



# A delay differential equation solver for MONOLIX MLXPLORE

Raphaël Kuate, Marc Lavielle, Eric Blaudez, Kaelig Chatel, Jerome Marquet,  
Jean-François Si Abdallah

► **To cite this version:**

Raphaël Kuate, Marc Lavielle, Eric Blaudez, Kaelig Chatel, Jerome Marquet, et al.. A delay differential equation solver for MONOLIX

MLXPLORE. [Research Report] RR-8489, INRIA. 2014, pp.19. <hal-00952874>

**HAL Id: hal-00952874**

**<https://hal.inria.fr/hal-00952874>**

Submitted on 5 Mar 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# A delay differential equation solver for MONOLIX & MLXPLORE

Raphaël Kuate, Marc Lavielle, Eric Blaudez, Kaelig Chatel, Jerome  
Marquet, Jean-François Si Abdallah

**RESEARCH  
REPORT**

**N° 8489**

January 2014

Project-Teams Popix





## A delay differential equation solver for MONOLIX & MLXPLORE

Raphaël Kuate\*, Marc Lavielle<sup>†</sup>, Eric Blaudez<sup>‡</sup>, Kaelig Chatel<sup>§</sup>,  
Jerome Marquet<sup>¶</sup>, Jean-François Si Abdallah<sup>||</sup>

Project-Teams Popix

Research Report n° 8489 — January 2014 — 19 pages

**Abstract:** We describe the implementation of a Delay differential Equation (DDE) solver in MONOLIX, a platform for population modeling of longitudinal data, and MLXPLORE, a tool for the exploration of complex models. We use explicit Runge-Kutta schemes. Several examples for MLXPLORE and MONOLIX are proposed.

**Key-words:** delay differential equations, solver, algorithm, Runge-Kutta formulas, modeling, pharmacokinetics, pharmacodynamics, MONOLIX, MLXPLORE

---

\* ✉: raphael.kuate@inria.fr  
† ✉: marc.lavielle@inria.fr  
‡ ✉: eric.blaudez@lixoft.net  
§ ✉: kaelig.chatel@lixoft.net  
¶ ✉: jerome.marquet@lixoft.net  
|| ✉: jean-francois.si\_abdallah@lixoft.net

**RESEARCH CENTRE  
SACLAY – ÎLE-DE-FRANCE**

1 rue Honoré d'Estienne d'Orves  
Bâtiment Alan Turing  
Campus de l'École Polytechnique  
91120 Palaiseau

## Résolution numérique d'équations différentielles avec retard dans MONOLIX & MLXPLORE

**Résumé :** Ce document décrit l'implémentation d'un solveur d'équations différentielles avec retard (DDE) pour les logiciels MONOLIX et MLXPLORE.

**Mots-clés :** équations différentielles avec retard, solveur, algorithmique, formules de Runge-Kutta, modélisation, pharmacocinétique, pharmacodynamique, MONOLIX, MLXPLORE

## 1 Introduction

MONOLIX (MOdèles NON Linéaires à effets miXtes) [2, 27, 16, 37] is a platform of reference for model-based drug development. It combines the most advanced algorithms with unique ease of use. Pharmacometricians of preclinical and clinical groups can rely on MONOLIX for population analysis and to model pharmacokinetic/pharmacodynamic (PK/PD) and other complex biochemical and physiological processes [29, 26, 39]. MONOLIX is an easy, fast and powerful tool for parameter estimation in non-linear mixed effect models, model diagnosis and assessment, and advanced graphical representation.

MLXPLORE [1] is a graphical and interactive software for the exploration and visualization of complex pharmacometric models. MLXPLORE also includes the ability to study the statistical variability of the models, and to model and study complex administrations designs. MLXPLORE is based on the Mlxtran model description language popularized by the MONOLIX software, therefore benefiting from the remarkable flexibility and power of Mlxtran to easily encode complex pharmacometric, PKPD and statistical models. MLXPLORE does not require MONOLIX, although they make for a powerful combination, enabling to use the same, human-readable model description, to finely explore the properties of the model on the one hand, and on the other hand use the same model for advanced parameter estimation in the context of population analysis and mixed effect statistics.

Modeling complex biological phenomena with ordinary differential equations (ODEs) sometimes involves a large number of variables and parameters, which makes the analysis of these models cumbersome. Modeling such systems with delay differential equations (DDEs) helps to describe the important dynamics with fewer variables and parameters [44]. Another advantage of using a DDE model over an ODE model is that the parameters in the DDE model usually have a direct biological interpretation [4]. Thus, providing MONOLIX and MLXPLORE with a DDE solver is of a major importance.

Observing that the Taylor expansion of higher order of the solution of a DDE system needs derivatives of the delayed terms, we augment the DDE system being solved with the derivatives of the delayed terms. Then we apply classical explicit Runge-Kutta schemes on the new augmented system. We have implemented two variants of timestep: an adaptive timestep and a fixed timestep of which we propose its computation with respect to the CFL condition. We also propose an algorithm for the generation of the fixed time grid such that no interpolation of delayed terms be computed.

The document is organized as follows. In the second section we describe the implementations of the algorithms. Two variants of timestep are used: a fixed timestep and an adaptive timestep. In the fixed timestep case, a version uses the delayed solution, computed simultaneously during the simulation and a second version uses interpolated delayed solution. The numerical schemes implemented in the fixed timestep case are explicit Runge-Kutta formulas of order 2, 3 and 4. In the adaptive timestep version, the numerical schemes coded are Runge-Kutta embedded pairs [33, 19, 9, 43] of type 2(1), 3(2) and 4(3), using of course interpolated delayed solution. In the third section, we give some examples of numerical tests on DDEs, solved using MLXPLORE. In the fourth section, we present examples of estimation of parameters of DDE models, computed with MONOLIX.

