APPLYING STIFF INTEGRATORS FOR ODES AND DDES TO PROBLEMS WITH DISTRIBUTED DELAYS

NICOLA GUGLIELMI[*] AND ERNST HAIRER[†]

Abstract. There exist excellent codes for an efficient numerical treatment of stiff and differential-algebraic problems. Let us mention Radau5 which is based on the 3-stage Radau IIA collocation method, and its extension to problems with discrete delays Radar5. The aim of the present work is to present a technique that permits a direct application of these codes to problems having a right-hand side with an additional distributed delay term (which is a special case of an integro-differential equation). Models with distributed delays are of increasing importance in pharmacodynamics and pharmacokinetics for the study of the interaction between drugs and the body.

The main idea is to approximate the distribution kernel of the integral term by a sum of exponential functions or by a quasi-polynomial expansion, and then to transform the distributed (integral) delay term into a set of ordinary differential equations. This set is typically stiff and, for some distribution kernels (e.g., Pareto distribution), it contains discrete delay terms with constant delay. The original equations augmented by this set of ordinary differential equations can have a very large dimension, and a careful treatment of the solution of the arising linear systems is necessary.

The use of the codes Radau5 and Radar5 is illustrated at three examples (two test equations and one problem taken from pharmacodynamics). The driver programs for these examples are publicly available from the homepages of the authors.

Key words. Stiff systems, differential-algebraic equations, delay equations, integro-differential equations, distributed delays, Runge-Kutta methods, approximation by sum of exponentials, gamma distribution, Pareto distribution, Radau5, Radar5.

AMS subject classifications. 65L06, 45D05, 65F05

1. Introduction.

The motivation of the present work is in dealing numerically with mathematical models consisting of delay integro-differential equations, as those arising in pharmacokinetics and pharmacodynamics, where the interaction between drugs and the body is studied (e.g., [16, 11, 5]). The problem is usually modeled by ordinary or delay, differential or differential-algebraic equations, which typically contain integral terms of convolution type (also termed distributed delay). Differential equations with a distributed delay term are a special case of Volterra integro-differential equations. There is a large literature on the numerical discretization of such problems, and their accuracy and stability is well investigated. Let us mention the monographs by H. Brunner [3, 4]. A different approach for problems with weakly singular kernel is the use of discretized fractional calculus by C. Lubich [13] (see also [8]), and the oblivious convolution quadrature [14, 12] for more general kernels. Available codes mostly make use of constant step size.

Implementing an integration method for the considered class of problems is more or less straight-forward, if constant time steps are used. However, in realistic situations one is often confronted with initial layers (like the problem of Section 6.3) and/or fast transitions between different states, so that flexibility in the choice of step size is an essential ingredient for efficiency. Although there exist excellent codes for solving nonstiff and stiff ordinary differential equations, codes for differential-algebraic equations, and codes for delay differential equations equipped with sophisticated step size

[*]Division of Mathematics, Gran Sasso Science Institute, Via Crispi 7, I-67100 L'Aquila, Italy. Email: `nicola.guglielmi@gssi.it`

[†]Section de Mathématiques, Université de Genève, CH-1211 Genève 24, Switzerland. Email: `ernst.hairer@unige.ch`

strategies, we are not aware of a code based on a variable step size time integrator that can efficiently treat problems with distributed delays.

Instead of extending a numerical integrator for Volterra integro-differential equations to a variable step size code, we propose to change the problem in such a way that any code for stiff and differential-algebraic (delay) equations can be applied. The idea is to approximate the distribution kernel of the integral term by a sum of exponential functions (multiplied by a polynomial) and then to transform the integral delay term into a set of differential equations. The use of an approximation by a sum of exponential functions is not new in numerical analysis. In [2] it is applied for an efficient computation of high-dimensional integrals. In [12] it is used for developing a variable step size integrator that is applied to interesting problems like a blow-up problem for a nonlinear Abel integral equation and a fractional diffusion-reaction system. New in the present work is that we do not propose another time integrator, but we show that a large class of problems with distributed delay can be solved by standard software for stiff and differential-algebraic (delay) equations. This is illustrated at the hand of RADAU5 [9] for stiff and differential-algebraic equations, and its extension RADAR5 [6, 7] for problems including delay arguments. New is also an application to problems in pharmacokinetics and pharmacodynamics.

Outline of the paper. Section 2 introduces the class of problems that can be treated with the codes presented in this work. It comprises nonstiff, stiff, differential-algebraic, and delay differential equations, with the additional feature that distributed delay terms are included. In approximating the kernel of a distributed delay term by a sum of exponential functions multiplied by a polvnomial, the problem is transformed into a system without distributed delay terms. Since the resulting system is typically of a much larger dimension than the original problem, Section 3 presents an algorithm that reduces considerably the complexity of the required solution of linear systems. Furthermore, the stability of the proposed algorithm and the determination of the accuracy parameters are discussed. Section 4 describes the approach by Beylkin and Monzón [1] for approximating the factor $t^{-\alpha}$ (typically present in weakly singular kernels) by a sum of exponential functions. Parameters in the approximation are selected to keep the approximation error under a given level. Section 5 presents important examples of distributed delays (the gamma distribution and the Pareto distribution) commonly used in pharmacodynamics and pharmacokinetics. Finally, Section 6 provides numerical evidence of the efficiency of the proposed approach by applying the codes RADAU5 and RADAR5 to two test examples and to an example from chemotherapy-induced myelosuppression. The codes together with drivers for problems with distributed delays are made publicly available.

