

MLXTRAN

THE MODEL CODING LANGUAGE FOR

- MONOLIX
- MLXPLORE¹
- SIMULIX

February 2014 •

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

Contents

1	General structure of MLXTRAN	3
2	The bloc DEFINITION:	4
2.1	Probability distributions	4
2.2	Definition of a statistical model	5
3	The block EQUATION:	6
3.1	Intermediate variable	7
3.2	Ordinary Differential Equations	7
3.3	Delayed Differential Equations	8
3.4	Usual mathematical functions	9
3.5	Probability distribution functions	10
3.6	Operators	11
3.7	Conditional statements	13
3.8	Examples	13
4	The section [COVARIATE]	19
5	The section [INDIVIDUAL]	20
6	The section [LONGITUDINAL]	22
6.1	The structural model	22
7	Pharmacokinetic model/Dynamical system	22
7.1	The pkmodel function	23
7.2	The PK macros	25
7.3	Independent variables from doses	30
7.4	Analytical solutions	30
7.4.1	General description	30
7.4.2	Miscellaneous	33

7.5	Examples	33
7.6	Examples	37
8	Miscellaneous	38
8.1	Macros	38
8.2	Main independent variable	39
8.3	Annotations	39
8.4	Reserved names	39

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.

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.

[LONGITUDINAL] The structural model, i.e. functions which vary with time, is implemented in a section `[LONGITUDINAL]`. The structural model is a parametric model, i.e. it is totally defined by a finite vector of parameters ψ .

Section `[LONGITUDINAL]` is mandatory.

[INDIVIDUAL] If the vector of parameters ψ is assumed to have a probability distribution, then this distribution is implemented in a section `[INDIVIDUAL]`. This distribution depends on a vector of population parameters and possibly on a vector of covariates.

Section `[INDIVIDUAL]` is optional.

[COVARIATE] If the individual covariates are assumed to have a probability distribution, this distribution is implemented in a section `[COVARIATE]`. It depends on a vector of population parameters.

Section `[COVARIATE]` is optional.

Any of these sections is composed of several blocks:

input A section starts with the declaration of the *inputs* of the section, i.e. the variables used in the associated submodel. According to the context, those variables may be defined in a previous section of the model or in the project.

EQUATION: In each section, MLXTRAN supports flexible equation-based descriptions implemented in a block `EQUATION:`.

The block `EQUATION:` of the section `[LONGITUDINAL]` can include ODEs.

DEFINITION: For sections `[INDIVIDUAL]` and `[COVARIATE]`, MLXTRAN supports explicit definition-based descriptions of probability distributions in a block `DEFINITION:`.

For Section `[LONGITUDINAL]` only:

PK: For PK models, section `[LONGITUDINAL]` can include a block `PK:` (optional).

2 The bloc DEFINITION:

An explicit definition of a random variable can be provided in the block **DEFINITION:**.

2.1 Probability distributions

A random variable X can be defined through its probability distribution. It is a list including a `distribution` name with specific parameters. The elements of this list are identified according to their names (the order does not matter).

Example:

```
[INDIVIDUAL]
input= {mu, sigma}

DEFINITION:
X = {distribution=normal, mean=mu, sd=omega}
```

MLXTRAN for MLXPLORE 1 only supports distributions based on the normal distribution. More precisely, we consider that there exists a monotonic transformation h such that $h(X)$ is normally distributed:

probability distribution	<code>distribution</code>	transformation h
normal	<code>normal</code>	$h(X) = X$
log-normal	<code>lognormal</code>	$h(X) = \log(X)$
logit-normal	<code>logitnormal</code>	$h(X) = 1/(1 + e^{-X})$
probit-normal	<code>probitnormal</code>	$h(X) = \Phi^{-1}(X)$

where Φ is the cumulative distribution function of the $\mathcal{N}(0, 1)$ distribution.

These distributions are defined by the mean and standard deviation of the normal distribution, i.e of the transformed variable $h(X)$.