## 2 Algorithms

The algorithms implemented use the well-known Runge-Kutta numerical schemes [3, 10, 11, 14, 8, 43, 45, 46].

## 2.1 Runge-Kutta schemes

**ODEs** Let  $Y$  be a time-dependent vector of real unknowns and let  $\dot{Y}$  be the time derivative of  $Y$ . Let the vector  $F(t, Y)$  be a function of  $Y$  and of the time  $t$ , and let  $Y_0$  be the initial condition at the starting time  $t_0$ . The usual form of a differential equation of order 1 which can be found in PK/PD or biological [20, 34, 36, 42, 25] models writes:

$$\begin{cases} \dot{Y}(t) = F(t, Y), & t \in [t_0, T], \\ Y(t_0) = Y_0. \end{cases} \quad (2.1.1)$$

The numerical approximation of the solution of equation (2.1.1) using standard explicit Runge-Kutta numerical scheme is done as follows. Let  $t_n \in [t_0, T]$  be a value of the time grid, and denote  $Y_n$  the value of  $Y$  at time  $t_n$ . Let  $s, p \geq 1$  be two integers and let  $h_n = t_{n+1} - t_n$  be the timestep. An explicit Runge-Kutta scheme of  $s$  stages and of order  $p$  writes:

$$\begin{cases} Y_{n+1} = Y_n + h_n \sum_{i=1}^s b_i K_i, \\ K_i = F\left(t_n + c_i h_n, Y_n + \sum_{j=1}^{i-1} a_{i,j} K_j\right), & i = 2, \dots, s, \\ K_1 = F(t_n, Y_n), \end{cases} \quad (2.1.2)$$

where the coefficients  $a_{ij}, b_i, c_i \in \mathbb{R}$  are obtained by solving a system of equations [12, 13] which results of an identification of (2.1.2) with the Taylor expansion of order  $p$  of  $Y$ , with respect to (2.1.1).

**DDEs** Many developments have been made [31, 5, 28, 41, 40, 7, 15, 6] on numerical methods for delay differential equations. We use standard explicit Runge-Kutta formulas, adapted to DDEs. For sake of simplicity, we describe the derivation of an explicit Runge-Kutta numerical scheme for DDEs, in the case  $s = p = 2$ .

Let  $N$  be a positive integer and  $\tau \in \mathbb{R}_+^N$  be the vector of delays and denote  $Y_{\tau_i} = Y(t - \tau_i)$ . Let  $Y_0$  be a function of the time  $t$ , and of  $\tau_i, 1 \leq i \leq N$ . The usual form of a delayed differential equation of order 1 which can be found in PK/PD or biological [4, 30, 44, 24] models writes:

$$\begin{cases} \dot{Y}(t) = F(t, Y, Y_{\tau_1}, \dots, Y_{\tau_N}), & t \in [t_0, T], \\ Y(t) = Y_0(t, \tau), & t \leq t_0. \end{cases} \quad (2.1.3)$$

The Taylor expansion of order 2 of  $Y$  gives:

$$Y_{n+1} = Y_n + h_n \dot{Y}_n + \frac{1}{2} h_n^2 \ddot{Y}_n + \mathcal{O}(h_n^3). \quad (2.1.4)$$

Denote  $F_Y$  the matrix of the derivatives of  $F$  with respect to  $Y$ . The time derivative of  $Y$  of order 2, expressed in terms of  $F$  with respect to (2.1.3), writes:

$$\ddot{Y} = \dot{F} + F_Y F + \sum_{i=1}^N F_{Y_{\tau_i}} \dot{Y}_{\tau_i}. \quad (2.1.5)$$

If one does not neglect  $\dot{Y}_{\tau_i}$ , it appears from (2.1.5) that one needs  $\dot{Y}_{\tau_i}$  in order to identify the coefficients of an explicit Runge-Kutta formula for the numerical approximation of (2.1.3). Thus, system (2.1.3) is augmented as follows:

$$\begin{cases} \dot{Y}(t) = F(t, Y, Y_{\tau_1}, \dots, Y_{\tau_N}), & t \in [t_0, T], \\ \dot{Y}_{\tau_i}(t) = F(t - \tau_i, Y_{\tau_i}, Y_{\tau_1 + \tau_i}, \dots, Y_{\tau_N + \tau_i}), & t \in [t_0, T], \quad 1 \leq i \leq N, \\ Y(t) = Y_0(t, \tau), & t \leq t_0. \end{cases} \quad (2.1.6)$$

Define  $Z := \{Y, Y_{\tau_1}, \dots, Y_{\tau_N}\}$ , set  $Z_0(t) = Y_0(t, \tau)$ ,  $t \leq t_0$ , and define

$$F(t, Z) := \left\{ \begin{array}{c} F(t, Y, Y_{\tau_1}, \dots, Y_{\tau_N}) \\ F(t - \tau_1, Y_{\tau_1}, Y_{\tau_1+\tau_1}, \dots, Y_{\tau_N+\tau_1}) \\ F(t - \tau_2, Y_{\tau_2}, Y_{\tau_1+\tau_2}, \dots, Y_{\tau_N+\tau_2}) \\ \vdots \\ F(t - \tau_N, Y_{\tau_N}, Y_{\tau_1+\tau_N}, \dots, Y_{\tau_N+\tau_N}) \end{array} \right\}.$$

One has:

$$\begin{cases} \dot{Z}(t) = F(t, Z), & t \in [t_0, T], \\ Z(t) = Z_0(t), & t \leq t_0. \end{cases} \quad (2.1.7)$$

System (2.1.7) is used instead of (2.1.3) for the computations with standard Runge-Kutta formulas. Thus, an explicit Runge-Kutta scheme of  $s$  stages and of order  $p$  for the numerical approximation of (2.1.3) writes:

$$\begin{cases} Z_{n+1} = Z_n + h_n \sum_{i=1}^s b_i K_i, \\ K_i = F\left(t_n + c_i h_n, Z_n + \sum_{j=1}^{i-1} a_{i,j} K_j\right), \quad i = 2, \dots, s, \\ K_1 = F(t_n, Z_n), \end{cases} \quad (2.1.8)$$

where the coefficients  $a_{ij}, b_i, c_i \in \mathbb{R}$  are obtained by solving a system of equations which results of an identification of (2.1.8) with the Taylor expansion of order  $p$  of  $Z$ , with respect to (2.1.7).