2. Differential equations with distributed delay.

We consider differential equations of the form

$$M\dot y(t) = f\big(t, y(t), y(t-\tau), I(y)(t)\big), \qquad y(0) = y_0, \quad y(t) = \eta(t) \ \text{ for } \ t < 0, \tag{2.1}$$

where $y \in \mathbb{R}^d$, M is a constant $d \times d$ matrix, the delay satisfies $\tau \geq 0$, and

$$I(y)(t) = \int_0^t k(t-s)\, g\big(s, y(s)\big)\, \mathrm{d}s \tag{2.2}$$

is a distributed delay term. The vector functions $\eta(t)$, $f(t, y, v, I)$, and the scalar functions $k(t)$, $g(t, y)$ are assumed to be smooth. The kernel $k(t)$ is allowed to have an integrable singularity at the origin.

This formulation contains ordinary differential equations, differential-algebraic equations (for singular M), delay differential equations, and integro-differential equations as special cases. All considerations of the present work extend straight-forwardly to the case, where the delay $\tau = \tau(t, y(t)) \geq 0$ is time and state dependent, and where several delay terms and several integral terms $I_i(t)(y)$ (with different kernels $k_i(t)$ and different functions $g_i(t, y)$) are present. For notational convenience we restrict our study to the formulation (2.1). We focus on the situation, where at least one integral term (2.2) is present in (2.1).

Assume that we have at our disposal a reliable code for solving differential equations of the form (2.1), where no integral term is present. Let us mention the code Radau5 of [9] for stiff and differential-algebraic equations and its extension Radar5 of [6] for delay differential-algebraic equations. Our aim is to demonstrate how such codes can be applied to the solution of (2.1) with integral terms included, without changing them. The main idea is to approximate the kernel $k(t)$ by a finite sum of exponential functions multiplied by a polynomial. We thus assume that

$$k(t) = \sum_{i=1}^{n} p_i(t)\, \mathrm{e}^{-\gamma_i t}, \qquad p_i(t) = \sum_{j=0}^{m_i} c_{i,j}\, t^j \tag{2.3}$$

with real coefficients $c_{i,j}$ and γ_i, so that the distributed delay term (2.2) can be written as

$$I(y)(t) = \sum_{i=1}^{n} \sum_{j=0}^{m_i} c_{i,j} z_{i,j}(t), \qquad z_{i,j}(t) = \int_0^t (t-s)^j\, \mathrm{e}^{-\gamma_i (t-s)} g\big(s, y(s)\big)\, \mathrm{d}s. \tag{2.4}$$

Differentiation of $z_{i,j}(t)$ with respect to time yields, for $i = 1, \ldots, n$,

$$\dot z_{i,j}(t) = -\,\gamma_i\, z_{i,j}(t) + \begin{cases} g\big(t, y(t)\big) & j = 0 \\ j\, z_{i,j-1}(t) & j = 1, \ldots, m_i. \end{cases} \tag{2.5}$$

We insert (2.4) into the differential equation (2.1), we denote $z_i = \big(z_{i,0}, \ldots, z_{i,m_i}\big)^\top$, and we add these differential equations for $z_{i,j}(t)$ to the original equations. This gives the augmented system of differential equations

$$\begin{aligned} M\dot y(t) &= f\Big(t, y(t), y(t-\tau), \sum_{i=1}^{n}\sum_{j=0}^{m_i} c_{i,j} z_{i,j}(t)\Big), && \begin{array}{l} y(0) = y_0 \\ y(t) = \eta(t) \quad \text{for} \quad t < 0 \end{array} \\ \dot z_1(t) &= J_1 z_1(t) + e_1 g\big(t, y(t)\big) && z_1(0) = 0 \\ &\cdots && \cdots \\ \dot z_n(t) &= J_n z_n(t) + e_n g\big(t, y(t)\big) && z_n(0) = 0 \end{aligned} \tag{2.6}$$

for the super vector $\big(y, z_1, \ldots, z_n\big)$ of dimension $d + (m_1+1) + \ldots + (m_n+1)$. Here, we use the notation

$$J_i = \begin{pmatrix} -\gamma_i & 0 & \cdots & \cdots & 0 \\ 1 & -\gamma_i & \ddots & & \vdots \\ 0 & 2 & -\gamma_i & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & m_i & -\gamma_i \end{pmatrix} \in \mathbb{R}^{m_i+1, m_i+1}, \qquad e_i = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{m_i+1}$$

for the lower bidiagonal matrix and the first unit vector, both of dimension $m_i + 1$. This gives a formulation, where also those codes can be applied, that are not adapted to the treatment of distributed delays (2.1).

3. Solving distributed delay equations via the augmented system. For the case that the original system (2.1) is nonstiff and the exponents γ_i in (2.3) are of moderate size, any code for nonstiff ODEs or DDEs can be applied to the augmented system (2.6). However, in important applications (see Section 4) some of the γ_i can be very large, so that the augmented system is stiff independent of the behaviour of (2.1). Therefore, a stiff solver (e.g., RADAU5 for the case of ODEs, and RADAR5 in the presence of retarded arguments) has to be applied to (2.6). Stiff solvers are in general implicit and require an efficient solution of linear systems of the form (see [9, Section IV.8])

$$\Big((\gamma h)^{-1}\mathcal{M} - \mathcal{J}\Big)u = a, \tag{3.1}$$

where $a = (a_0, a_1, \ldots, a_n)^\top$ is a given vector and $u = (u_0, u_1, \ldots, u_n)^\top$ is the solution of the system. Here, γ is a real or complex parameter of the stiff integrator, h is the time step size, $\mathcal{M} = \mathrm{Diag}\,\big(M, I_1, \cdots, I_n\big)$ is a block diagonal matrix with I_i being the identity matrix of dimension $m_i + 1$, and the Jacobian matrix of the augmented system is

$$\mathcal{J} = \begin{pmatrix} J & C_1 & C_2 & \cdots & C_n \\ B_1 & J_1 & 0 & \cdots & 0 \\ B_2 & 0 & J_2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ B_n & 0 & \cdots & 0 & J_n \end{pmatrix}, \tag{3.2}$$

where $J = (\partial f/\partial y)(t, y, v, I)$ is a square matrix of dimension d, and the rank-one matrices B_i and C_i are given as follows: $B_i = e_i g_y^\top$ with $g_y = (\partial g/\partial y)(t, y)$, and $C_i = f_I c_i^\top$ with $f_I = (\partial f/\partial I)(t, y, v, I)$ and $c_i = (c_{i,0}, c_{i,1}, \ldots, c_{i,m_i})^\top$. All derivatives are evaluated at the current integration point.

