, 1-p)
;f = wsmm(f1, p, f2, 1-p)
```

Mixtures can also be defined in the block **PK:**.

3.5 Usual mathematical functions

The following usual mathematical functions are available. They perform scalar operations, either on integers or real numbers. These operations are performed using a double precision for their floating-point implementation.

Function	Syntax	Remarks
Smallest value	`min(a, b)`	
Largest value	`max(a, b)`	
Absolute value	`abs(a)`	
Square root	`sqrt(a)`	
Exponential	`exp(a)`	
Natural logarithm	`log(a)`	
Common logarithm	`log10(a)`	
Logistic function	`logit(a)`	
Inverse of the logistic function	`invLogit(a)`	
Probit function	`probit(a)`	Aliases: `norminv`, `qnorm`.
Normal CDF	`normcdf(a)`	Alias: `pnorm`.
Sine	`sin(a)`	
Cosine	`cos(a)`	
Tangent	`tan(a)`	
Inverse sine	`asin(a)`	
Inverse cosine	`acos(a)`	
Inverse tangent	`atan(a)`	
Hyperbolic sine	`sinh(a)`	
Hyperbolic cosine	`cosh(a)`	
Hyperbolic tangent	`tanh(a)`	
Four quadrant inverse tangent	`atan2(a, b)`	
Logarithm of gamma function	`gammaln(a)`	Alias: `lgamma`.
Downward rounding	`floor(a)`	
Upward rounding	`ceil(a)`	
Factorial	`factorial(a)`	
Logarithm of factorial	`factln(a)`	Alias: `lfactorial`.
Remainder after division	`rem(a)`	

3.6 Probability distribution functions

The probability density function (pdf) and cumulative distribution function (cdf) are available for continuous distributions.

Distribution	pdf	cdf	parameters	support	equation (pdf)
Beta	betapdf	betacdf	α, β	$[0;1]$	$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$
Chi-squared	chi2pdf	chi2cdf	k	$[0;+\infty[$	$\frac{x^{\frac{k}{2}-1}e^{-\frac{x}{2}}}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)}$
Exponential	exppdf	expcdf	$\lambda[=1]$	$[0;+\infty[$	$\lambda e^{-\lambda x}$
Extreme value (Gumbel)	evpdf	evcdf	$\mu[=0], \beta[=1]$	$\mathbb{R}$	$\frac{e^{-e^{-\frac{x-\mu}{\beta}}}e^{-\frac{x-\mu}{\beta}}}{\beta}$
Fisher-Snedecor	fpdf	fcdf	d_1, d_2	$[0;+\infty[$	$\frac{\sqrt{\frac{(d_1x)^{d_1}d_2^{d_2}}{(d_1x+d_2)^{d_1+d_2}}}}{xB\left(\frac{d_1}{2},\frac{d_2}{2}\right)}$
Gamma	gampdf	gamcdf	$k, \theta[=1]$	$[0;+\infty[$	$x^{k-1}\frac{e^{-\frac{x}{\theta}}}{\Gamma(k)\theta^k}$
Gompertz	gpzpdf	gpzcdf	η, b	$[0;+\infty[$	$b\eta e^{bx}e^{-\eta\left(e^{bx}-1\right)}$
Log-normal	lognpdf	logncdf	$\mu[=0], \sigma[=1]$	$[0;+\infty[$	$\frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$
Normal	normpdf	normcdf	$\mu[=0], \sigma[=1]$	$\mathbb{R}$	$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
Rayleigh	raylpdf	raylcdf	$\sigma[=1]$	$[0;+\infty[$	$\frac{xe^{-\frac{x^2}{2\sigma^2}}}{\sigma^2}$
Student's t	tpdf	tcdf	k	$\mathbb{R}$	$\frac{\Gamma\left(\frac{k+1}{2}\right)}{\sqrt{k\pi}\,\Gamma\left(\frac{k}{2}\right)}\left(1+\frac{t^2}{k}\right)^{-\frac{k+1}{2}}$
Uniform	unifpdf	unifcdf	$a[=0], b[=1]$	$[a;b]$	$\frac{1}{b-a}$
Weibull	wblpdf	wblcdf	$k, \lambda[=1]$	$[0;+\infty[$	$\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}e^{-\left(\frac{x}{\lambda}\right)^k}$

3.7 Operators

The following operators are available, under their mathematical acceptation.

Arithmetic operators The arithmetic operators perform operations on real numbers. These operations are performed using a double precision for their floating-point implementation. The specific operations on integers have a function call syntax, e.g. factorial.

Operator	Syntax	Remarks
Equal	`a = b`	Defines variables.
Addition	`a + b`	
Subtraction	`a - b`	
Multiplication	`a * b`	
Division	`a / b`	
Power	`a ^ b`	Right-associative. Takes precedence on unary plus and minus.
Unary plus	`+a`	
Unary minus	`-a`	

Logical operators The logical operators perform operations on boolean values. They appear as conditions within conditional statements.

Operator	Syntax	Remarks
Negation	`~a`	Another syntax is `!a`.
Or	`a | b`	Another syntax is `a || b`.
And	`a & b`	Another syntax is `a && b` .

Relational operators The relational operators perform operations on real numbers. These operations are performed using a double precision for their floating-point implementation.

Operator	Syntax	Remarks
Equal to	`a == b`	
Not equal to	`a ~= b`	Another syntax is `a != b`.
Greater than	`a > b`	
Less than	`a < b`	
Greater than or equal to	`a >= b`	
Less than or equal to	`a <= b`	

3.8 Conditional statements

A conditional statement can be built by combining the keywords `if`, `elseif`, `else` and `end`. Several `elseif` keywords can be chained, and the conditions are exclusive in sequence. A default value can be provided using the keyword `else`, but also as a simple definition preceding the conditional structure. An unspecified conditional value is null. The enclosed variables are not local variables, which means that a variable with remaining conditional definitions is still incomplete and cannot be referenced into another definition, even under the same condition.

Only intermediate variables can be defined within conditional statements, the structure of the model cannot depend on such conditions. The derivatives of an ODE, or the PK elements of a prediction sub-model, cannot be defined within conditionals, for example. Adding conditional intermediate variables for the values of the derivatives, or the parameters of the PK element, easily provides this flexibility.

Example:

```
T0=5
if t<T0
   f=A
else
   f= A*exp(-k*(t-T0))
end
```

3.9 Examples

Conditional derivatives for an HCV Neumann model The prediction of interest is the logarithm of the viral load for a Hepatitis C viral (HCV) model, with a start and stop of treatment.

$$
\begin{cases}
TC\,|_{t=0} = \delta c/\beta p \\
IC\,|_{t=0} = (s - d\,TC)/\delta \\
VL\,|_{t=0} = p\,IC/c \\
\frac{\mathrm{d}}{\mathrm{dt}}TC = (s - d\,TC - \beta(1-\eta)TC\,VL)\ \mathbb{1}_{\{0<t<=T_{\text{end}}\}} + (s - d\,TC - \beta\,TC\,VL)\ \mathbb{1}_{\{T_{\text{end}}<t\}} \\
\frac{\mathrm{d}}{\mathrm{dt}}IC = (\beta(1-\eta)TC\,VL - \delta\,IC)\ \mathbb{1}_{\{0<t<=T_{\text{end}}\}} + (\beta\,TC\,VL - \delta\,IC)\ \mathbb{1}_{\{T_{\text{end}}<t\}} \\
\frac{\mathrm{d}}{\mathrm{dt}}VL = ((1-\epsilon)p\,IC - c\,VL)\ \mathbb{1}_{\{0<t<=T_{\text{end}}\}} + (p\,IC - c\,VL)\ \mathbb{1}_{\{T_{\text{end}}<t\}}
\end{cases}
$$

```
; Neumann-Lam model (Neumann et al., Science, 282, 1998)