## 2.2 Solver features

### 2.2.1 Fixed timestep

The solver has a feature which uses a fixed timestep, with two sub-features. The first sub-feature uses cubic Hermite interpolation for the computation of delayed terms, since the solution and its time first order derivative are computed simultaneously. The second sub-features does not use interpolation. Let  $\Omega$  be the discretization of the time interval.  $\Omega = \{t_k, t_k \in [t_0, T]\}$  is computed such that:

$$\forall t_n \in \Omega, t_n - \tau_i \geq t_0 \implies t_n - \tau_i \in \Omega, \quad 1 \leq i \leq N. \quad (2.2.1)$$

The table  $T_\Omega$  containing elements of  $\Omega$  is returned with a table of numbers of the entries of  $T_\Omega$  which contains  $t_n - \tau_i, \forall t_n \in \Omega, 1 \leq i \leq N$ . Thus for any  $Y_n = Y(t_n)$  computed, one knows the delayed terms  $Y(t_n - \tau_i)$  already computed, since statement (2.2.1) is always satisfied.

In the second sub-feature where the time grid contains delayed terms, the computation of  $T_\Omega$  is done using Algorithm 1, for a given a timestep  $\Delta_t$ .

**Computation of the timestep** In numerical simulation of time-dependent equations, numerical instabilities can occur, particularly for explicit numerical schemes [35, 21, 38, 47, 3, 32, 23]. Since the timestep  $\Delta_t$  is fixed, one must compute  $\Delta_t$  with respect to the Courant-Friedrichs-Lewy [17, 18] condition. Let  $\#z$  be the size of  $Z$  and let  $I_{\#z}$  be the identity matrix of the size of  $Z$ . The first order time derivative of  $Z$  is approached from (2.1.7) with the following:

$$\dot{Z} \approx F_Z Z,$$

of which an explicit numerical discretization writes:

$$Z_{n+1} \approx \left[ I_{\#z} + \Delta_{t_n} \{F_Z\}_n \right] Z_n.$$



**Algorithm 1** Time grid computation.

---

```

1:  $T_\Omega$  inserts  $t_0$ ,  $t_1 \leftarrow T$ , inserts  $t_1$  into  $T_\Omega$  in descending order.
2: while the biggest stepsize between  $t_1$  and  $t_0$  is greater than  $\Delta_t$  do
3:    $t_j \leftarrow t_1$ ,  $t_k \leftarrow t_1$ 
4:   while  $t_j \geq t_0$  do
5:     for all  $i, 1 \leq i \leq N$  do
6:        $t_i \leftarrow t_j - \tau_i$ 
7:       if  $t_i \geq t_0$  then
8:         put a flag on  $t_i$ 
9:         inserts  $t_i$  into  $T_\Omega$  in descending order.
10:      end if
11:    end for
12:     $t_j \leftarrow$  the next entry of  $T_\Omega$  with a flag, after  $t_k$  in descending order
13:    in  $T_\Omega$ , remove the flag on  $t_j$ 
14:     $t_k \leftarrow t_j$ 
15:  end while
16:   $t_1 \leftarrow t_1 - \Delta_t$ 
17:  if  $t_1 \geq t_0$  then
18:    inserts  $t_1$  into  $T_\Omega$  in descending order.
19:  end if
20: end while
21: rearrange  $T_\Omega$  in ascending order

```

---

The timestep  $\Delta_t$  is then computed such that (2.2.2) be satisfied.

$$\left\| \left[ I_{\#z} + \Delta_t F_Z \right] Z \right\| = CFL < 1. \quad (2.2.2)$$

Let  $z_i, 1 \leq i \leq \#z$ , be the  $i$ -th entry of  $Z$ . The timestep  $\Delta_t$  is obtained from (2.2.2) with

$$z_i = \frac{1}{\sqrt{\#z}}, \quad 1 \leq i \leq \#z, \quad \text{i.e., } \|Z\| = 1.$$

Let  $Z^i, 1 \leq i \leq \#z$ , be the vector of the size  $\#z$  containing  $z_i$  at the  $i$ -th entry and zeros elsewhere. Since  $F_Z$  may be time dependent, let  $F_Z^k$  be its value for  $t = t_k$ . The  $i$ -th column of  $F_Z^k$  is obtained by evaluating the following expression

$$F_Z^{i,k} = \frac{F(t_k, Z + \alpha Z^i) - F(t_k, Z)}{\alpha}, \quad 1 \leq i \leq \#z, \quad 0 < \alpha \ll 1.$$

For each  $t_k$  chosen, one obtains a value  $\Delta_{t_k}$  of  $\Delta_t$  from (2.2.2). The time interval  $[t_0, T]$  is uniformly meshed and

$$t_k = t_0 + \frac{k}{\#nk} (T - t_0), \quad 0 \leq k \leq \#nk, \quad \#nk < 20.$$

The timestep  $\Delta_t$  is taken as the minimum of the  $\Delta_{t_k}$ 's. However, one checks if  $F_Z$  is not time dependent by comparing  $F(t_0, Z)$  with  $F(t_1, Z)$ , and the computations are stopped if  $F(t_0, Z) \equiv F(t_1, Z)$ .