Assume for example that X is log-normally distributed:

$$X \sim \log \mathcal{N}(\mu, \omega) \Leftrightarrow \log(X) \sim \mathcal{N}(\mu, \omega)$$

Here, μ and ω are the mean and standard-deviation of $\log(X)$.

```
[INDIVIDUAL]
input= {mu, sigma}

DEFINITION:
X = {distribution=lognormal, mean=mu, sd=omega}
```

The variance can be used instead of the standard-deviation

Example:

```
[INDIVIDUAL]
input= {mu, sigma2}
```

DEFINITION:

```
X = {distribution=logitnormal, mean=mu, var=sigma2}
```

In summary, the following elements are available for defining a distribution:

- **distribution**: probability distribution. Available values: `normal`, `lognormal`, `logitnormal`, `probitnormal`.
- **mean**: mean of the associated normal distribution.
- **sd**: standard deviation of the associated normal distribution (`standardDeviation` and `sd` are synonymous keywords).
- **var**: The variance of the associated normal distribution (`variance` and `var` are synonymous keywords).

2.2 Definition of a statistical model

MLXTRAN allows to encode very general linear and non linear statistical models, involving covariates and based on probability distributions.

Linear Gaussian model: a linear Gaussian statistical model for the variable X assumes that there exists a transformation h , a typical value X_{pop} , a vector of individual covariates (c_1, \dots, c_L) , a vector of coefficients $(\beta_1, \dots, \beta_L)$ and a random variable η normally distributed such that

$$h(X) = h(X_{pop}) + \sum_{\ell=1}^L \beta_{\ell} c_{\ell} + \eta$$

This model can be implemented with MLXTRAN, using the keywords `typical`, `covariate` and `coefficient`.

Example:

DEFINITION:

```
X = {distribution=lognormal, typical=Xpop, covariate={c1, c2},
      coefficient={beta1, beta2}, sd=omega}
```

Non linear Gaussian model: a nonlinear Gaussian statistical model for the variable X assumes that there exists a transformation h , a vector of individual covariates (c_1, \dots, c_L) , a vector of coefficients $(\beta_1, \dots, \beta_M)$, a function μ and a random variable η normally distributed such that

$$h(X) = \mu(c_1, \dots, c_L, \beta_1, \dots, \beta_M) + \eta$$

The mean of $h(X)$ can be defined in a block EQUATION:

Example:

EQUATION:

```
mu = log(beta1*c1/(beta2 + c2))
```

DEFINITION:

```
X = {distribution=lognormal, mean=mu, sd=omega}
```

Instead of using the mean of the transformed parameters $h(X)$, another parametrization which involves the prediction of the parameter itself can be preferred. For example, let

$$\tilde{X} = \frac{\beta_1 c_1}{\beta_2 + c_2}$$

such that $h(X) \sim \mathcal{N}(h(\tilde{X}), \omega)$

EQUATION:

```
Xpred = beta1*c1/(beta2 + c2)
```

DEFINITION:

```
X = {distribution=lognormal, prediction=Xpred, sd=omega}
```

More generally, if X is a random variable such that $h(X) \sim \mathcal{N}(\mu, \omega)$, the mean μ and the prediction m satisfy $h(m) = \mu$.²

Non Gaussian model: a non Gaussian model for X can be defined, at the condition that X can be defined as a nonlinear function of normally distributed random variables.

For example, let

$$X = \frac{\beta_1 + \eta_1}{1 + \beta_2 e^{\eta_2}}$$

It is not possible to express explicitly the distribution of X as a transformation of a normal distribution. We therefore need a block EQUATION for implementing this model:

DEFINITION:

```
eta1 = {distribution=normal, mean=0, sd=omega1}
```

```
eta2 = {distribution=normal, mean=0, sd=omega2}
```

EQUATION:

```
X = (beta1 + eta1)/(1+beta2*exp(eta2))
```

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.

²Indeed, considering the mathematical representation $h(X) = \mu + \eta$ where $\eta \sim \mathcal{N}(0, \omega)$, the prediction of X is obtained by setting $\eta = 0$.

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= {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 τ 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 7.5 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

$t > t_0$ they are defined as the solution of the ODE system with initial values fixed at $t = t_0$. 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 $t \leq t_0$. Then, for $t > t_0$, 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.