3.1. Efficient solution of the linear system (3.1)**.** The linear system (3.1) is of dimension $d + N$, where $N = (m_1 + 1) + \ldots + (m_n + 1)$ can be much larger than d, the dimension of the original problem. Without exploiting the special structure of (3.1) its solution costs $\mathcal{O}\big((d + N)^3\big)$ flops. We propose to solve the linear system in the following way.

The first block of (3.1) is equivalent to

$$\Big((\gamma h)^{-1}M - J\Big)u_0 = a_0 + \sum_{i=1}^{n} C_i u_i, \tag{3.3}$$

and the other blocks give

$$\big((\gamma h)^{-1}I_i - J_i\big)u_i = a_i + B_i u_0. \tag{3.4}$$

This equation permits to express u_i in terms of u_0. Inserted into (3.3) yields

$$\Big((\gamma h)^{-1}M - J\Big)u_0 = a_0 + \sum_{i=1}^{n} C_i \Big((\gamma h)^{-1}I_i - J_i\Big)^{-1}\big(a_i + B_i u_0\big), \tag{3.5}$$

which can be written as

$$\Big((\gamma h)^{-1}M - \widehat{J}\Big)u_0 = a_0 + f_I \sum_{i=1}^{n} c_i^\top \Big((\gamma h)^{-1}I_i - J_i\Big)^{-1} a_i,$$

$$\widehat{J} = J + f_I g_y^\top \sum_{i=1}^{n} c_i^\top \Big((\gamma h)^{-1}I_i - J_i\Big)^{-1} e_i. \tag{3.6}$$

Note that $\widehat{J}$ is a rank-one perturbation of J. Since $\big((\gamma h)^{-1}I_i - J_i\big)$ is a lower bidiagonal matrix of dimension $m_i + 1$, the computation of the constant in the rank-one perturbation and the computation of the right-hand side of the linear system (3.6) require not more than $\mathcal{O}(N)$ flops. Having computed u_0, the u_i are obtained from (3.4). All in all, this gives an algorithm that has a cost of $\mathcal{O}(d^3) + \mathcal{O}(N)$ flops.

3.2. Stable approximation of the distributed delay term: exponential case. It may happen that several γ_n in the kernel approximation (2.3) are very large and positive, so that the summands are not well scaled. Consider, for example, the kernel (5.1) defined by a gamma distribution (see Section 5.1 below), which is approximated by

$$k(t) \approx ch \sum_{n=M}^{N-1} \mathrm{e}^{\alpha n h} \mathrm{e}^{-(\mathrm{e}^{nh}+\kappa)t}. \tag{3.7}$$

For large positive n (nh can be of size 50 or more), the exponent $\gamma_n = \mathrm{e}^{nh} + \kappa$ as well as the coefficient $\mathrm{e}^{\alpha n h}$ are very large, and the computation has to be done with care. For the kernel approximation above, the integral term (2.2) is approximated as

$$\begin{aligned} I(y)(t) &= \int_0^t k(t-s) g\big(s, y(s)\big)\,\mathrm{d}s \approx ch \sum_{n=M}^{N-1} \mathrm{e}^{\alpha n h} z_n(t), \\ z_n(t) &\approx \int_0^t \mathrm{e}^{-\gamma_n (t-s)} g\big(s, y(s)\big)\,\mathrm{d}s, \end{aligned} \tag{3.8}$$

where $z_n(t)$ is the solution of the differential equation (see Section 2)

$$\dot{z}_n(t) = -\gamma_n z_n(t) + g\big(t, y(t)\big), \qquad z_n(0) = 0.$$

It is discretized numerically by a time integrator, typically a Runge–Kutta method. For example, the θ-method, gives the approximation $z_n^k \approx z_n(t_k)$ via

$$z_n^{k+1} = z_n^k - \gamma_n \Delta t\Big(\theta z_n^{k+1} + (1-\theta) z_n^k\Big) + \Delta t g_n^{k+1}, \quad g_n^{k+1} = g\Big(t_k + \theta\Delta t, \theta y_n^{k+1} + (1-\theta) y_n^k\Big),$$

which can also be written as

$$z_n^{k+1} = R(-\gamma_n \Delta t) z_n^k + \Delta t S(-\gamma_n \Delta t) g_n^{k+1}, \quad R(\mu) = \frac{1 + (1-\theta)\mu}{1 - \theta\mu}, \quad S(\mu) = \frac{1}{1-\theta\mu}.$$

Solving this recursion and using $z_n^0 = 0$ yields

$$z_n^{k+1} = \Delta t \sum_{j=0}^{k} R(-\gamma_n \Delta t)^j S(-\gamma_n \Delta t) g_n^{k+1-j}. \tag{3.9}$$

Assuming $|g(t, y(t))| \le G$ to be bounded, and $|R(-\gamma_n \Delta t)| < 1$ we obtain (with $\mu = -\gamma_n \Delta t$)

$$|z_n^{k+1}| \le \Delta t |S(\mu)| \frac{1 - |R(\mu)|^{k+1}}{1 - |R(\mu)|} G \le \frac{\Delta t |S(\mu)| G}{1 - |R(\mu)|}. \tag{3.10}$$

Any other Runge–Kutta method will lead to a similar recursion, where $R(\mu)$ is its stability function, and $S(\mu)$ is a rational function bounded by $\mathcal{O}(\mu^{-1})$. This shows that for a stable approximation of the integral term (3.8), A_0-stability (i.e., $|R(\mu)| < 1$ for $\mu < 0$) is a necessary condition, and $|R(\infty)| < 1$ is recommended. These conditions imply that $|z_n^{k+1}| \le \mathcal{O}(\Delta t |\mu|^{-1}) = \mathcal{O}(\gamma_n^{-1})$.