**Coefficients of the Runge-Kutta formulas** In the fixed timestep feature of the solver, the numerical schemes implemented are explicit Runge-Kutta formulas of order 2, 3 and 4:

$$\begin{cases} Z_{n+1} = Z_n + h_n \sum_{i=1}^s b_i K_i, \\ K_i = F\left(t_n + c_i h_n, Z_n + \sum_{j=1}^{i-1} a_{i,j} K_j\right), \quad i = 2, \dots, s, \\ K_1 = F(t_n, Z_n). \end{cases}$$

The coefficients  $a_{i,j}$ ,  $b_i$  and  $c_i$  can be stored in a table called Butcher tableau defined in table 1.

$c_2$	$a_{2,1}$				
$\vdots$	$\vdots$	$\ddots$			
$c_{s-1}$	$a_{s-1,1}$	$a_{s-1,2}$	$\dots$		
$c_s$	$a_{s,1}$	$a_{s,2}$	$\dots$	$a_{s,s-1}$	
	$b_1$	$b_2$	$\dots$	$b_{s-1}$	$b_s$

Table 1: The Butcher tableau of the Runge-Kutta coefficients.

The coefficients used are given in the Butcher tableaux, table 2, and can be found in [12, 11, 10].

Order 2:

$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$
	$\frac{1}{2}$ $\frac{1}{2}$

Order 3:

$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	-1   2
	$\frac{1}{6}$ $\frac{2}{3}$ $\frac{1}{6}$

Order 4:

$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	0 $\frac{1}{2}$
1	0   0   1
	$\frac{1}{6}$ $\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{6}$

Table 2: Some Butcher tableaux of Runge-Kutta explicit scheme of order 2, 3 and 4.

### 2.2.2 Adaptive timestep

The model considered is the augmented system (2.1.7). In an adaptive timestep method, the timestep is computed such that the method of order  $p$  be more accurate than an embedded method of order  $p' < p$ . Generally, one choses  $p' = p - 1$ . In a step  $n$ , the local error  $E_n = \|Z_n - Z'_n\|$  between the approximation  $Z$  of order  $p$  and the approximation  $Z'$  of order  $p'$  is used to validate or not the step and to compute the timestep for the next step.

$$Z_{n+1} = Z_n + h_n \sum_{i=1}^s b_i K_i, \quad Z'_{n+1} = Z'_n + h_n \sum_{i=1}^s b'_i K_i.$$

The coefficients of the Runge-Kutta formula for these two approximations are computed such that the coefficients  $a_{i,j}$ , and  $c_i$  (2.1.8) of the two approximations be the same. Some embedded

pairs also use the FSAL (first same as the last) strategy *i.e.*, when a step is accepted, the last coefficient  $K_s$  becomes the first coefficient  $K_1$  of the next step, which reduces the method to a  $s-1$  stage method. Many strategies [14, 41, 43, 22, 45, 46] have been developed for the computation of adaptive timesteps. The strategy used is a standard formula described in [14, 45]. Given a tolerance  $TOL$  the algorithm computes the next timestep  $h_{n+1}$  with the following formula:

$$h_{n+1} = \theta h_n \left( \frac{TOL}{E_n} \right)^{\frac{1}{p}}, \quad \theta \in (0, 1).$$

The next stepsize  $h_{n+1}$  is accepted if  $TOL \geq E_n$ . If  $TOL < E_n$  the current step  $n$  is rejected and recomputed again with  $h_n \leftarrow h_{n+1}$ .

The first timestep  $h_1$  is computed [14] as

$$h_1 = \left( \frac{TOL}{\max \{ \|F(t_0, Z_0)\|, 10^{-p} \}} \right)^{\frac{1}{p}}.$$

The Butcher tableau for Runge-Kutta embedded pairs is defined in table 3.

$c_2$	$a_{2,1}$				
$\vdots$	$\vdots$	$\ddots$			
$c_{s-1}$	$a_{s-1,1}$	$a_{s-1,2}$	$\dots$		
$c_s$	$a_{s,1}$	$a_{s,2}$	$\dots$	$a_{s,s-1}$	
	$b_1$	$b_2$	$\dots$	$b_{s-1}$	0
	$b'_1$	$b'_2$	$\dots$	$b'_{s-1}$	$b'_s$

Table 3: The Butcher tableau of coefficients of the Runge-Kutta embedded pairs.

**Runge-Kutta coefficients** In the adaptive timestep feature of the solver, the numerical schemes implemented are Runge-Kutta embedded FSAL pairs of M. Sofroniou and G. Spaletta [43], of type 2(1), 3(2) and 4(3). The Butcher tableaux are given in tables 4 and 5.

Type 2(1):

1	1		
1	$\frac{1}{2}$	$\frac{1}{2}$	
	$\frac{1}{2}$	$\frac{1}{2}$	0
	1	$-\frac{1}{6}$	$\frac{1}{6}$

Type 3(2):

$\frac{1}{2}$	$\frac{1}{2}$			
1	-1	2		
1	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$	
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$	0
	$\frac{22 - \sqrt{82}}{72}$	$\frac{14 + \sqrt{82}}{36}$	$\frac{-4 + \sqrt{82}}{144}$	$\frac{16 - \sqrt{82}}{48}$

Table 4: The Butcher tableaux of Runge-Kutta FSAL pairs of M. Sofroniou and G. Spaletta [43], of types 2(1) and 3(2).