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.

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

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

3.6 Operators

The following operators are available, under their mathematical acceptance.

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. Right-associative. Takes precedence on unary plus and minus.
Addition	$a + b$	
Subtraction	$a - b$	
Multiplication	$a * b$	
Division	a / b	
Power	$a ^ b$	
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	$\sim a$	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	$a == b$	Another syntax is $a != b$.
Not equal to	$a \neq b$	
Greater than	$a > b$	
Less than	$a < b$	
Greater than or equal to	$a \geq b$	
Less than or equal to	$a \leq b$	

3.7 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
```

3.8 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.

$$\left\{ \begin{array}{l} TC|_{t=0} = \delta c / \beta p \\ IC|_{t=0} = (s - dTC) / \delta \\ VL|_{t=0} = pIC / c \\ \frac{d}{dt} TC = (s - dTC - \beta(1 - \eta)TCVL) \mathbb{1}_{\{0 < t \leq T_{\text{end}}\}} + (s - dTC - \beta TCVL) \mathbb{1}_{\{T_{\text{end}} < t\}} \\ \frac{d}{dt} IC = (\beta(1 - \eta)TCVL - \delta IC) \mathbb{1}_{\{0 < t \leq T_{\text{end}}\}} + (\beta TCVL - \delta IC) \mathbb{1}_{\{T_{\text{end}} < t\}} \\ \frac{d}{dt} VL = ((1 - \epsilon)pIC - cVL) \mathbb{1}_{\{0 < t \leq T_{\text{end}}\}} + (pIC - cVL) \mathbb{1}_{\{T_{\text{end}} < t\}} \end{array} \right.$$

```

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

[LONGITUDINAL]
input = {s, d, beta, delta, p, c, eta, epsilon, 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)

```

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:

$$\left\{ \begin{array}{l} N(t) = S(t) + E(t) + I(t) + R(t) \\ \dot{S}(t) = A(t) - S(t) - \lambda \frac{S(t)I(t)}{N(t)} + \gamma I(t - \tau) \exp(-d\tau), \\ \dot{E}(t) = \lambda \frac{S(t)I(t)}{N(t)} - dE(t) - \frac{\lambda S(t - \omega)I(t - \omega) \exp(-d\omega)}{N(t - \omega)}, \\ \dot{I}(t) = -(\gamma + \epsilon + d)I(t) + \frac{\lambda S(t - \omega)I(t - \omega) \exp(-d\omega)}{N(t - \omega)}, \\ \dot{R}(t) = \gamma I(t) - dR(t) - \gamma I(t - \tau) \exp(-d\tau), \end{array} \right. \quad t \in [0, 350]. \quad (1)$$

The history is given by

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

The parameters are

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

```
[LONGITUDINAL]
input = {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)
```

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

$$\left\{ \begin{array}{l} \dot{I}_2(t) = -\alpha_{I_2} I_2(t) - n_{I_2} b_{TI_2} \frac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t), \\ \dot{T}_A(t) = \rho b_{TI_2} \frac{I_2(t - \tau_D)}{I_2(t - \tau_D)/I_2^* + 1} T_A(t - \tau_D) \\ \quad - b_{TI_2} \frac{I_2(t - \tau_S)}{I_2(t - \tau_S)/I_2^* + 1} T_A(t - \tau_S) - \alpha_{AR} T_A(t), \\ \dot{T}_D(t) = b_{TI_2} \frac{I_2(t - \tau_S)}{I_2(t - \tau_S)/I_2^* + 1} T_A(t - \tau_S) \\ \quad - b_{TI_2} \frac{I_2(t - \tau_D)}{I_2(t - \tau_D)/I_2^* + 1} T_A(t - \tau_D), \\ \dot{T}_R(t) = \alpha_{AR} T_A(t) - \alpha_R T_R(t), \end{array} \right. \quad t \in [0, 100]. \quad (2)$$

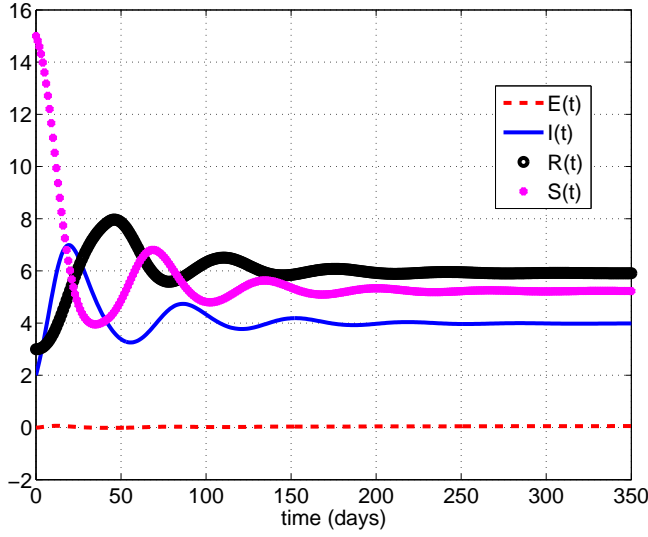


Figure 1: The SEIRS epidemic model of Genik & van den Driessche, example 4.4.1 of [?]

The parameters of model (2) are

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

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

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

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

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

[LONGITUDINAL]