Let us go back to the question of large coefficients in the approximation (3.8). In the exact solution, a large coefficient $\mathrm{e}^{\alpha nh}$ is compensated by the small factor $\mathrm{e}^{-(\mathrm{e}^{nh}+\kappa)t}$ in (3.7). In the numerical approximation, it is compensated by $\gamma_n^{-1} \le \mathrm{e}^{-nh}$.

3.3. Quasipolynomial case. Consider now the situation, where some of the polynomials in (2.3) have a degree at least 1. In this case $z_n(t)$ is a vector and satisfies the differential equation

$$\dot z_n(t) = J_n z_n(t) + e_n g(t, y(t)), \qquad z_n(0) = 0 \in \mathbb{R}^{m_n+1}.$$

A Runge-Kutta discretization yields

$$\begin{aligned} z_n^{k+1} &= R(\Delta t J_n) z_n^k + \Delta t S(\Delta t J_n) e_n g_n^{k+1} \quad \text{and} \\ z_n^{k+1} &= \Delta t \sum_{j=0}^{k} R(\Delta t J_n)^j S(\Delta t J_n) e_n g_n^{k+1-j} \end{aligned} \tag{3.11}$$

with $R(\Delta t J_n)$ and $S(\Delta t J_n)$ now rational matrix functions. They are of the form (with $\gamma = \gamma_n$, $m = m_n$, and $\mu = -\gamma_n \Delta t$)

$$R(\Delta t J_n) = \begin{pmatrix} R(\mu) & 0 & 0 & \cdots & 0 \\ \binom{1}{1}\Delta t R'(\mu) & R(\mu) & 0 & & \vdots \\ \binom{2}{2}\Delta t^2 R''(\mu) & \binom{2}{1}\Delta t R'(\mu) & R(\mu) & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \binom{m}{m}\Delta t^m R^{(m)}(\mu) & \binom{m}{m-1}\Delta t^{m-1} R^{(m-1)}(\mu) & \cdots & \binom{m}{1}\Delta t R'(\mu) & R(\mu) \end{pmatrix}$$

This is a consequence of $R(-\Delta t \gamma I + \Delta N) = \sum_{l=0}^{m-1} \frac{\Delta t^l}{l!} R^{(l)}(\mu) N^l$, where N is the nilpotent matrix whose only non-zero elements are in the first subdiagonal. For the matrix $S(\Delta t J_n)$ we get the same formulas with $R(\mu)$ replaced by $S(\mu)$. We are interested to bound the norm of the vector z_n^{k+1} in (3.11). The first component $z_{n,0}^{k+1}$ is given by the relation (3.9) and therefore satisfies the estimate (3.10). To estimate the second component $z_{n,1}^{k+1}$ we have to compute the elements of the second row of $R(\Delta t J_n)^j$. The diagonal element is $R(\mu)^j$ and the element left to it is $j \Delta t R'(\mu) R(\mu)^{j-1}$. From (3.11) we thus get

$$z_{n,1}^{k+1} = \Delta t \sum_{j=0}^{k} \Big(j \Delta t R'(\mu) R(\mu)^{j-1} S(\mu) + R(\mu)^j \Delta t S'(\mu) \Big) g_n^{k+1-j}.$$

Applying the triangle inequality, using the bound $|g(t,y(t))| \le G$ and the estimates (for $0 \le r < 1$)

$$\sum_{j=0}^{k} r^j = \frac{1-r^{k+1}}{1-r} \le \frac{1}{1-r}, \quad \sum_{j=0}^{k} j\, r^{j-1} \le \frac{1}{(1-r)^2}, \quad \sum_{j=0}^{k} \binom{j}{l} r^{j-l} \le \frac{1}{(1-r)^{l+1}}, \tag{3.12}$$

(the second and last inequalities are obtained by differentiating the infinite geometric series and by truncating higher order terms, which are positive) we get

$$|z_{n,1}^{k+1}| \le \Delta t^2 \Big(\frac{|R'(\mu) S(\mu)|}{(1-|R(\mu)|)^2} + \frac{|S'(\mu)|}{1-|R(\mu)|} \Big) G. \tag{3.13}$$

Applying the fact that $|S(\mu)| = \mathcal{O}(\mu^{-1})$, $|S'(\mu)| = \mathcal{O}(\mu^{-2})$, and $|R'(\mu)| = \mathcal{O}(\mu^{-2})$, we find that $|z_{n,1}^k| = \mathcal{O}(\gamma_n^{-2})$, and the same conclusion as in Section 3.2 can be drawn.

For general m we use the multinomial theorem (the indices i_l are non-negative)

$$\Big(\sum_{l=0}^{m-1} \frac{\Delta t^l}{l!} R^{(l)}(\mu)\, N^l \Big)^j = \sum_{i_1+\ldots+i_m=j} \binom{j}{i_1,\ldots,i_m} \prod_{l=0}^{m-1} \Big(\frac{\Delta t^l}{l!} R^{(l)}(\mu)\, N^l \Big)^{i_{l+1}} \tag{3.14}$$

and the inequalities (3.12) for values of l up to m. Since the nilpotent matrix satisfies $N^m = 0$, only the terms with

$$i_2 + 2\, i_3 + \ldots + (m-1)\, i_m \le m-1 \tag{3.15}$$

give raise to non-vanishing expressions in (3.14). Therefore, only i_1 can be large. It can take only the values $j,\, j-1, \ldots, j-m+1$, because otherwise $i_2 + \ldots + i_m$ would have to be larger than $m-1$, contradicting the restriction (3.15).