Type 4(3):

$\frac{2}{5}$	$\frac{2}{5}$			
$\frac{3}{5}$	$-\frac{3}{20}$	$\frac{2}{4}$		
1	$\frac{19}{44}$	$-\frac{15}{44}$	$\frac{10}{11}$	
1	$\frac{11}{72}$	$\frac{25}{72}$	$\frac{25}{72}$	$\frac{11}{72}$
0	$\frac{11}{72}$	$\frac{25}{72}$	$\frac{25}{72}$	$\frac{11}{72}$
	$\frac{1251515}{8970912}$	$\frac{3710105}{8970912}$	$\frac{2519695}{8970912}$	$\frac{61105}{8970912}$
				$\frac{119041}{757576}$

Table 5: The Butcher tableau of Runge-Kutta FSAL pairs of M. Sofroniou and G. Spaletta [43], of type 4(3).

### 3 Simulation of DDEs using MLXPLORE

#### Example 3.1

The model solved is the SEIR epidemic model of Genik & van den Driessche, example 4.4.1 of

[40] defined by:

$$\begin{cases} N(t) = S(t) + E(t) + I(t) + R(t) \\ \dot{S}(t) = A(t) - S(t) - \lambda \frac{S(t)I(t)}{N(t)} + \gamma I(t - \tau) \exp(-d\tau), \\ \dot{E}(t) = \lambda \frac{S(t)I(t)}{N(t)} - dE(t) - \lambda \frac{S(t - \omega)I(t - \omega) \exp(-d\omega)}{N(t - \omega)}, \\ \dot{I}(t) = -(\gamma + \epsilon + d)I(t) + \lambda \frac{S(t - \omega)I(t - \omega) \exp(-d\omega)}{N(t - \omega)}, \\ \dot{R}(t) = \gamma I(t) - dR(t) - \gamma I(t - \tau) \exp(-d\tau), \end{cases} \quad t \in [0, 350]. \quad (3.1.1)$$

The history is given by

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

The parameters are

$$A = 0.33, d = 0.006, \lambda = 0.308, \gamma = 0.04, \epsilon = 0.06, \tau = 42, \omega = 0.15. \quad (3.1.2)$$

The MLXPLORE script of this example is given in the section below

#### MLXPLORE script

```
<MODEL>
[PREDICTION]
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)

<PARAMETER>
A = 0.33
d = 0.006
lambda = 0.308
gamma = 0.04
epsilon = 0.06
tau = 42
omega = 0.15
<OUTPUT>
list = {S,E,I,R}
grid = 0:1:350
```

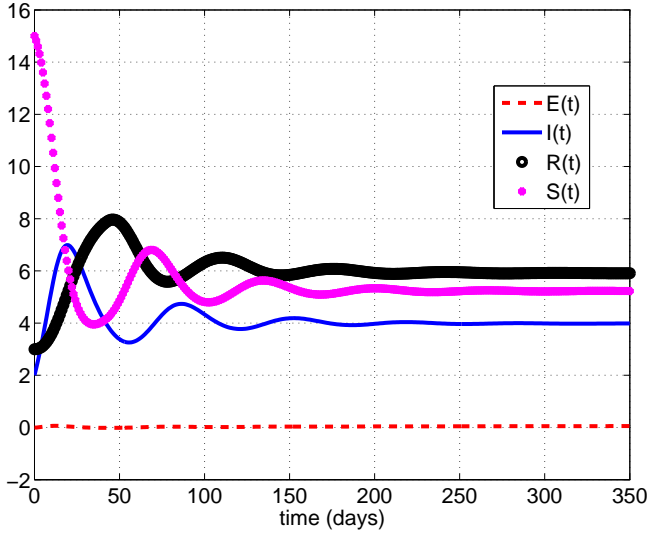


Figure 1: The SEIR epidemic model of Genik & van den Driessche, example 4.4.1 of [40].

### Example 3.2

The model solved is the Interleukin-2: model studied by Baker *et al.* in [4] and defined by:

$$\left\{ \begin{array}{l} \dot{I}_2(t) = -\alpha_{I_2} I_2(t) - n_{I_2} b_{TI_2} \frac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t), \\ \dot{T}_A(t) = \rho b_{TI_2} \frac{I_2(t - \tau_D)}{I_2(t - \tau_D)/I_2^* + 1} T_A(t - \tau_D) \\ \quad - b_{TI_2} \frac{I_2(t - \tau_S)}{I_2(t - \tau_S)/I_2^* + 1} T_A(t - \tau_S) - \alpha_{AR} T_A(t), \\ \dot{T}_D(t) = b_{TI_2} \frac{I_2(t - \tau_S)}{I_2(t - \tau_S)/I_2^* + 1} T_A(t - \tau_S) \\ \quad - b_{TI_2} \frac{I_2(t - \tau_D)}{I_2(t - \tau_D)/I_2^* + 1} T_A(t - \tau_D), \\ \dot{T}_R(t) = \alpha_{AR} T_A(t) - \alpha_R T_R(t), \end{array} \right. \quad t \in [0, 100]. \quad (3.2.1)$$

The parameters of model (3.2.1) are

$$\begin{aligned} \alpha_{I_2} = 0, \quad \alpha_R = 1.5 \cdot 10^{-2}, \quad \alpha_{AR} = 6.6 \cdot 10^{-2}, \quad n_{I_2} = 4755, \\ b_{TI_2} = 1.8 \cdot 10^{-11}, \quad I_2^* = 6 \cdot 10^{10}, \quad \tau_D = 6.6, \quad \tau_S = 10, \quad \rho = 2. \end{aligned} \quad (3.2.2)$$