```
input = {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

```

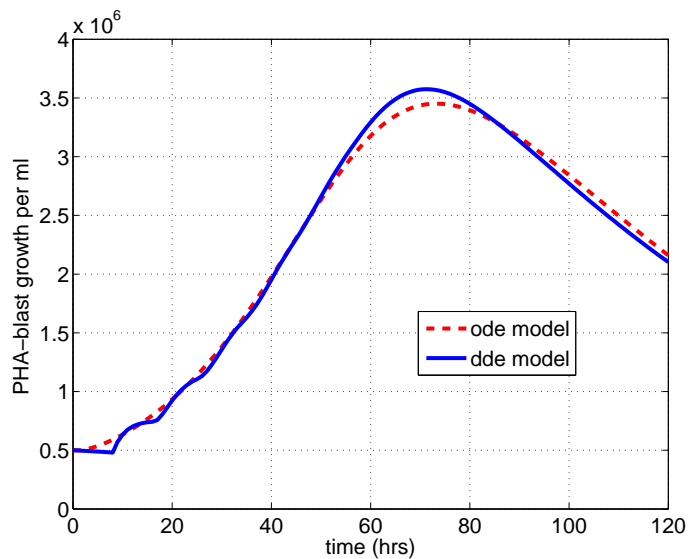


Figure 2: The Interleukin-2: model studied by Baker et al. in [?]

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:

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

The history is given by

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

The parameters are

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

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

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

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.

```
[LONGITUDINAL]
input = {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)
```

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

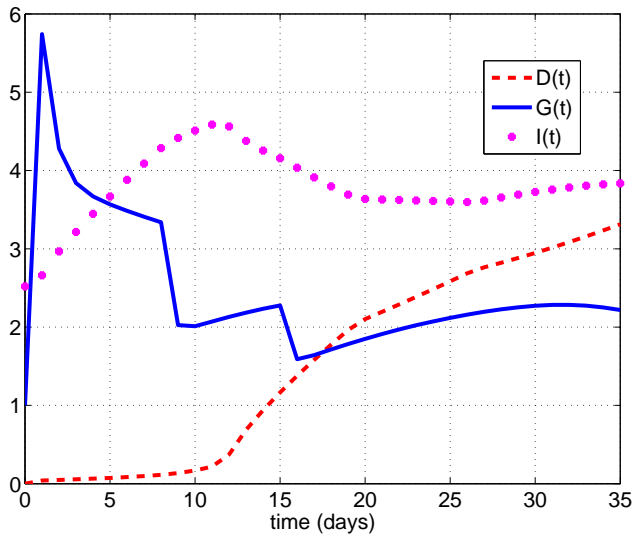


Figure 3: The PKPD model for unperturbed and perturbed arthritis development of Koch et al. [?], Experiment A with the dosing schedules: 10 mg/kg on day 1, 8, 15.

4 The section [COVARIATE]

The section [COVARIATE] defines the **distribution** of the covariates. The `input` is the list of parameters required for defining this distribution (mean, standard deviation, ...).

The section [COVARIATE] combines simple **DEFINITION:** and **EQUATION:** blocks for defining one or several random variables.

In this first example, weight is normally distributed, with mean 70 and standard deviation 10, $w \sim \mathcal{N}(70, 10)$:

```
[COVARIATE]
DEFINITION:
c = {distribution=normal, mean=70, sd=10}
```

The mean and the standard deviation can also be defined as parameters, $w \sim \mathcal{N}(w_{\text{pop}}, \omega_w)$:

```
[COVARIATE]
input={weight_pop, omega_weight}
DEFINITION:
c = {distribution=normal, mean=weight_pop, sd=omega_weight}
```

Covariates defined in the block **DEFINITION:** are continuous random variables (normal, log-normal, logit-normal and probit-normal distributions are available). Categorical covariates can be defined as discrete transformations of continuous random variables. Consider for instance a covariate that takes its values in $\{0, 1\}$, with $\mathbb{P}(c = 0) = 1 - \mathbb{P}(c = 1) = 0.4$ (example examples/MLXTran/stat4_mlxplore). Using the fact that

$$\mathbb{P}(\mathcal{N}(0, 1) \leq -0.2533) = 0.4$$

We can define $c = 0$ if $z \leq -0.2533$ and $c = 1$ otherwise, where $z \sim \mathcal{N}(0, 1)$.

```
[COVARIATE]
DEFINITION:
z = {distribution=normal, mean=0, sd=1}

EQUATION:
if z < -0.2533 ; P(z < -0.2533) = 0.4
  cz = 0
else
  cz = 1
end
c = cz
```

5 The section [INDIVIDUAL]

The section [INDIVIDUAL] defines the conditional distribution of the individual parameters. Individuals are assumed to be independent. So the syntax reads simply as the distribution of the parameters for a unique individual. It can depend on covariates. Those covariates and the population parameters are the components of the `input` of this section.

Several keywords are available for modelling the distribution of the individual parameters in the block DEFINITION:

- `distribution`: the probability distribution of the individual parameter (normal, log-normal, logit-normal and probit-normal distributions are available).
- `reference/typical`: The typical value of the individual parameter (`reference` and `typical` are synonymous).
- `prediction`: The predicted value of the individual parameter, usually defined as a possibly non linear function of covariates.
- `covariate`: A set of continuous covariates to be used in a linear model.
- `coefficient`: A set of coefficients associated to the covariates. Along with `covariate`, it defines a linear model of covariates.

There exists different possibilities for encoding the same statistical model for the individual parameters.

As an illustration, let us use the examples `stat4a_mlxplore.txt`, `stat4b_mlxplore.txt`, `stat4c_mlxplore.txt`, `stat4d_mlxplore.txt` of the demo examples available in `training material/Statistical Models`. A simple structural model $f(t) = A e^{k(t-t_0)}$ is used. It depends on 2 individual parameters A and k . Furthermore, A depends on a covariate c .

1. we consider the following mathematical representation of the statistical model:

$$A \sim \mathcal{N}(A_{\text{pop}} + \beta_A c, \omega_A)$$

$$\log(k) \sim \mathcal{N}(\log(k_{\text{pop}}) + \beta_k c, \omega_k)$$

Here, the parameters of the normal distribution are the mean and the standard-deviation. The linear relationship between the parameters and the covariates can be implemented explicitly:

stat4a_mlxlore.txt