INPUT:
parameter = {s, d, beta, delta, p, c, eta, epsilon}
regressor = T_end

EQUATION:
odeType = stiff

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; compute the initial values
t0 = 0
TC_0 = delta*c/(beta*p)
IC_0 = (s-d*TC_0)/delta
VL_0 = p*IC_0/c
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; inhibition before and after the end of treatment

if t > T_end
  coeff = 0
else
  coeff = 1
end
eta_cond = coeff*eta
eps_cond = coeff*epsilon

ddt_TC = s - d*TC - beta*(1-eta_cond)*TC*VL
ddt_IC = beta*(1-eta_cond)*TC*VL - delta*IC
ddt_VL = (1-eps_cond)*p*IC - c*VL

LVL = max(log10(VL), 0)

OUTPUT:
output = LVL
```

A system of several ODE can be defined in the block **EQUATION:**. The ODE system is defined as `stiff` using `odeType`. Here, `ddt_TC`, `ddt_IC` and `ddt_VL` are the derivatives with respect to time `t`. `TC_0`, `IC_0` and `VL_0` are the initial values of the system, i.e. the values of `TC`, `IC` and `VL` at time `t0` (they are constant before `t0`). The conditional derivatives are handled by adding the conditional statement for `coeff`. The component `VL` of the ODE system is used to define the prediction of interest.

DDE for a SEIRS epidemic model The model solved is the SEIRS epidemic model of Genik & van den Driessche, example 4.4.1 of [?] defined by:

$$\begin{cases} N(t) = & S(t)+E(t)+I(t)+R(t) \\ \dot{S}(t) = & A(t)-S(t)-\lambda\dfrac{S(t)I(t)}{N(t)}+\gamma I(t-\tau)\exp(-d\tau), \\ \dot{E}(t) = & \lambda\dfrac{S(t)I(t)}{N(t)}-dE(t)-\dfrac{\lambda S(t-\omega)I(t-\omega)\exp(-d\omega)}{N(t-\omega)}, \\ \dot{I}(t) = & -(\gamma+\epsilon+d)I(t)+\dfrac{\lambda S(t-\omega)I(t-\omega)\exp(-d\omega)}{N(t-\omega)}, \\ \dot{R}(t) = & \gamma I(t)-dR(t)-\gamma I(t-\tau)\exp(-d\tau), \qquad t\in[0,350]. \end{cases} \tag{1}$$

The history is given by

$$S(t)=15,\ E(t)=0,\ I(t)=2,\ R(t)=3,\ t\le 0.$$

The parameters are

$$A=0.33,\ d=0.006,\ \lambda=0.308,\ \gamma=0.04,\ \epsilon=0.06, \tau=42,\ \omega=0.15.$$

```
INPUT:
parameter = {A,d,lambda,gamma,epsilon,omega,tau}

EQUATION:

t0 = 0
S_0 = 15
E_0 = 0
I_0 = 2
R_0 = 3
N = S + E + I + R

ddt_S = A - d*S -lambda*S*I/N +gamma*delay(I,tau)*exp(-d*tau)
ddt_E =  - lambda*delay(S,omega)*delay(I,omega)*exp(-d*omega)
             /(delay(I,omega) + delay(S,omega) + delay(E,omega)
                + delay(R,omega))
                + lambda*S*I/N - d*E
ddt_I = - (gamma + epsilon + d)*I
           + lambda*delay(S,omega)*delay(I,omega)*exp(-d*omega)
             /(delay(I,omega) + delay(S,omega)  + delay(E,omega)
                + delay(R,omega))
ddt_R = gamma*I - d*R  - gamma*delay(I,tau)*exp(-d*tau)

OUTPUT:
output = {S, E, I, R}
```

DDE for an Interleukin-2 model The model solved is the Interleukin-2 model studied by Baker et al. in [?] and defined by:

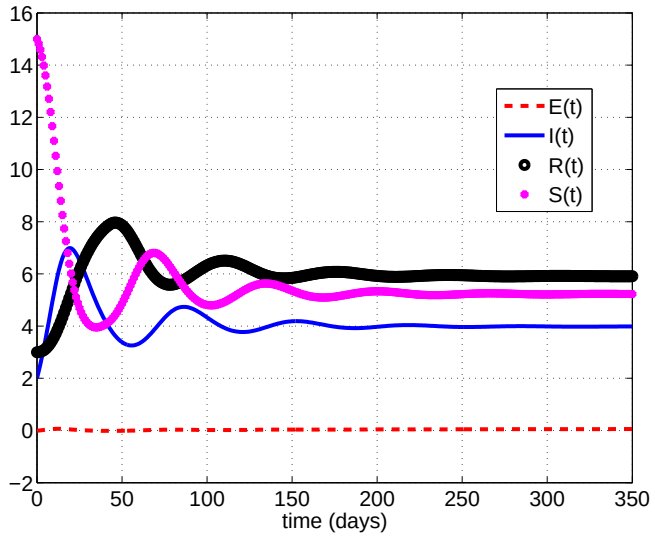


Figure 1: The SEIRS epidemic model of Genik & van den Driessche, example 4.4.1 of [**?**]

$$
\begin{cases}
\dot{I}_2(t) = & -\alpha_{I_2} I_2(t) - n_{I_2} b_{TI_2} \dfrac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t), \\
\dot{T}_A(t) = & \rho b_{TI_2} \dfrac{I_2(t-\tau_D)}{I_2(t-\tau_D)/I_2^* + 1} T_A(t-\tau_D) \\
& - b_{TI_2} \dfrac{I_2(t-\tau_S)}{I_2(t-\tau_S)/I_2^* + 1} T_A(t-\tau_S) - \alpha_{AR} T_A(t), \\
\dot{T}_D(t) = & b_{TI_2} \dfrac{I_2(t-\tau_S)}{I_2(t-\tau_S)/I_2^* + 1} T_A(t-\tau_S) \\
& - b_{TI_2} \dfrac{I_2(t-\tau_D)}{I_2(t-\tau_D)/I_2^* + 1} T_A(t-\tau_D), \\
\dot{T}_R(t) = & \alpha_{AR} T_A(t) - \alpha_R T_R(t), \qquad t \in [0, 100].
\end{cases}
\tag{2}
$$

The parameters of model (2) are

$$
\begin{gathered}
\alpha_{I_2} = 0,\ \alpha_R = 1.5 \cdot 10^{-2},\ \alpha_{AR} = 6.6 \cdot 10^{-2},\ n_{I_2} = 4755, \\
b_{TI_2} = 1.8 \cdot 10^{-11},\ I_2^* = 6 \cdot 10^{10},\ \tau_D = 6.6,\ \tau_S = 10,\ \rho = 2.
\end{gathered}
$$

The model (2) is compared in [**?**] with an ODE model of the same phenomenon, defined as follows:

$$
\begin{cases}
\dot{I}_2(t) = & -\alpha_{I_2} I_2(t) - n_{I_2} b_{TI_2} \dfrac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t), \\
\dot{T}_A(t) = & \rho b_D T_D(t) - b_{TI_2} \dfrac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t) - \alpha_{AR} T_A(t), \\
\dot{T}_D(t) = & b_{TI_2} \dfrac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t) - b_D T_D(t), \\
\dot{T}_R(t) = & \alpha_{AR} T_A(t) - \alpha_R T_R(t), \qquad t \in [0, 100].
\end{cases}
\tag{3}
$$

The parameters of model (2) are

$$\alpha_{I_2} = 0,\ \alpha_R = 3.5 \cdot 10^{-2},\ \alpha_{AR} = 0.02,\ n_{I_2} = 4755,$$
$$b_{TI_2} = 1.0 \cdot 10^{-11},\ I_2^* = 6 \cdot 10^{10},\ b_D = 1/8,\ \rho = 2.$$

The history of models (2) and (3) is given by

$$I_2(t) = 2 \cdot 10^{10},\ T_A(t) = 3.8 \cdot 10^5,\ T_D(t) = 0,\ T_R(t) = 1.2 \cdot 10^5,\ t \le 0.$$