The model (3.2.1) is compared in [4] with an ODE model of the same phenomenon, defined as follows:

$$\left\{ \begin{array}{l} \dot{I}_2(t) = -\alpha_{I_2} I_2(t) - n_{I_2} b_{TI_2} \frac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t), \\ \dot{T}_A(t) = \rho b_D T_D(t) - b_{TI_2} \frac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t) - \alpha_{AR} T_A(t), \\ \dot{T}_D(t) = b_{TI_2} \frac{I_2(t)}{I_2(t)/I_2^* + 1} T_A(t) - b_D T_D(t), \\ \dot{T}_R(t) = \alpha_{AR} T_A(t) - \alpha_R T_R(t), \end{array} \right. \quad t \in [0, 100]. \quad (3.2.3)$$

The parameters of model (3.2.3) 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 (3.2.1) and (3.2.3) is given by

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

The MLXPLORE script of the dde model of this example is given in the section below

### MLXPLORE script

<MODEL>

[PREDICTION]

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

<PARAMETER>

alphaI2 = 0

alphaR = 1.5e-2

alphaAR = 6.6e-2

nI2 = 4755

bTI2 = 1.8e-11

starI2 = 6e10

tauD = 6.6

tauS = 10

rho = 2

<OUTPUT>

list = {TV}

grid = 0:1:100

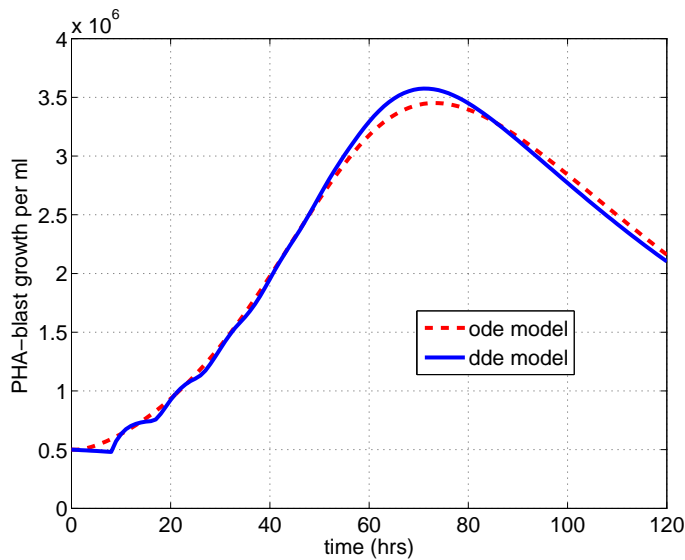


Figure 2: The Interleukin-2: model studied by Baker *et al.* in [4]

### Example 3.3

The model solved is the PKPD model for unperturbed and perturbed arthritis development of Koch *et al.* [24] 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 [24] is

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

However,  $c(t)$  can be fitted with MLXPLORE, knowing the volumes  $V_1, V_2$  of the compartments, the parameters  $\alpha, \beta, CL$ , the *dose* (amount in Mlxtran) and the administration times of the PK model. The MLXPLORE script of this example is given in section the below

#### MLXPLORE script

<MODEL>

[PREDICTION]



```
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)
```

**<DESIGN>**

**[ADMINISTRATION]**

```
adm1 = {time = 1, amount = 10}
```

```
adm2 = {time = 8, amount = 10}
```

```
adm3 = {time = 15, amount = 10}
```

**[TREATMENT]**

```
trt1 = {adm1, adm2, adm3}
```

**<PARAMETER>**

```
CL = 0.0029
```

```
V1 = 0.0265
```

```
V2 = 0.0270
```

```
alpha = 0.2256
```

```
beta = 0.0065
```

```
a = 1
```

```
b = 0.5
```

```
sigma1 = 0.154
```

```
sigma2 = 0.065
```

```
sigma3 = 0.003
```

```
k1 = 0.183
```

```
k2 = 0.092
```

```
k3 = 5
```

```
k4 = 0.064
```

```
k5 = 0.016
```

```
tau = 11.2
```

**<OUTPUT>**

```
list = {G,I,D}
```

```
grid = 0:1:35
```

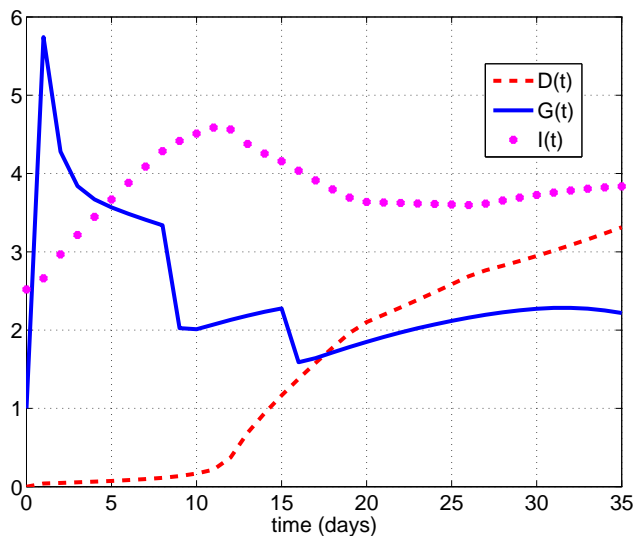


Figure 3: The PKPD model for unperturbed and perturbed arthritis development of Koch *et al.* [24], Experiment A with the dosing schedules: 10 mg/kg on day 1, 8, 15.

## 4 Estimation of parameters of DDEs using MONOLIX

### 4.1 Example 3.1

The data used for the estimations of parameters  $A, \lambda, \gamma, \epsilon, \tau, d$  of model (3.1.1) are simulated with MLXPLORE, using the model with values of the parameters given in (3.1.2), of which a random noise has been added to the parameter  $d$  with a variance  $\omega_d = 1.2$ .