```
[INDIVIDUAL]
input={A_pop, omega_A, k_pop, omega_k, beta_A, beta_k, c}

DEFINITION:
A = {distribution=normal, typical=A_pop, covariate=c,
     coefficient=beta_A, sd=omega_A}
k = {distribution=logNormal, typical=k_pop, covariate=c,
     coefficient=beta_k, sd=omega_k}
```

2. Other equivalent representation of the statistical model, based on the means of the distributions:

$$\mu_A = A_{\text{pop}} + \beta_A c$$

$$\mu_k = \log(k_{\text{pop}}) + \beta_k c$$

$$A \sim \mathcal{N}(\mu_A, \omega_A)$$

$$k \sim \log \mathcal{N}(\mu_k, \omega_k)$$

Here, the parameters of the log-normal distribution are the mean and the standard-deviation of the log of the r.v. (the same convention is used for any other distribution defined as a transformation of the normal distribution: logit-normal, probit-normal, ...).

stat4b_mlxlore.txt

```
EQUATION:
A_mean=A_pop + beta_A*c
logk_mean=log(k_pop) + beta_k*c

DEFINITION:
A = {distribution=normal, mean=A_mean, sd=omega_A}
k = {distribution=logNormal, mean=logk_mean, sd=omega_k}
```

3. Other equivalent representation of the statistical model, based on predictions:

$$\tilde{A} = A_{\text{pop}} + \beta_A c$$

$$\tilde{k} = k_{\text{pop}} e^{\beta_k c}$$

$$A \sim \mathcal{N}(\tilde{A}, \omega_A)$$

$$k \sim \log \mathcal{N}(\log(\tilde{k}), \omega_k)$$

This representation is useful when the prediction of the parameters are defined, but not of transformations.

stat4c_mlxlore.txt

EQUATION:

```
A_pred=A_pop + beta_A*c
k_pred=k_pop*exp(beta_k*c)
```

DEFINITION:

```
A = {distribution=normal, prediction=A_pred, sd=omega_A}
k = {distribution=logNormal, prediction=k_pred, sd=omega_k}
```

4. Other equivalent representation of the statistical model, using random effects:

$$\begin{aligned}\eta_A &\sim \mathcal{N}(0, \omega_A) \\ \eta_k &\sim \mathcal{N}(0, \omega_k) \\ A &= A_{\text{pop}} + \beta_A c + \eta_A \\ k &= k_{\text{pop}} e^{\beta_k c + \eta_k}\end{aligned}$$

stat4d_mlxlore.txt

DEFINITION:

```
eta_A = {distribution=normal, mean=0, sd=omega_A}
eta_k = {distribution=normal, mean=0, sd=omega_k}
```

EQUATION:

```
A = A_pop + beta_A*c + eta_A
k = k_pop*exp(beta_k*c + eta_k)
```

6 The section [LONGITUDINAL]

6.1 The structural model

The section [LONGITUDINAL] defines the computation of the predictions. Predictions form a deterministic function of usual and random variables. They are defined at the individual level. This is the structural part of the complete model.

No block **DEFINITION:** can be used in the section [LONGITUDINAL], whereas additional 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. The main independent variable is denoted as `t`, and is available for direct calculus of the predictions as well as computing the parameters of a `pkmodel`, for example. Time is a frequent example of main independent variable.

Usually, the `input` of this section will include the individual parameters.

7 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.

7.1 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 `C1`, `Vm` and `Km`.
- `C1`: Along with `v`, defines a linear elimination of clearance `C1`. Excludes `k`, `Vm` and `Km`.
- `Vm`: Along with `v` and `Km`, defines a Michaelis-Menten elimination of maximum elimination rate `Vm`. Excludes `k` and `C1`.
- `Km`: Along with `v` and `Vm`, defines a Michaelis-Menten elimination of Michaelis-Menten constant `Km`. Excludes `k` and `C1`.

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

- $\{Cc, Ce\} = \text{pkmodel}(\dots)$: 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.

7.2 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 `ki_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 `kj_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.

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**

Example:

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

```
EQUATION:
ddt_Ad = -ka*Ad
ddt_Ac = ka*Ad-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),
- `ka`: rate of the first order absorption (mandatory for a first-order absorption),
- `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)
```

7.3 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.

7.4 Analytical solutions

7.4.1 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 by the ODE solver, and to save computation time.

For instance the following script:

```
input = {V, k}
EQUATION:
Cc = pkmodel(V, k, k12=0.2, k21=0.3)
```

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 = {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)
```

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 = {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)
```


- 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 multiple 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 = {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)
    
```