We start with $i_1 = j$. In this case all other i_l are zero and there is only one term in (3.14). The sum over j from 0 to k is bounded by the truncated geometric series (see (3.10)).

For $i_1 = j-1$ only one index among $\{i_2, \ldots, i_m\}$ can be non-zero (equal to 1, if $m \ge 2$). In each case the multinomial coefficient in (3.14) equals j, and the second sum in (3.12) provides the desired bound (see (3.13)).

For $i_1 = j-2$ we have two possibilities. Either one among the indices $\{i_2, \ldots, i_m\}$ equals 2 (if $m \ge 3$), or two of them equal 1 (provided that $m \ge 4$). In each case the multinomial coefficient in (3.14) equals $j(j-1)/2$ or $j(j-1)$, and the estimate of (3.12) with $l = 2$ can be applied. Using $S''(\mu) = \mathcal{O}(\mu^{-3})$ and similar estimates for the stability function $R(\mu)$, we obtain $|z_{n,2}^k| = \mathcal{O}(\gamma_n^{-3})$.

This procedure can be continued until $i_1 = j-m+1$. We obtain the bounds $|z_{n,l}^k| = \mathcal{O}(\gamma_n^{-l-1})$ for the $(l+1)$th component of the vector z_n^k.

3.4. Determination of the accuracy parameters. The variables $z_{i,j}(t)$ are auxiliary variables in the system (2.6) and the question arises, how accurate they should be. For the original problem (2.1) the user has to specify the desired accuracy of the solution. This is typically done with help of the parameters *Atol* and *Rtol*, with the aim of having a local error for $y(t)$ that is bounded by $Atol + |y| Rtol$ (for more details see [9, p. 124]). Of course, such a requirement can be applied component-wise. For solution components with a very small modulus that have a strong impact on the other solution components it is in general advisable to use a parameter *Atol* that is much smaller than *Rtol*.

In the situation of the present work, it is not necessary that all variables $z_{i,j}(t)$ are computed very accurately. Since the solution $y(t)$ only depends on a linear combination of them (with possibly very small coefficients), it is sufficient that this linear combination is sufficiently accurate. This can be achieved as follows: we augment the dimension of $y(t)$ by one and introduce the new variable $y_{d+1}(t)$ by

$$0=\sum_{i=1}^{n}\sum_{j=0}^{m_i} c_{i,j}z_{i,j}(t)-y_{d+1}(t). \tag{3.16}$$

We replace the double sum in (2.6) by $y_{d+1}(t)$ and we add the relation (3.16) to (2.6). This gives a differential-algebraic system where the number of algebraic variables is augmented by one. The advantage of this is that we can require different accuracies for the $z_{i,j}$ and for their sum (3.16). For a given tolerance Tol we propose (in general) to put $Atol = Rtol = Tol$ for all $d + 1$ components of the augmented vector y, and we put $Atol = Rtol = \omega\, Tol$ (with $\omega \geq 1$) for all variables $z_{i,j}$. To increase efficiency without spoiling accuracy we propose to take $\omega = 100$ or even larger (see the second experiment in Section 6.1 and Table 6.2).

4. Approximating the kernel with a sum of exponentials.

Important kernels, such as the gamma distribution and the Pareto distribution, contain the function $t^{-\alpha}$ as factor. This section is devoted to approximate it by a sum of exponential functions.

4.1. Approach of Beylkin and Monzón.

Since the Laplace transform of the function $z^{\alpha-1}$ is $\Gamma(\alpha)\, t^{-\alpha}$ for $\alpha > 0$, we have the integral representation

$$t^{-\alpha}=\frac{1}{\Gamma(\alpha)}\int_0^\infty \mathrm{e}^{-tz}z^{\alpha-1}\,\mathrm{d}z=\frac{1}{\Gamma(\alpha)}\int_{-\infty}^{\infty}\mathrm{e}^{-te^{s}}\mathrm{e}^{\alpha s}\,\mathrm{d}s,\qquad \alpha>0. \tag{4.1}$$

To express this function as a sum of exponentials, [1] proposes to approximate the integral to the right by the trapezoidal rule. With a step size $h > 0$ this yields

$$T(t,h)=\frac{h}{\Gamma(\alpha)}\sum_{n=-\infty}^{\infty}\mathrm{e}^{\alpha nh}\mathrm{e}^{-\mathrm{e}^{nh}t}. \tag{4.2}$$

• *Error of the trapezoidal rule.* It follows from [15, Theorem 5.1] that the error due to this approximation can be bounded by

$$\big|t^{-\alpha}-T(t,h)\big|\leq\frac{2C}{\mathrm{e}^{2\pi a/h}-1},\qquad C=\sup_{b\in(-a,a)}\frac{1}{\Gamma(\alpha)}\int_{-\infty}^{\infty}\mathrm{e}^{-te^{s}\cos b}\mathrm{e}^{\alpha s}\,\mathrm{d}s=(t\cos a)^{-\alpha}$$

for any $h > 0$ and any $0 < a < \pi/2$. This gives the estimate

$$\big|t^{-\alpha}-T(t,h)\big|\leq c_\alpha t^{-\alpha},\qquad c_\alpha=\frac{2(\cos a)^{-\alpha}}{\mathrm{e}^{2\pi a/h}-1}. \tag{4.3}$$

To achieve $c_\alpha \leq \varepsilon$, the step size h has to satisfy the inequality in

$$h\leq\frac{2\pi a}{\ln\big(1+\frac{2}{\varepsilon}(\cos a)^{-\alpha}\big)},\qquad a=\frac{\pi}{2}\Big(1-\frac{\alpha}{(\alpha+1)\,\ln\varepsilon^{-1}}\Big). \tag{4.4}$$

The value of a is chosen so that, for a given $\varepsilon > 0$, the estimate for h in (4.4) turns out to be close to maximal.