```
INPUT:
parameter = {alphaI2,alphaR,alphaAR,nI2,bTI2,starI2,tauD,tauS,rho}

EQUATION:

t0 = 0
I2_0 = 2e10
TA_0 = 3.8e5
TD_0 = 0
TR_0 = 1.2e5

ddt_I2 = - alphaI2*I2 - nI2*bTI2*I2*TA/(I2/starI2 + 1)
ddt_TA  = rho*bTI2*delay(I2,tauD)*delay(TA,tauD)
                 /(delay(I2,tauD)/starI2 + 1)
                - bTI2* delay(I2,tauS)*delay(TA,tauS)
                /(delay(I2,tauS)/starI2 + 1) - alphaAR*TA
ddt_TD  =  bTI2 *delay(I2,tauS)*delay(TA,tauS)/(delay(I2,tauS)/starI2 +1)
                - bTI2 *delay(I2,tauD)*delay(TA,tauD)
                 /(delay(I2,tauD)/starI2 +1 )
ddt_TR  = alphaAR*TA - alphaR*TR

TV = TA + TD + TR

OUTPUT:
output = TV
```

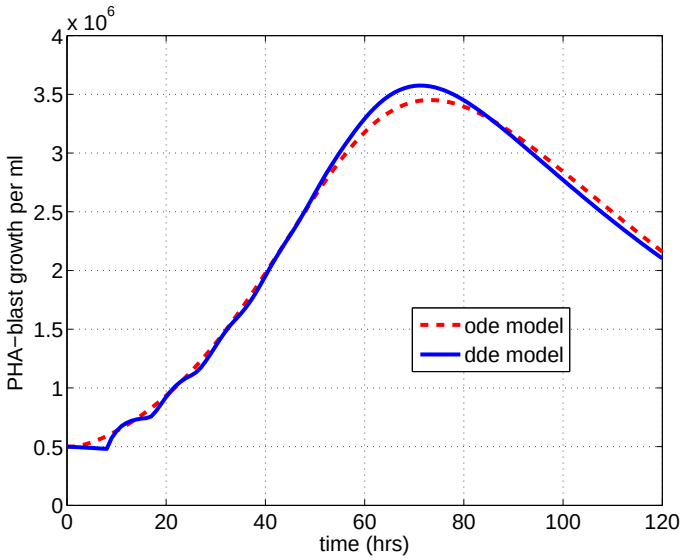


Figure 2: The Interleukin-2: model studied by Baker et al. in [?]

DDE for a PKPD model of unperturbed and perturbed arthritis development The model solved is the PKPD model for unperturbed and perturbed arthritis development of Koch et al. [?] defined by:

$$\begin{cases} \dot{G}(t) = & k_3 - \Big(\sigma_1 \exp\left(-\sigma_2 c(t)\right) + \sigma_3\Big) c(t) G(t) - \dfrac{k_1}{k_2}\Big(1 - \exp\left(-k_2 t\right)\Big) G(t), \\ \dot{I}(t) = & k_4 G(t) - k_4 G(t-\tau), \\ \dot{D}(t) = & k_4 G(t-\tau) - k_5 D(t), \end{cases} \qquad t \in [0, 35].$$

The history is given by

$$G(t) = a \exp(bs),\ I(t) = I_0,\ D(t) = 0,\ t \leq 0.$$

The parameters are

$$a = 1,\ b = 0.5,\ k_1 = 0.183,\ k_2 = 0.092,\ k_3 = 5,\ k_4 = 0.064,\ k_5 = 0.016,\ \tau = 11.2.$$

The parameter $c(t)$ is given by a PK data *dose* of a standard linear 2-compartment i.v. model. The equation used in [?] is

$$c(t) = dose\Big(A_{iv} \exp(-\alpha t) + B_{iv} \exp(-\beta t)\Big).$$

However, $c(t)$ can be computed, knowing the volumes V_1, V_2 of the compartments, the parameters α, β, CL, the *dose* (amount in Mlxtran) and the administration times of the PK model.

```
INPUT:
```

```
parameter = {a, b, alpha, beta, sigma1, sigma2,
                        sigma3, k1, k2, k3,k4,k5,tau,CL,V1,V2}

EQUATION:

t0 = 0
I_0 = 2.52
D_0 =  0
G_0 = a*exp(b*t)

K12 = alpha*beta*V2/CL
K21 = alpha*beta*V1/CL
C = pkmodel(k12=K12,k21=K21,V=V1,Cl=CL)

ddt_G = k3  - (sigma1*exp(- sigma2*C) + sigma3)*C*G
              - k1*(1- exp(- k2*t))*G/k2
ddt_I = k4*G - k4*delay(G,tau)
ddt_D  =  - k5*D + k4*delay(G,tau)

OUTPUT:
output = {G, I, D}
```

Doses of amount 10 where given at time 1, 8, 15.

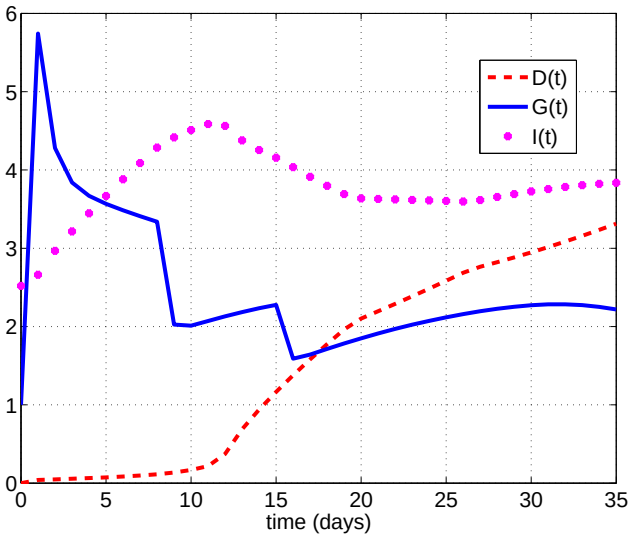


Figure 3: The PKPD model for unperturbed and perturbed arthritis development of Koch et al. [?], Experiment A with the dosing schedules: 10 mg/kg on day 1, 8, 15.

4 The block OBSERVATION:

The block **OBSERVATION:** defines random variables modelling observations of the structural model. Typically, they will be used as model outputs, under the block **OUTPUT:**. An observation

variable can be discrete or continuous. Its actual type and corresponding keywords and equations are set as follows:

Example:

```
OBSERVATION:
Concentration = {type=continuous, prediction=Cc, errorModel=proportional}
```

In the following, most of the keywords and equations are mandatory, so the optional fields are stated.

An observation variable for count data is defined using the `type count`. Its additional fields are:

- `P(Y=k)`: Probability of a given count value `k`, for the observation named `Y`. `k` is a natural number. A transformed probability can be provided instead of a direct one. The transformation can be `log`, `logit`, or `probit`. The bounded variable `k` supersedes in this scope any predefined variable `k`.

Example:

```
OBSERVATION:
Y = {
  type = count
  log(P(Y=k)) = -lambda + k*log(lambda) - factln(k)
}
```

An observation variable for ordered categorical data is defined using the `type categorical`. Its additional fields are:

- `categories`: List of the available ordered categories. They are represented by increasing successive integers.
- `P(Y=i)`: Probability of a given category integer `i`, for the observation named `Y`. A transformed probability can be provided instead of a direct one. The transformation can be `log`, `logit`, or `probit`. The probabilities are defined following the order of their categories. They can be provided for events where the category is a boundary, instead of an exact match. All boundaries must be of the same kind. Such an event is denoted by using a comparison operator. When the value of a probability can be deduced from others, its definition can be spared.

Example:

```
OBSERVATION:
level = {
  type = categorical
  categories = {0, 1, 2, 3}
  logit(P(level<=0)) = th1
```

```
  logit(P(level<=1)) = th1 + th2
  logit(P(level<=2)) = th1 + th2 + th3
}
```

An observation variable for ordered categorical data modelled as a Markov chain is defined using the `type categorical`, along with the `dependence` definition `Markov`. Its additional fields are:

- `categories`: List of the available ordered categories. They are represented by increasing successive integers. It is defined right after `type`.
- `P(Y_1=i)`: Initial probability of a given category integer `i`, for the observation named `Y`. This probability belongs to the first observed value. A transformed probability can be provided instead of a direct one. The transformation can be `log`, `logit`, or `probit`. The probabilities are defined following the order of their categories. They can be provided for events where the category is a boundary, instead of an exact match. All boundaries must be of the same kind. Such an event is denoted by using a comparison operator. When the value of a probability can be deduced from others, its definition can be spared. The initial probabilities are optional as a whole, and the default initial law is uniform.
- `P(Y=j|Y_p=i)`: Probability of transition to a given category integer `j` from a previous category `i`, for the observation named `Y`. A transformed probability can be provided instead of a direct one. The transformation can be `log`, `logit`, or `probit`. The probabilities are grouped by law of transition for each previous category `i`. Each law of transition provides the various transition probabilities of reaching `j`. They can be provided for events where the reached category `j` is a boundary, instead of an exact match. All boundaries must be of the same kind for a given law. Such an event is denoted by using a comparison operator. When the value of a transition probability can be deduced from others within its law, its definition can be spared.

Example:

```
OBSERVATION:
State = {
  type = categorical
  categories = {1,2,3}
  dependence = Markov
  P(State_1=1) = a1
  P(State_1=2) = a2
  logit(P(State<=1|State_p=1)) = a11
  logit(P(State<=2|State_p=1)) = a11+a12
  logit(P(State<=1|State_p=2)) = a21
  logit(P(State<=2|State_p=2)) = a21+a22
  logit(P(State<=1|State_p=3)) = a31
  logit(P(State<=2|State_p=3)) = a31+a32
}
```

An observation variable for ordered categorical data modelled as a continuous Markov chain is also defined using the `type categorical`, along with the `dependence` definition `Markov`. But here transition rates are defined instead of transition probabilities. Its additional fields are:

- `categories`: List of the available ordered categories. They are represented by increasing successive integers. It is defined right after `type`.
- `P(Y_1=i)`: Initial probability of a given category integer `i`, for the observation named `Y`. This probability belongs to the first observed value. A transformed probability can be provided instead of a direct one. The transformation can be `log`, `logit`, or `probit`. The probabilities are defined following the order of their categories. They can be provided for events where the category is a boundary, instead of an exact match. All boundaries must be of the same kind. Such an event is denoted by using a comparison operator. When the value of a probability can be deduced from others, its definition can be spared. The initial probabilities are optional as a whole, and the default initial law is uniform.
- `transitionRate(i,j)`: Transition rate departing from a given category integer `i` and arriving to a category `j`. They are grouped by law of transition for each departure category `i`. One definition of transition rate can be spared by law of transition, as they must sum to zero.

Example:

```
OBSERVATION:
State = {
  type = categorical
  categories = {1,2}
  dependence = Markov
  P(State_1=1) = p1
  transitionRate(1,2) = q12
  transitionRate(2,1) = q21
}
```

An observation variable for time to event or repeated time to event data is defined using the `type event`. Its additional fields are:

- `eventType`: Type of the events. The exact time of the events can be observed, or censored per interval. The respective keywords are `exact` and `intervalCensored`. By default, an exact time is assumed.
- `maxEventNumber`: Maximum number of events. It is useful for simulation only, and by default the number of simulated events is unlimited.
- `rightCensoringTime`: Right censoring time of events. It is useful for simulation only, and by default it is the actual time of the last record.
- `intervalLength`: Length of censoring intervals. It is useful for simulation only, and by default it is the tenth part of the global length.

- `hazard`: Hazard function.

Example:

```
Seizure = {
  type = event
  eventType = intervalCensored
  maxEventNumber = 1
  rightCensoringTime = 120
  intervalLength = 10
  hazard = gamma*Cc
}
```

A continuous observation variable is defined using the `type event`. Its additional fields are:

- `prediction`: Name of the base prediction variable.
- `errorModel`: Error model to apply on the predicted value. Available error models are: `constant`, `proportional`, `combined1`, `combined2`, `proportionalc`, `combined1c`, `combined2c`, `exponential`, `logit`, `band(0,10)` and `band(0,100)`.
- `autocorrelation`: Autocorrelation of the observed values. By default there is no autocorrelation.

Example:

```
OBSERVATION:
Concentration = {
  type = continuous
  prediction = Cc
  errorModel = proportional
  autocorrelation = yes
}
```

5 Pharmacokinetic model/Dynamical system

Pharmacokinetics and/or dynamical systems have a set of domain-specific keywords and macros to handle administrations (i.e. source terms) and compartmental models. Complex absorptions, exchanges, eliminations and so on can be translated using the language of the domain, quite as a diagram would do.

5.1 The pkmodel function

The function `pkmodel` defines a standard PK model. The PK model is defined according to the provided set of named arguments. Most of the arguments are optional, and the `pkmodel`

function enables different parametrizations, to select different models of absorptions, eliminations, etc. A various panel of PK models is available, with one central compartment and one model of absorption. The concentration within the central compartment is the main output of the function. If an effect compartment is defined, its concentration defines a second output. The administration of the doses is always supposed to be of type `1`, and the default absorption is the IV Bolus one. The label of the central compartment is `1`. In addition to the block **`PK:`**, a `pkmodel` can also be defined within a block **`EQUATION:`**.

Example:

```
EQUATION:
Cc = pkmodel(V, Cl)
```

This simple set of named arguments for `pkmodel` defines a PK model with one central compartment of label `1`, with the default IV Bolus absorption for doses of administration type `1` and a linear elimination with the clearance parametrization in `Cl`. The volume of the central compartment is `V`. Its concentration is the output `Cc` of the function.

More complex pk models can easily be defined using the function `pkmodel`

Example:

```
EQUATION:
{Cc, Ce} = pkmodel(Tlag, ka, p, V, Vm, Km, k12, k21, k13, k31, ke0)
```

This more complex set of named arguments for `pkmodel` defines a PK model of one central compartment, with a first order absorption of rate `ka` for doses of administration type `1`, and a Michaelis-Menten elimination of parameters `Vm` and `Km`. The volume of the central compartment is `V`. Its concentration is the first output `Cc` of the function. An effect compartment is defined with a rate `ke0` and its concentration is the second output `Ce`. Two peripheral compartments of labels `2` and `3` are linked to the central compartment of label `1`. The respective couples of exchange rates are `(k12, k21)` and `(k13, k31)`. The first order absorption has a lag time of `Tlag` and the absorbed amount is scaled with a proportion of `p`.

The complete set of named arguments and output for `pkmodel` follows. They belong to the central compartment, the absorption process, the elimination process, the exchanges with the peripheral compartments, or an effect compartment. Some arguments are mutually exclusive, or require other ones to be also defined. Most of them are optional, so the mandatory arguments are stated.

Arguments for the central compartment

- `Cc = pkmodel(...)`: The first output is the concentration within the central compartment. Another name can be used. Mandatory.
- `V`: Volume of the central compartment. Mandatory.

Arguments for the absorption process of a standard PK model

- `Tk0`: Defines a zero order absorption of duration `Tk0`. Excludes `ka`, `Ktr` and `Mtt`.
- `ka`: Defines a first order absorption of rate `ka`. Excludes `Tk0`.
- `Ktr`: Along with `ka` and `Mtt`, defines an absorption with transit compartments of transit rate `Ktr`. Excludes `Tk0`.
- `Mtt`: Along with `ka` and `Ktr`, defines an absorption with transit compartments of mean transit time `Mtt`. Excludes `Tk0`.
- `Tlag`: Lag time before the absorption.
- `p`: Final proportion of the absorbed amount. Can affect the effective rate of the absorption, not its duration.

Arguments for the elimination process of a standard PK model

- `k`: Defines a linear elimination of rate `k`. Excludes `Cl`, `Vm` and `Km`.
- `Cl`: Along with `V`, defines a linear elimination of clearance `Cl`. Excludes `k`, `Vm` and `Km`.
- `Vm`: Along with `V` and `Km`, defines a Michaelis-Menten elimination of maximum elimination rate `Vm`. Excludes `k` and `Cl`.
- `Km`: Along with `V` and `Vm`, defines a Michaelis-Menten elimination of Michaelis-Menten constant `Km`. Excludes `k` and `Cl`.

Arguments for the peripheral compartments of a standard PK model

- `k12`: Along with `k21`, defines a peripheral compartment `2` of input rate `k12`.
- `k21`: Along with `k12`, defines a peripheral compartment `2` of output rate `k21`.
- `k13`: Along with `k31`, defines a peripheral compartment `3` of input rate `k13`.
- `k31`: Along with `k13`, defines a peripheral compartment `3` of output rate `k31`.

Arguments for the effect compartment of a standard PK model

- `{Cc, Ce} = pkmodel(...)`: The second output is the concentration within the effect compartment. Another name can be used.
- `ke0`: Defines an effect compartment with transfer rate `ke0` from the central compartment.

Knowing the label of the central compartment and the administration type supported by the function `pkmodel`, one can build up a custom PK model by adding other PK elements to a base standard PK model. Referencing this label and administration type of value `1` allows to connect the additional PK elements.

5.2 The PK macros

The block **PK:** defines PK elements for a compartmental model. As in the diagram of such a model, those elements are linked to build up the complete model. An element can reference another one using its label. The referenced element must be already defined. The base elements are the compartments. These PK elements provide a greater flexibility for complex PK models beyond the scope of the function `pkmodel`. They can also extend a custom ODE system.

In the following, most of the named arguments are optional, so the mandatory arguments are stated.

Compartments The macro `compartment` defines a PK model compartment.

Dose absorptions can be defined to fill it, and other special compartments can be linked. Its `amount` and `concentration` are variables; it supports defining a prediction or an equation for pharmacodynamics, for example.

Arguments for macro `compartment` are:

- `cmt`: Label of the compartment. Its default value is `1`.
- `amount`: Name of variable defined as the amount within the compartment. Its dynamics are defined by dose absorptions, eliminations, and the transfer rates with other compartments. These dynamics can extend an ODE system that defines a component with this name.
- `volume`: Name of predefined variable to use as the volume of the compartment. It enables to define `concentration`. By default, `concentration` cannot be defined and the volume is `1`.
- `concentration`: Name of the variable defined as the concentration within the compartment.

Example:

```
PK:
compartment(cmt=1, amount=Ac, volume=V, concentration=Cc)
```

If no dose absorption is involved for a `compartment`, and if the amount and all its dynamics are defined as an ODE component with its derivative, then its macro definition can be spared.

Peripheral compartments The macro `peripheral` defines a peripheral compartment. It is equivalent to a simple compartment with two transfers of amount towards and from another compartment. This base compartment must have been previously defined, and is referenced by its label. Usually, only the two transfer rates need to be explicitly defined.

Arguments for macro `peripheral` are:

- `kij` or `k_i_j`: Input rate from the compartment of label `i`. It also defines a label `j` for the peripheral compartment. Here, both labels `i` and `j` must be integers. Mandatory.
- `kji` or `k_j_i`: Output rate to the compartment of label `i`. It also defines a label `j` for the peripheral compartment. Here, both labels `i` and `j` must be integers. Mandatory.
- `amount`: See previous `amount`.
- `volume`: See previous `volume`.
- `concentration`: See previous `concentration`.

Example:

```
peripheral(k12, k21, amount=Ap, volume=V2, concentration=Cp)
```

Effect compartments The macro `effect` defines an effect compartment. It is linked to a simple compartment and used through the variable for its effect concentration.

Arguments for macro `effect` are:

- `cmt`: Label of the linked base compartment. Its default value is `1`.
- `ke0`: Transfer rate from the linked base compartment. Mandatory.
- `concentration`: Name of the variable defined as the concentration within the effect compartment. Mandatory.

Example:

```
effect(cmt=1, ke0, concentration=Ce)
```

Absorptions The design from the project or dataset defines the different administrations of doses. These administrations are singled out using different types. Absorptions for doses are defined by the model and reference the subject administration type. Several absorption processes can be defined for a given administration, the final absorption being their sum. All macros defining an absorption have the following named arguments:

- `type`: Administration type of doses subject to the absorption process. Its default value is `1`. Alias: `adm`.
- `Tlag`: Lag time before the absorption.
- `p`: Final proportion of the absorbed amount. Can affect the effective rate of the absorption, not its duration. `p` can take any positive value (including > 1).

depot The macro depot defines an absorption targeting a depot. A component of an ODE system is defined as the target depot for the doses. Each dose triggers as its absorption an input source term for this system, shifting the depot variable by a proportion of the dose amount. A zero order, first order, or transit process can also be defined, using the same argument sets as the macro absorption, apart from cmt.

Arguments for macro depot are:

- target: Name of the component of an ODE system that is shifted by the absorption. Mandatory.
- type/adm: type of administration (type and adm are equivalent).
- Tlag
- p
- Tk0: Duration of the zero order process.
- ka: Rate of the first order process.
- Ktr, Mtt: Define transit components of transit rate Ktr and mean transit time Mtt for a transit process.

Example:

```
PK:
depot(type=1, target=Ad, Tlag, p=F)
depot(type=2, target=Ac, Tk0)

EQUATION:
ddt_Ad = -ka*Ad
ddt_Ac =  ka*Ad - k*Ac
```

Example:

```
PK:
depot(type=1, target=Ac, Tlag, p=F, ka)
depot(type=2, target=Ac, Tk0)

EQUATION:
ddt_Ac =  - k*Ac
```

IV The macro iv defines an absorption for intravenous doses. Doses without an administration rate or infusion time are instantaneously absorbed within the associated compartment, as an IV bolus. Other doses are absorbed according to a zero order process, as an IV infusion.

Arguments for macro iv are:

- cmt: Label of the compartment filled in by the absorption process. Its default value is 1.

- type/adm
- Tlag
- p

Example:

```
PK:
iv(type=1, cmt=1, p)
```

Zero order absorption processes The macro absorption with an argument Tk0 defines a zero order absorption process

As a common model for oral administrations, administration has the alias oral.

Arguments for macro administration/oral used with $Tk0$ are:

- cmt: Label of the compartment filled in by the absorption process. Its default value is 1.
- Tk0: Duration of the zero order absorption (mandatory),
- type/adm
- Tlag
- p

Example:

```
PK:
absorption(cmt=1, Tk0, Tlag)
```

First order absorption The macro absorption with an argument ka defines a first order absorption process.

Arguments for macro administration/oral used with ka are:

- cmt: Label of the compartment filled in by the absorption process. Its default value is 1.
- ka: rate of the first order absorption (mandatory),
- type/adm
- Tlag
- Ktr, Mtt: Define transit compartments of transit rate Ktr and mean transit time Mtt for the absorption.

- p

Example:

```
PK:
absorption(cmt=1, ka, p=F)
```

Eliminations Different elimination processes can be defined for compartments. Several eliminations can even be defined for the same compartment, the final elimination flow being their sum. All macros defining an elimination have the following named arguments:

- cmt: Label of the compartment emptied by the elimination process. Its default value is 1.
- V: Volume involved in the elimination process. Its default, and usual, value is the volume of the emptied compartment.

Linear elimination The macro elimination with an argument k or Cl defines a linear elimination.

Arguments for macro elimination used for a linear elimination are:

- k: Rate of the elimination. Mandatory, unless Cl is defined instead of.
- Cl: Clearance of the elimination. Mandatory, unless k is defined instead of.

Example:

```
elimination(cmt=1,k)
```

Michaelis-Menten elimination The macro elimination with arguments Vm and Km defines a Michaelis-Menten elimination.

Arguments for macro elimination used for a non linear elimination are:

- Vm, Km: Maximum elimination rate and Michaelis-Menten constant. Mandatory.

Example:

```
elimination(cmt=1,Vm,Km)
```

Transfers The macro `transfer` defines a transfer of amount from a first compartment to a second one. Its named arguments are:

- `from`: Label of the source compartment for the transfer. Its default value is `1`.
- `to`: Label of the target compartment for the transfer. Its default value is `1`.
- `kt`: Rate of the transfer. Mandatory.

For usual compartmental models, it is often easier to use `peripheral`.

Example:

```
transfer(from=1, to=2, kt)
```

5.3 Independent variables from doses

Doses from the design can also be referenced through several matching independent variables. This allows for some direct calculation, especially for modelling custom single dose absorptions. Doses aren't discriminated according to their administration types here. These values are directly related to the design, they are unaffected by the PK elements.

Time of the last administered dose The keyword `tDose` defines the time of the last administered dose. This is a step function. Its value is unaffected by any lag time `Tlag`. Indeed, a lag time targets the dose absorption. It is null before the first dose.

Amount of the last administered dose The keyword `amtDose` defines the amount of the last administered dose. This is a step function. Its value is unaffected by any proportional factor `p`, as this keyword targets the absorbed amount by the end of the whole absorption process. It is null before the first dose.

Infusion time of the last administered dose The keyword `inftDose` defines the infusion time of the last administered dose. This is a step function. Its value is unaffected by any proportional factor `p`, as this keyword targets the absorbed amount by the end of the whole absorption process. The rate of the final absorption can be different from the rate of the administration process. It is null before the first dose.

5.4 Analytical solutions

5.4.1 General description

In PK modelling some very common models have a known analytical solution, therefore not requiring to call the ODE solver. Using analytical solutions allows to get exact results, instead of approximations bu the ODE solver, and to save computation time.

For instance the following script:

```
INPUT:
parameter = {V, k}
EQUATION:
Cc = pkmodel(V, k, k12=0.2, k21=0.3)
OUTPUT:
output = Cc
```

describes a classical two compartments model whose solution is given in [*Mathematical Expressions of the Pharmacokinetic and Pharmacodynamic Models implemented in the Monolix software, September 2008*]

MLXTRAN automatically detects all the common models described in [*Mathematical Expressions of the Pharmacokinetic and Pharmacodynamic Models implemented in the Monolix software, September 2008*] and when such a model is used **and its coefficients do not depend on time**, the ODE system is replaced by its corresponding analytical solution.

Hereunder a non-exhaustive list of situations where an analytical solution will be used:

- It is possible to require concentration / drug amount in a peripheral compartment. For instance, following model will create an analytical solution:

```
INPUT:
parameter = {V}
PK:
compartment(cmt=1, volume=V, concentration=Cc)
iv(cmt=1)
elimination(cmt=1, k=0.2)
peripheral(k12=0.6, k21=0.8, amount=Ap)
EQUATION:
Cc = 3*t + sin(Ap)
OUTPUT:
output = Cc
```

  Note from the previous model that an analytical combination of analytical solutions is still an analytical solution.

- As already stated there is no analytical solution when a coefficient of the ODE system (such as a volume, a transfer coefficient between compartments or an absorption rate) depends on time. However dose coefficients can depend on time, since their time-dependency is only assessed once, when the dose is delivered. As a consequence the drug proportion, time lag or drug amount can depend on time and the following model will create an analytical solution

```
INPUT:
parameter = {V}
```

```
PK:
compartment(cmt=1, volume=V, concentration=Cc)
iv(cmt=1, p=amtDose/(amtDose+10), Tlag = t/(t + 10))
elimination(cmt=1, k=0.2)
peripheral(k12=0.6*V/(V + 0.1), k21=0.8, amount=Ap)
EQUATION:
out = 3*t + 2*Cc + sin(Ap)
OUTPUT:
output = out
```

- Note from above model that an analytical formula can be used in the definition of the transfer rate (see k12 in model hereunder). However the transfer rate shall not depend on time, otherwise the analytical solution is no longer valid.
- It is possible to mix administration types and to generate mutiple analytical solutions. Below is an example with iv and oral administration which contains two analytical solutions: a first one for a three compartments model (compartments 1, 2 and 3) and a second one for a two compartments model (compartments 5 and 6)

```
INPUT:
parameter = {alpha, beta}
PK:
compartment(cmt=1, volume=V, concentration=Cc)
iv(cmt=1, p=amtDose/(amtDose+10), Tlag = t/(t + 10),type=1)
oral(cmt=1, Tk0=0.1, type=2)
elimination(cmt=1, k=0.2)
peripheral(k12=0.6, k21=0.8, amount=Ap2)
peripheral(k13=0.6 + alpha, k21=0.8 + beta, amount=Ap3)
compartment(cmt=5, volume=2*V, concentration=Cc)
iv(cmt=5,type=1)
oral(cmt=5, Tk0=0.1, type=6)
elimination(cmt=5, k=0.2)
peripheral(k56=0.6, k65=0.8, amount=Ap6)
OUTPUT:
output = Cc
```

Note that when there are several administrations the solution is computed as the sum of the solutions for each administration (since the system is linear).

Below is a non-exhaustive list of situations where analytical solutions cannot be used:

- If volume depends on time, analytical solutions cannot be used:

```
INPUT:
parameter = {V}
PK:
compartment(cmt=1, volume=V+2*t, concentration=Cc)
iv(cmt=1)
elimination(cmt=1, k=0.2)
peripheral(k12=0.6, k21=0.8, amount=Ap)
EQUATION:
```

```
Cc = 3*t + sin(Ap)
OUTPUT:
output = Cc
```

- If `pkmodel` macros are replaced by their corresponding ODE system, the analytical solution will not be generated. For instance the following script will not generate an analytical solution:

```
INPUT:
parameter = {k}
EQUATION:
ddt_Ap = -k*Ap
OUTPUT:
output = Ap
```

  but its macro counterpart will generate one:

```
INPUT:
parameter = {k}
PK:
compartment(cmt=1, amount=Ap)
elimination(cmt=1, k)
OUTPUT:
output = Ap
```

- Analytical solutions will not be generated when Michaelis Menten eliminations are used (Mtt coefficient), maximum elimination rate (Vm coefficient), transit compartments with transit rate (see Ktr coefficient) or mean transit time (see Mtt coefficient). Indeed such systems do not have an an analytical solution and amounts in the different compartments can only be computed by solving the ODE system.
- When the input data file contains IOV, coefficients of the ODE can vary between occasions and hence with time. Therefore there is no analytical solution generated in this case.

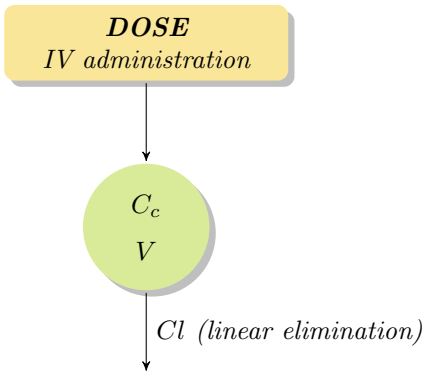
5.4.2 Miscellaneous

- **How to set / unset analytical functions:** In the configuration file, inside the **[PlugInCodeGeneration]** section, the parameter `bUseAnalyticalFunctions` can be set to true / false to switch ON / OFF the use of analytical solutions. The configuration file is located at: `<UserHome>/lixoft/monolix/config/config.ini`
- **How to compare analytical and ODE results ?** Users who wish to verify the consistency of the results from the ODE and from the analytical solution can run the same model twice as follows: first run with `bUseAnalyticalFunctions` set to `true` and second time set to `false`. The configuration file is checked at software startup, therefore the software needs to be restarted for the modification to be taken into acount.
- **How to check if an analytical solution is used ?** Several possibilities:

- run the model twice with different settings as described above; if the results are slightly different it means that an analytical solution is used when requested, since the ODE solution is an approximation of the analytical solution.
- examine the plug-in source code and search for string `analytical_setParameters` which means that plug-in calls an analytical solution.

5.5 Examples

1-compartment model with IV administration and linear elimination The prediction of interest is the concentration within the central compartment of a 1-compartment model with an IV administration and a linear elimination.




```
; PK model, IV administration,
; linear elimination

[PREDICTION]
input = {V, Cl}

PK:
Cc = pkmodel(V, Cl)
```

The block `PK:` describes the PK model. The function `pkmodel` with arguments `V` and `Cl` defines a linear elimination with a parameterization in clearance, in addition to its default output concentration and IV absorption.

Combining three macros to define the central compartment, the IV absorption, and the linear elimination builds up the same PK model. As there is only one compartment the label `cmt` can be left with its default value for these macros. The concentration and the elimination are based on the `volume` defined within the compartment macro. The absorption macro also keeps for its subject doses the default administration `type`.

Example:

```
PK:
compartment(cmt=1,concentration=Cc, volume=V)
iv(cmt=1)
elimination(cmt=1,Cl)
```

This model can also be defined with an ODE system. The system describes the elimination and the doses define input source terms of the system, with a `depot` macro.

Example:

```
PK:
depot(target=Ac)
EQUATION:
ddt_Ac = -Cl/V * Ac
```

```
Cc = Ac/V
```

The absorption model could be modified to include a lag time

Example:

```
Cc = pkmodel(Tlag, V, Cl)
```

or a proportion on the absorbed amount

Example:

```
Cc = pkmodel(p, V, Cl)
```

The whole absorption could be replaced with a zero order absorption

Example:

```
Cc = pkmodel(Tk0, V, Cl)
```

or a first order absorption

Example:

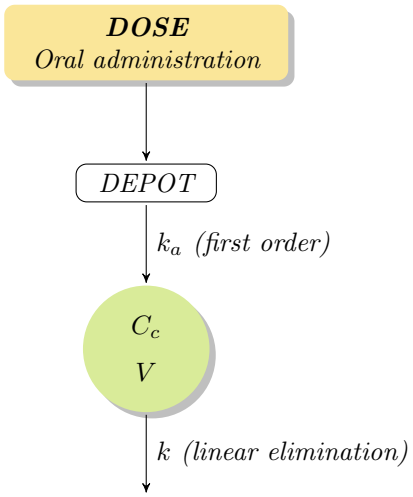
```
Cc = pkmodel(ka, V, Cl)
```

or a first order absorption with transit compartments

Example:

```
Cc = pkmodel(ka, Ktr, Mtt, V, Cl)
```

Analytical solution for a 1-compartment model with single dose The prediction of interest is the concentration within the central compartment of a 1-compartment model with an oral administration, first order absorption, and a linear elimination.

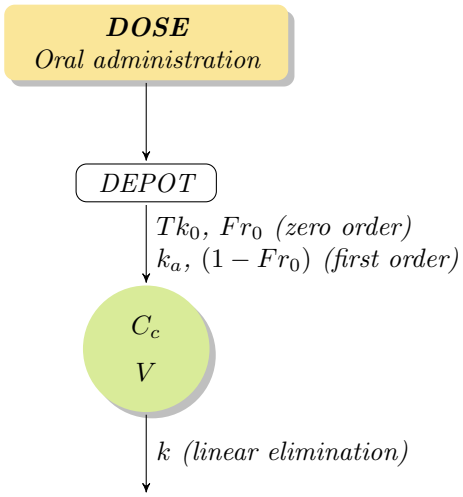



```
; PK model, oral administration,
; first order absorption, linear elimination

[PREDICTION]
input = {ka, V, k}
EQUATION:
dt = max(t-tDose, 0)
Cc = amtDose*ka/(V*(ka-k))
     *(exp(-k*dt)-exp(-ka*dt))
```

The analytical solution is defined here for a single dose problem and is based on `tDose` and `amtDose`.

Sequential zero order-first order absorption processes The prediction of interest is the concentration within the central compartment of a 1-compartment model with an oral administration, a sequential zero order-first order absorption, and a linear elimination.



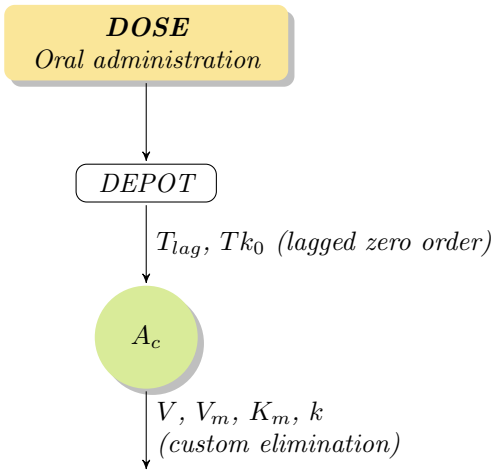

```
; PK model, oral administration, sequential
; zero-order & first order absorptions,
; linear elimination

[PREDICTION]
input = {Fr0, Tk0, ka, V, k}

PK:
compartment(concentration=Cc, volume=V)
oral(Tk0, p=Fr0)
oral(ka, Tlag=Tk0, p=1-Fr0)
elimination(k)
```

Two different absorptions are defined, for the same administration `type` and with the same compartment to fill in. A linear elimination is defined for the same compartment. The administration `type` is the default value, 1, as is the label of the subject compartment for the different macros here. The two absorptions defined for the compartment are cumulated, each dose of this `type` triggers both of them. The artificial lag time allows the first order absorption to come after the zero order one, in a sequence. Each one concerns only a fraction of the administered amount, respectively fractions $Fr0$ and $(1 - Fr0)$.

Custom elimination with an absorption macro as extension The prediction of interest is the concentration within the central compartment of a 1-compartment model with an oral administration, a zero order absorption with a lag time, and an elimination cumulating a linear and a Michaelis-Menten non-linear ones.



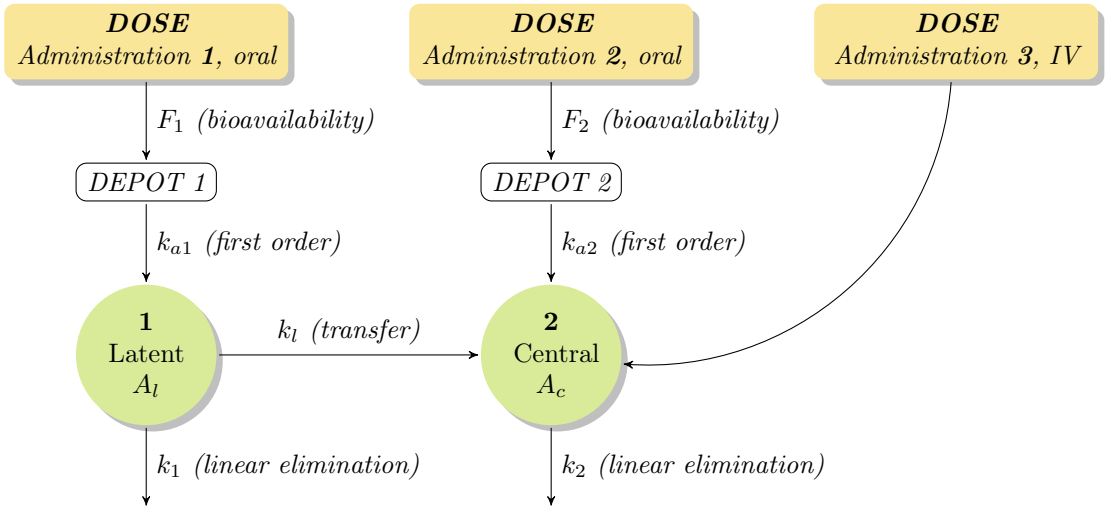