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 = {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)
    
```

- 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 = {k}
EQUATION:
dAp_dt = -k*Ap
    
```

but its macro counterpart will generate one:

```

input = {k}
PK:
compartment(cmt=1, amount=Ap)
elimination(cmt=1, k=k)
    
```

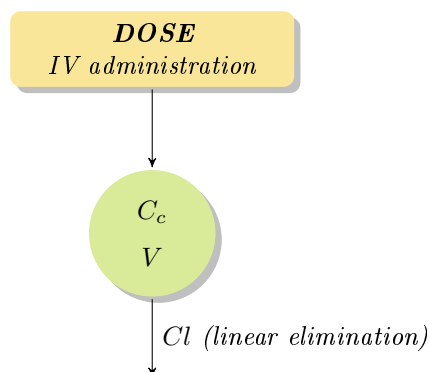
- Analytical solutions will not be generated when Michaelis Menten eliminations are used (Mtt coefficient), maximum elimination rate (Vm coefficient), transit compartments with transit rate (see Ktr coefficient) or mean transit time (see Mtt coefficient). Indeed such systems do not have an analytical 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.

7.4.2 Miscellaneous

- **How to set / unset analytical functions:** In the configuration file, inside the **[Plug-InCodeGeneration]** 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/mlxplore/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 account.
- **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.

7.5 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)

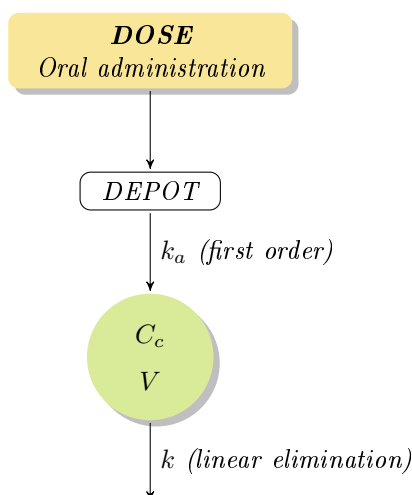
```

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.

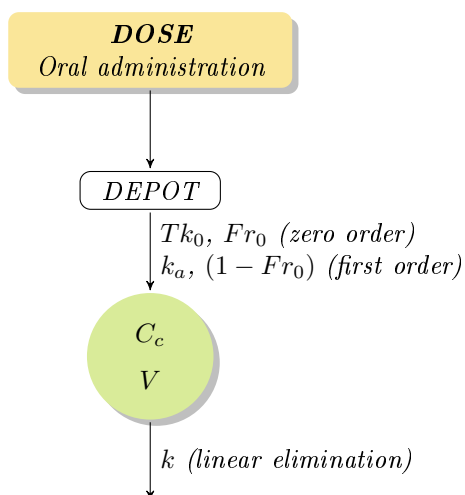


```
; PK model, oral administration,
; first order absorption, linear elimination

[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

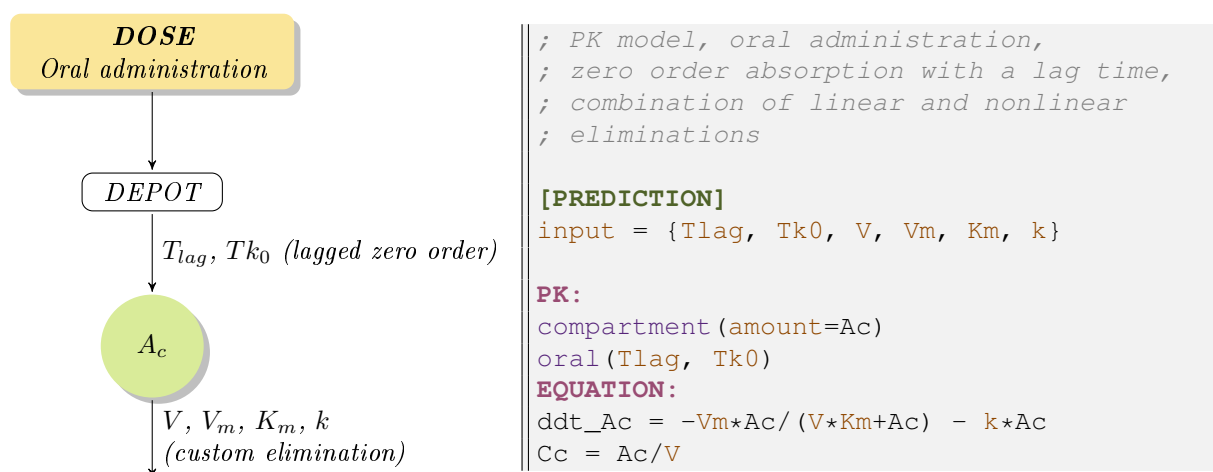
[PREDICTION]
input = {Fr0, Tk0, ka, V, k}