• *Error from truncating the series* (4.2) *at* $-\infty$. Since the function $s \mapsto \mathrm{e}^{-t\mathrm{e}^s}\mathrm{e}^{\alpha s}$ is monotonically increasing for $s \le \ln(\alpha/t)$ we have, for $Mh \le \ln(\alpha/t)$,

$$\begin{aligned} E_M(t,h) &= \frac{h}{\Gamma(\alpha)} \sum_{n=-\infty}^{M-1} \mathrm{e}^{\alpha nh}\mathrm{e}^{-\mathrm{e}^{nh}t} \le \frac{1}{\Gamma(\alpha)} \int_{-\infty}^{Mh} \mathrm{e}^{-t\mathrm{e}^s}\mathrm{e}^{\alpha s}\,\mathrm{d}s \\ &= \frac{t^{-\alpha}}{\Gamma(\alpha)} \int_0^{t\mathrm{e}^{Mh}} \mathrm{e}^{-\sigma}\sigma^{\alpha-1}\,\mathrm{d}\sigma = t^{-\alpha}\Big(1 - \frac{\Gamma(\alpha, t\mathrm{e}^{Mh})}{\Gamma(\alpha)}\Big), \end{aligned} \tag{4.5}$$

where $\Gamma(\alpha, x) = \int_x^\infty \mathrm{e}^{-\sigma}\sigma^{\alpha-1}\,\mathrm{d}\sigma$ is the incomplete Gamma function.

• *Error from truncating the series* (4.2) *at* $+\infty$. The function $s \mapsto \mathrm{e}^{-t\mathrm{e}^s}\mathrm{e}^{\alpha s}$ is monotonically decreasing for $s \ge \ln(\alpha/t)$. The same computation as before therefore shows that for $Nh \ge \ln(\alpha/t)$

$$\begin{aligned} E^N(t,h) &= \frac{h}{\Gamma(\alpha)} \sum_{n=N}^{\infty} \mathrm{e}^{\alpha nh}\mathrm{e}^{-\mathrm{e}^{nh}t} \le \frac{1}{\Gamma(\alpha)} \int_{Nh}^{\infty} \mathrm{e}^{-t\mathrm{e}^s}\mathrm{e}^{\alpha s}\,\mathrm{d}s \\ &= \frac{t^{-\alpha}}{\Gamma(\alpha)} \int_{t\mathrm{e}^{Nh}}^{\infty} \mathrm{e}^{-\sigma}\sigma^{\alpha-1}\,\mathrm{d}\sigma = t^{-\alpha}\,\frac{\Gamma(\alpha, t\mathrm{e}^{Nh})}{\Gamma(\alpha)}. \end{aligned} \tag{4.6}$$

• *Choice of the step size h.* For a fixed accuracy requirement ε, we choose h according to (4.4). To study the quality of this approximation, we plot in Figure 4.1 for several values of α the relative error of the trapezoidal rule as a function of the step size h. We see that for $\alpha = 1/2$ the error of the trapezoidal rule is bounded by $\varepsilon = 10^{-5}$ whenever $h \le 0.78$. The estimate (4.4) gives $h \le 0.70$, which is a reasonably good approximation.

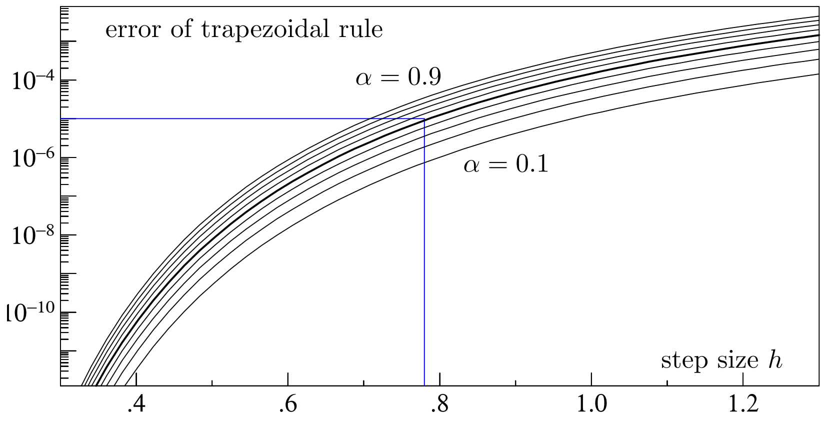

FIG. 4.1. *Relative error* $\big|t^{-\alpha} - T(t,h)\big|/t^{-\alpha}$ *as a function of the step size* h. *The different curves correspond to* $\alpha = 0.1, 0.2, \dots, 0.9$ *(bold curve for* $\alpha = 0.5$*).*

• *Choice of the truncation indices M and N.* For a treatment of the two truncation errors, we restrict t to the interval $0 < \delta \le t \le T < \infty$, and we use the estimates (4.5) and (4.6). The incomplete Gamma function is monotonically decreasing and satisfies $\Gamma(\alpha, 0) = \Gamma(\alpha)$, $\Gamma(\alpha, \infty) = 0$. We let x_* be such that $\Gamma(\alpha, x_*) \ge \Gamma(\alpha)(1-\varepsilon)$, and x^* be such that $\Gamma(\alpha, x^*) \le \Gamma(\alpha)\,\varepsilon$. Both truncation errors are bounded by $\varepsilon\, t^{-\alpha}$ for $t \in [\delta, T]$ if M and N are chosen according to $T\mathrm{e}^{Mh} \le x_*$ and $\delta\mathrm{e}^{Nh} \ge x^*$. Since