A similar example can be found in the MONOLIX demos folder at  
 lixoft/monolix/monolix430/demos/dde/seir\_projct.mlxtran

```
*****
*      ex4p4p1dde-2_projct.mlxtran
*      January 14, 2014 at 15:45:09
*      Monolix version: 4.3.0
*****
Estimation of the population parameters
```

	parameter
A	: 0.332
lambda	: 0.309
gamma	: 0.0399
epsilon	: 0.0602
d	: 0.0068
tau	: 42.1
omega_A	: 0.0118
omega_lambda	: 0.00428
omega_gamma	: 0.013
omega_epsilon	: 0.0267

```

omega_d      :    0.811
omega_tau    :    0.00235

```

## 4.2 Example 3.2

The data used for the estimations of parameters  $\tau_D, \tau_S$  of model (3.2.1) are simulated with MLXPLORE, using the model with values of the parameters given in (3.2.2).

A similar example can be found in the MONOLIX demos folder at  
 lixoft/monolix/monolix430/demos/dde/arthritis\_projct.mlxtran

```

*****
*      I12_projct.mlxtran
*      January 14, 2014 at 09:02:42
*      Monolix version: 4.3.0
*****
Estimation of the population parameters

          parameter
tauD      :          6.73
tauS      :          9.65

omega_tauD : 0.000115
omega_tauS : 0.000237

```

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Algorithms</b>	<b>3</b>
2.1	Runge-Kutta schemes . . . . .	4
2.2	Solver features . . . . .	5
2.2.1	Fixed timestep . . . . .	5
2.2.2	Adaptive timestep . . . . .	7
<b>3</b>	<b>Simulation of DDEs using MLXPLORE</b>	<b>9</b>
3.1	Example . . . . .	9
3.2	Example . . . . .	11
3.3	Example . . . . .	13
<b>4</b>	<b>Estimation of parameters of DDEs using MONOLIX</b>	<b>15</b>
4.1	Example 3.1 . . . . .	15
4.2	Example 3.2 . . . . .	16

## References

- [1] MLXPLORE *A software for model exploration and visualization.*  
<http://www.lixoft.eu/products/mlxplore/mlxplore-documentation/>.
- [2] MONOLIX *A software for the analysis of nonlinear mixed effects models.*  
<http://www.lixoft.eu/products/monolix/documentation/>.

- 
- [3] Christopher TH Baker and Christopher AH Paul. Computing stability regions-Runge-Kutta methods for delay differential equations. *IMA Journal of Numerical Analysis*, 14(3):347–362, 1994.
- [4] 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.
- [5] C.T.H. Baker, C.A.H. Paul, and D.R. Willé. Issues in the numerical solution of evolutionary delay differential equations. *Advances in Computational Mathematics*, 3(1):171–196, 1995.
- [6] Robert J. Bauer, Gary Mo, and Wojciech Krzyzanski. Solving delay differential equations in S-ADAPT by method of steps. *Computer Methods and Programs in Biomedicine*, 111(3):715 – 734, 2013.
- [7] Alfredo Bellen and Marino Zennaro. *Numerical methods for delay differential equations*. Oxford University Press, 2003.
- [8] G. Vanden Berghe, H. De Meyer, M. Van Daele, and T. Van Hecke. Exponentially-fitted explicit Runge-Kutta methods. *Computer Physics Communications*, 123(1-3):7 – 15, 1999.
- [9] P. Bogacki and L.F. Shampine. A 3(2) pair of Runge-Kutta formulas. *Applied Mathematics Letters*, 2(4):321 – 325, 1989.
- [10] J. C. Butcher. A history of Runge-Kutta methods. *Appl. Numer. Math.*, 20(3):247–260, March 1996.
- [11] J. C. Butcher and G. Wanner. Runge-Kutta methods: Some historical notes. *Appl. Numer. Math.*, 22(1-3):113–151, November 1996.
- [12] John C Butcher. Coefficients for the study of Runge-Kutta integration processes. *J. Austral. Math. Soc.*, 3:185–201, 1963.
- [13] John C Butcher. On the attainable order of Runge-Kutta methods. *Math. Comp*, 19(408-417):2, 1965.
- [14] M. Calvo, D.J. Higham, J.I. Montijano, and L. Ráindez. Stepsize selection for tolerance proportionality in explicit Runge-Kutta codes. *Advances in Computational Mathematics*, 7(3):361–382, 1997.
- [15] Rodrigo Castro, Ernesto Kofman, and François E. Cellier. Quantization-based integration methods for delay-differential equations. *Simulation Modelling Practice and Theory*, 19(1):314 – 336, 2011. Modeling and Performance Analysis of Networking and Collaborative Systems.
- [16] PhylindaL.S. Chan, Philippe Jacqmin, Marc Lavielle, Lynn McFadyen, and Barry Weatherley. The use of the SAEM algorithm in MONOLIX software for estimation of population pharmacokinetic-pharmacodynamic-viral dynamics parameters of maraviroc in asymptomatic HIV subjects. *Journal of Pharmacokinetics and Pharmacodynamics*, 38(1):41–61, 2011.
- [17] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzgleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.