```
; PK model, oral administration,
; zero order absorption with a lag time,
; combination of linear and nonlinear
; eliminations

[PREDICTION]
input = {Tlag, Tk0, V, Vm, Km, k}

PK:
compartment(amount=Ac)
oral(Tlag, Tk0)
EQUATION:
ddt_Ac = -Vm*Ac/(V*Km+Ac) - k*Ac
Cc = Ac/V
```

The custom elimination is defined with an ODE system and is extended with an absorption macro. It requires to define a name for the amount variable that will be an ODE component as well as the quantity ruled by the absorption macro.

Complex PK model with a latent compartment The predictions of interest are the concentration within the central compartment of a 2-compartment model, and the amount within the second, latent, compartment. There are three administration types, a transfer from the latent compartment to the central one, and a linear elimination for each compartment. Each compartment has a first order absorption with its bioavailability. In addition, the central compartment has an IV absorption.




```
; PK model, combination of oral and IV administrations,
; a latent compartment and linear eliminations

[PREDICTION]
input = {F1, F2, ka1, ka2, kl, V, k1, k2}

PK:
```

```
compartment(cmt=1, amount=Al)
compartment(cmt=2, amount=Ac)
oral(type=1, cmt=1, ka=ka1, p=F1)
oral(type=2, cmt=2, ka=ka2, p=F2)
iv(type=3, cmt=2)
transfer(from=1, to=2, kt=kl)
elimination(cmt=1, k=k1)
elimination(cmt=2, k=k2)
Cc = Ac/V
```

Each administration is singled out by its `type`, and get an absorption defined for its doses. Alike, each compartment is singled out by its `cmt`, and is referenced by other PK elements using it. Two different absorptions contribute to fill in the central compartment directly, in addition to the transfer coming from the latent compartment. The definition for a compartment precedes the elements that refers to it. The concentration is defined by a calculus based on the amount of interest, within the central compartment. Parameters `volume` and `concentration` could have been set up for the central compartment, instead of this equation.

6 The block OUTPUT:

The block **`OUTPUT:`** declares variables whose values are exported for the various tasks. For example, they can be processed by the algorithms, used in graphics, or registered in files. The keyword `output` declares the main variables to output, whereas `table` declares the variables to record in tables.

Example:

```
OUTPUT:
output = {Concentration, Effect}
table = {Ap, Rin}
```

6.1 Examples

Analytical solution of an exponential decay model The prediction of interest is the analytical solution of an exponential decay model. An initial state of equilibrium is followed by an exponential decay after the time of reference.

$$f = \begin{cases} A & \text{if } t <= 5 \\ A\,e^{-k\,(t-5)} & \text{if } t > 5 \end{cases}$$

```
INPUT:
parameter = {A,k}

EQUATION:
T0=5
if t<T0
   f=A
else
   f= A*exp(-k*(t-T0))
end

OUTPUT:
output = f
```

Exponential decay model with an ODE This is the same model as before, but it is defined by an ODE (Ordinary Differential Equations) initial value problem.

$$\begin{cases} f = A & \text{if } t <= 5 \\ f'(t) = -k\,f(t) & \text{if } t > 5 \end{cases}$$

```
INPUT:
parameter = {A,k}

EQUATION:
t0=5
f_0=A
ddt_f= - k*f

OUTPUT:
output = f
```

An ODE (Ordinary differential equations) can be defined in the block `EQUATION:`. Here, `ddt_A` is the derivative of `A` with respect to time `t`. `A_0` is the initial value of the system, i.e. the value of `A` at time `t0` (`A` is constant before `t0`).

7 Miscallenous

7.1 Macros

The macros have a syntax that can be similar to the syntax of an usual function, as `min`, but they define themselves elements of the model such as a compartment for example. Alike, they can define variables as an output, but they cannot be included themselves into an arithmetic expression, as an usual function. Defining the very structure of the model, they cannot be enclosed within a conditional statement.

More importantly, their arguments are named arguments, as follows.

Most arguments of a macro are fully optional and provide a default value, or are part of separated sets of arguments. Therefore the arguments of a macro are not identified according

to their ordering, but to their name. These named arguments follow a special lookup to define their values.

Indeed, if a variable is already defined with the same name as the argument, listing this name within the macro call is sufficient for the lookup to pick up the value of the argument from the variable. This assumes that the value type is compatible with the argument, thus it is available for equations, but not for labels or pure names.

The other option is simply to define the value of the argument inline within the macro call, as an equality.

Example:

```
ka = 0.8*t
F = 0.3
oral(type=2, ka, p=1-F)
; Argument "ka" denotes the set of arguments for a first order absorption.
; Argument "type" gets an inline label of "2".
; Argument "ka" gets the equation "0.8*t" from the variable "ka".
; Argument "p" gets an inline equation of "1-F".
```

The output of a macro is a single variable, or a list of variables. Their equations are defined by the macro itself, according to its input arguments, and their names are provided as an output list. Therefore, contrary to input arguments, the ordering or output argument matters.

Example:

```
{out1, out2} = pkmodel(Tk0=2.5, V=10, k=0.3, ke0=0.7)
; Argument "ke0" enables an effect compartment.
; A variable "out1" is defined as the central compartment's concentration.
; A variable "out2" is defined as the effect compartment's concentration.
```

7.2 Main independent variable

The keyword `t` defines the main independent variable. This is the variable itself, not its last record from the design. This is also the differentiation variable for any ODE and the time variable for the pharmacokinetics.

7.3 Annotations

The model can be documented using comments.

A comment begins from the character `;` to the end of the line. It is ignored by the parser.

Example:

```
V={distribution=lognormal, typical=V_pop, sd=omega_V} ; This is a comment.
; This is also a comment.
```

7.4 Reserved names

Some naming patterns and keywords are reserved, and cannot be used as new names.

The naming patterns for the ODE derivatives and initial values are reserved.

The names of the usual mathematical functions and distribution functions are reserved.

The list of keywords from table 1 is reserved.

Table 1: Reserved keywords

absorption	continuous	false	odeType	t_0
adm	count	from	oral	t0
amount	dependence	hazard	OUTPUT	table
amtDose	depot	if	output	target
autocorrelation	DESCRIPTION	inftDose	P	tDose
band	effect	INPUT	parameter	to
bsmm	elimination	input	peripheral	transfer
categorical	else	intervalCensored	PK	transitionRate
categories	elseif	intervalLength	pkmodel	true
cmt	end	iv	prediction	type
combined1	EQUATION	linear	proportional	volume
combined1c	error	Markov	proportionalc	wsmm
combined2	errorModel	maxEventNumber	regressor	xDose
combined2c	event	no	rightCensoringTime	yes
compartment	eventType	nonStiff	stiff	
concentration	exact	normal	student	
constant	exponential	OBSERVATION	t	