$\Gamma(\alpha,x) \geq \Gamma(\alpha) - x^\alpha/\alpha$, we can approximate x_* by the relation $x_*^\alpha = \Gamma(\alpha+1)\,\varepsilon$. From $\Gamma(\alpha,x) \leq x^{\alpha-1}\mathrm{e}^{-x}$ a suitable x^* is given by $(x^*)^{\alpha-1}\mathrm{e}^{-x^*} \leq \Gamma(\alpha)\,\varepsilon$, which is approximately $x^* = -\ln\bigl(\Gamma(\alpha)\,\varepsilon\bigr)$.

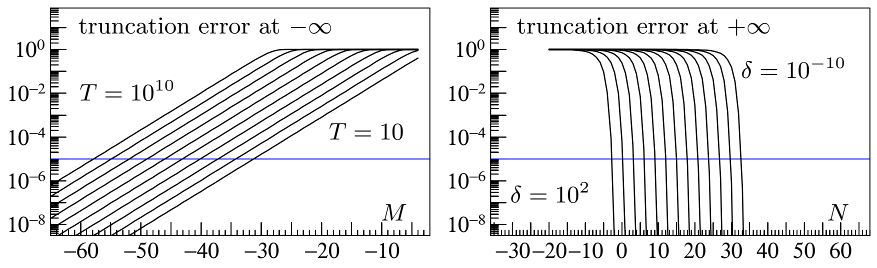

FIG. 4.2. *Relative error of the the truncation errors as a function of M and N, respectively.*

To get an impression of the size of the truncation indices M and N, we fix the values $\alpha = 0.5$ and $h = 0.78$, which correspond to $\varepsilon = 10^{-5}$ (see Figure 4.1). In the left picture of Figure 4.2 we plot max $\bigl\{E_M(t,h)/t^{-\alpha}\bigm| t \in [10^{-10},T]\bigr\}$ (the relative truncation error) as a function of M, for $T = 10, 10^2, \ldots, 10^{10}$. In the right picture of Figure 4.2 we plot max $\bigl\{E^N(t,h)/t^{-\alpha}\bigm| t \in [\delta,500]\bigr\}$ as a function of N, for $\delta = 10^2, 10, 1, 10^{-1}, \ldots, 10^{-10}$. Replacing the right end of the interval $[\delta,500]$ by a larger value does not change the picture. We have included the curves for large values of δ, because they are of interest for a treatment of the Pareto distribution (see Section 5.2).

As a conclusion we see that for an accuracy requirement of $\varepsilon = 10^{-5}$ and for an interval $[\delta,T]$ with $\delta = \varepsilon = 10^{-5}$ and $T = 100$, we can choose the step size $h = 0.78$, and the truncation indices $M = -35$ and $N = 18$ or (using h from (4.4)) $h = 0.70$, $M = -40$ and $N = 20$.

• *Total approximation error.* The error of approximating the function $t^{-\alpha}$ by a sum of exponentials is composed by the error of the trapezoidal rule and by those due to truncation. We have, for $\delta \leq t \leq T$,

$$\bigl|t^{-\alpha} - T_M^N(t,h)\bigr| \leq 3\,\varepsilon\, t^{-\alpha}, \qquad T_M^N(t,h) = \frac{h}{\Gamma(\alpha)} \sum_{n=M}^{N-1} \mathrm{e}^{\alpha n h}\mathrm{e}^{-\mathrm{e}^{nh}t}, \tag{4.7}$$

provided that h, M, and N are chosen as discussed above.

4.2. Approach of Braess and Hackbusch. An efficient computation of high-dimensional integrals is another motivation for studying the approximation of $t^{-\alpha}$ ($\alpha = 1/2$ or 1) by exponential sums (see [2]). The differences to the approach of Beylkin and Monzón are that the coefficients c_i and γ_i in the approximation $u_n(t) = \sum_{i=1}^n c_i \exp(-\gamma_i t)$ are computed for minimal n numerically with a Remez-like algorithm, and the absolute error $\bigl|t^{-\alpha} - u_n(t)\bigr|$ (not the relative error as in (4.7)) is kept below a tolerance ε.

5. Important examples of distributed delays. Because of their application in pharmacodynamics we are mainly interested in the situation where the kernel of the integral delay term (2.2) is a distribution, i.e., a non-negative function with $\int_0^\infty k(s)\,\mathrm{d}s = 1$. The following distributions are taken from the list given in [16].

5.1. Gamma distribution. For positive parameters $\kappa > 0$ and $0 < \alpha < 1$, the gamma distribution is given by

$$k(t) = \frac{\kappa^{1-\alpha}}{\Gamma(1-\alpha)}\, t^{-\alpha} \mathrm{e}^{-\kappa t}, \qquad t > 0. \tag{5.1}$$

We choose $\delta > 0$ such that $\int_0^\delta k(t)\,\mathrm{d}t \le \varepsilon$, and $T > 0$ such that $\int_T^\infty k(t)\,\mathrm{d}t \le \varepsilon$. Roughly speaking, the small parameter δ and the large parameter T have to be such that

$$\frac{(\kappa\,\delta)^{1-\alpha}}{\Gamma(2-\alpha)} \le \varepsilon \qquad \text{and} \qquad \frac{(\kappa\,T)^{-\alpha}\mathrm{e}^{-\kappa T}}{\Gamma(1-\alpha)} \le \varepsilon. \tag{5.2}$$

We use (4.7) for approximating the gamma distribution by a sum of exponential functions. For a given accuracy requirement ε, Algorithm 1 summarizes the computation of the parameters δ, T as well as the parameters h, M, N needed in (4.7).