- [18] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM J. Res. Dev.*, 11(2):215–234, March 1967.
- [19] J.R. Dormand and P.J. Prince. Runge-Kutta triples. *Computers & Mathematics with Applications*, 12(9, Part A):1007 – 1017, 1986.
- [20] Johan L. Gabrielsson and Daniel L. Weiner. Methodology for pharmacokinetic/pharmacodynamic data analysis. *Pharmaceutical Science & Technology Today*, 2(6):244 – 252, 1999.
- [21] Gopal K. Gupta, Ron Sacks-Davis, and Peter E. Tescher. A review of recent developments in solving ODEs. *ACM Comput. Surv.*, 17(1):5–47, March 1985.
- [22] Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, 31(3):363–396, September 2005.
- [23] Z. Kalogiratou, Th. Monovasilis, G. Psihoyios, and T.E. Simos. Runge-Kutta type methods with special properties for the numerical integration of ordinary differential equations. *Physics Reports*, (0), 2013.
- [24] 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.
- [25] Wojciech Krzyzanski and JuanJose Perez Ruixo. Lifespan based indirect response models. *Journal of Pharmacokinetics and Pharmacodynamics*, 39(1):109–123, 2012.
- [26] Marc Lavielle and Cyprien Mbogning. An improved SAEM algorithm for maximum likelihood estimation in mixtures of non linear mixed effects models. *Statistics and Computing*, pages 1–15, 2013.
- [27] Marc Lavielle and France Mentré. Estimation of population pharmacokinetic parameters of saquinavir in HIV patients with the MONOLIX software. *Journal of Pharmacokinetics and Pharmacodynamics*, 34(2):229–249, 2007.
- [28] Tatyana Luzyanina, Koen Engelborghs, Kurt Lust, and Dirk Roose. Computation, continuation and bifurcation analysis of periodic solutions of delay differential equations. *International Journal of Bifurcation and Chaos*, 07(11):2547–2560, 1997.
- [29] David Makowski and Marc Lavielle. Using SAEM to estimate parameters of models of response to applied fertilizer. *Journal of Agricultural, Biological, and Environmental Statistics*, 11(1):45–60, 2006.
- [30] Alexander Y Mitrophanov, Gordon Churchward, and Mark Borodovsky. Control of streptococcus pyogenes virulence: Modeling of the CovR/S signal transduction system. *Journal of theoretical biology*, 246(1):113–128, 2007.
- [31] Christopher A.H. Paul. Developing a delay differential equation solver. *Applied Numerical Mathematics*, 9(3-5):403 – 414, 1992.
- [32] Linda R. Petzold, Laurent O. Jay, and Jeng Yen. Numerical solution of highly oscillatory ordinary differential equations. *Acta Numerica*, 6:437–483, 1 1997.

- 
- [33] PJ Prince and JR Dormand. High order embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 7(1):67–75, 1981.
- [34] José Pérez-Urizar, Vinicio Granados-Soto, Francisco J Flores-Murrieta, and Gilberto Castañeda-Hernández. Pharmacokinetic-pharmacodynamic modeling: Why? *Archives of Medical Research*, 31(6):539 – 545, 2000.
- [35] E.H Rogers. Stability and convergence of approximation schemes. *Journal of Mathematical Analysis and Applications*, 20(3):442 – 453, 1967.
- [36] Sara E Rosenbaum. *Basic pharmacokinetics and pharmacodynamics: An integrated textbook and computer simulations*. John Wiley & Sons, 2011.
- [37] S Ruiz, JM Conil, B Georges, F Ravat, T Seguin, P Letocart, O Fourcade, and S Saivin. Cef-tazidime dosage regimen recommendations in burn patients based on a monolix population pharmacokinetic study. *Critical Care*, 16(1):1–189, 2012.
- [38] J.M. Sanz-Serna and M.N. Spijker. Regions of stability, equivalence theorems and the Courant-Friedrichs-Lewy condition. *Numerische Mathematik*, 49(2-3):319–329, 1986.
- [39] Radojka M. Savic, France Mentré, and Marc Lavielle. Implementation and evaluation of the SAEM algorithm for longitudinal ordered categorical data with an illustration in pharmacokinetics-pharmacodynamics. *The AAPS Journal*, 13(1):44–53, 2011.
- [40] L. F. Shampine, I. Gladwell, and S. Thompson. *Solving ODEs with MATLAB*. Cambridge University Press, 2003.
- [41] L.F. Shampine and S. Thompson. Solving DDEs in matlab. *Applied Numerical Mathematics*, 37(4):441 – 458, 2001.
- [42] A. M. Smith, J. A. McCullers, and F. R. Adler. Mathematical model of a three-stage innate immune response to a pneumococcal lung infection. *Journal of Theoretical Biology.*, 276:106–116, 2011.
- [43] M. Sofroniou and G. Spaletta. Construction of explicit Runge-Kutta pairs with stiffness detection. *Mathematical and Computer Modelling*, 40(11-12):1157 – 1169, 2004.
- [44] J Srividhya, MS Gopinathan, and Santiago Schnell. The effects of time delays in a phosphorylation–dephosphorylation pathway. *Biophysical chemistry*, 125(2):286–297, 2007.
- [45] Ch. Tsitouras. Runge-Kutta interpolants for high precision computations. *Numerical Algorithms*, 44(3):291–307, 2007.
- [46] Ch. Tsitouras, I.Th. Famelis, and T.E. Simos. On modified Runge-Kutta trees and methods. *Computers & Mathematics with Applications*, 62(4):2101 – 2111, 2011.
- [47] P. van der Houwen and B. Sommeijer. Explicit Runge-Kutta (-Nyström) methods with reduced phase errors for computing oscillating solutions. *SIAM Journal on Numerical Analysis*, 24(3):595–617, 1987.



**RESEARCH CENTRE  
SACLAY – ÎLE-DE-FRANCE**

1 rue Honoré d'Estienne d'Orves  
Bâtiment Alan Turing  
Campus de l'École Polytechnique  
91120 Palaiseau

Publisher  
Inria  
Domaine de Voluceau - Rocquencourt  
BP 105 - 78153 Le Chesnay Cedex  
[inria.fr](http://inria.fr)

ISSN 0249-6399