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

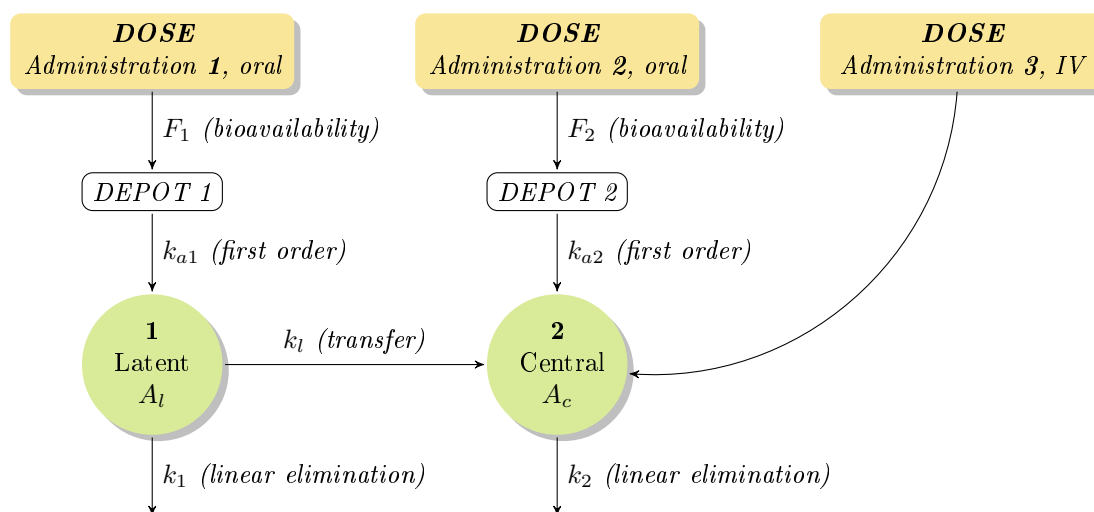
`type` triggers both of them. The artificial lag time allows the first order absorption to come after the zero order one, in a sequence. Each one concerns only a fraction of the administered amount, respectively fractions $Fr0$ and $(1 - Fr0)$.

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.



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=A1)
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.

7.6 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.

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

```
[LONGITUDINAL]
input = {A,k}

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

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.

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

```
[LONGITUDINAL]
input = {A,k}

EQUATION:
t0=5
f_0=A
ddt_f= - k*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`).

8 Miscelleneous

8.1 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.
```

8.2 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**.

8.3 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.
```

8.4 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	

References

- [1] C.T.H. Baker, G.A. Bocharov, and C.A.H. Paul. Mathematical modelling of the interleukin-2 t-cell system: A comparative study of approaches based on ordinary and delay differential equations. *Journal of Theoretical Medicine*, 1(2):117–128, 1997.
- [2] Gilbert Koch, Thomas Wagner, Christine Plater-Zyberk, Gezim Lahu, and Johannes Schropp. Multi-response model for rheumatoid arthritis based on delay differential equations in collagen-induced arthritic mice treated with an anti-gm-csf antibody. *Journal of Pharmacokinetics and Pharmacodynamics*, 39(1):55–65, 2012.
- [3] L. F. Shampine, I. Gladwell, and S. Thompson. *Solving ODEs with MATLAB*. Cambridge University Press, 2003.