Algorithm 1: Choice of the parameters for the Gamma distribution.

```
   Data: $\alpha$, $\kappa$, $\varepsilon$, $\delta_{\min}$, $t_f$
   Result: $h, T, \delta, M, N$
   begin
1      Set $a = \frac{\pi}{2}\Big(1 - \frac{\alpha}{(\alpha+1)\,\ln\varepsilon^{-1}}\Big)$
2      Set $h = \frac{2\pi a}{\ln\big(1 + \frac{2}{\varepsilon}(\cos a)^{-\alpha}\big)}$
3      Compute $T$ such that $(\kappa\,T)^{-\alpha}\mathrm{e}^{-\kappa T}/\,\Gamma(1-\alpha) = \varepsilon$
4      Set $T = \min\{t_f, T\}$
5      Set $x_* = (\Gamma(\alpha+1)\,\varepsilon)^{1/\alpha}$
6      Set $x^* = -\ln\big(\Gamma(\alpha)\,\varepsilon\big)$
7      Compute $M$ such that $T\mathrm{e}^{Mh} = x_*$
       Compute $\delta$ such that $(\kappa\,\delta)^{1-\alpha}/\,\Gamma(2-\alpha) = \varepsilon$
8      Set $\delta = \max\{\delta, \delta_{\min}\}$
9      Compute $N$ such that $\delta\mathrm{e}^{Nh} = x^*$
       return
```

• *Negative values of α.* For negative values of α the function $k(t)$ vanishes at the origin, but the sum of exponentials (4.2) does not. Let α satisfy $-1 < \alpha < 0$. The idea is to split $t^{-\alpha} = t \cdot t^{-\alpha-1}$, so that $\alpha + 1 \in (0, 1)$, and the formula (4.2) can be applied to the factor $t^{-\alpha-1}$. This results in a linear combination of functions $t\,\mathrm{e}^{-\gamma t}$, which can be treated as explained in Section 2.

For values of α satisfying $-2 < \alpha < -1$, we split $t^{-\alpha} = t^2 \cdot t^{-\alpha-2}$. The above procedure then leads to kernels of the form (2.3) with $m_i = 2$. An extension to other non-integer negative values of α is straight-forward.

5.2. Pareto distribution. The so-caled type I Pareto distribution [16] is given for $\alpha > 0$ and $\beta > 0$ by

$$k(t) = \begin{cases} 0 & 0 \le t < \beta \\ \alpha\,\beta^\alpha\, t^{-\alpha-1} & t \ge \beta. \end{cases} \tag{5.3}$$

Note that there is no singularity at the origin. For the approximation (4.7) of $t^{-\alpha-1}$, we can choose $\delta = \beta$ independent of the accuracy ε, so that the condition $\delta\mathrm{e}^{Nh} \ge x^*$

is less restrictive for the truncation index N. The condition $\int_T^\infty k(t)\,\mathrm{d}t \le \varepsilon$ for T reduces to $T^\alpha \ge \beta^\alpha/\varepsilon$. The truncation index M is determined by $T\mathrm{e}^{Mh} \le x_*$, where T is the minimum of $\beta/\varepsilon^{1/\alpha}$ and of the length of the integration interval T_{end} (see Algorithm 2).

Algorithm 2: Choice of the parameters for the Pareto distribution.

```
  Data: $\alpha$, $\beta$, $\varepsilon$, $t_f$
  Result: $h, T, M, N$
  begin
1     Set $T = \beta\,\varepsilon^{-1/\alpha}$
2     Set $T = \min\{t_f, T\}$
3     Set $a = \frac{\pi}{2}\Big(1 - \frac{(\alpha+1)}{(\alpha+2)\,\ln\varepsilon^{-1}}\Big)$
4     Set $h = \frac{2\pi a}{\ln\big(1 + 2\varepsilon^{-1}(\cos a)^{-(\alpha+1)}\big)}$
5     Set $x_* = \Gamma(\alpha+2)\,\varepsilon$
6     Set $x^* = -\ln\big(\Gamma(\alpha+1)\,\varepsilon\big)$
7     Compute $M$ such that $T\mathrm{e}^{Mh} = x_*$
      Compute $N$ such that $\beta\mathrm{e}^{Nh} = x^*$
      return
```

In this case the integral in (2.2) is from 0 to $t-\beta$. When replacing the kernel (on the interval $[\beta, T]$) by a sum of exponential functions, we are thus lead to

$$M\dot y(t) = f\big(t, y(t), y(t-\tau), I_P(t-\beta)\big) \tag{5.4}$$

where the function $I_P(t)$ is given by (with $\gamma_n = \mathrm{e}^{nh}$)

$$I_P(t) = \begin{cases} 0 & \text{if} \quad t \le 0 \\ \alpha\beta^\alpha \dfrac{h}{\Gamma(\alpha)} \displaystyle\sum_{n=M}^{N-1} \mathrm{e}^{(\alpha+1)nh}\mathrm{e}^{-\gamma_n\beta} z_n(t) & \text{if} \quad t \ge 0 \end{cases} \tag{5.5}$$

and $z_n(t) = \displaystyle\int_0^t \mathrm{e}^{-\gamma_n(t-s)} g\big(s, y(s)\big)\,\mathrm{d}s$ gives raise to the differential equation

$$\dot z_n(t) = -\gamma_n z_n(t) + g\big(t, y(t)\big), \qquad z_n(0) = 0. \tag{5.6}$$

As we have done in Section 3.4 we consider $I_p(t)$ as a new variable of the system, and we add the algebraic relation (5.5) to the augmented system (5.4)-(5.6). In this way we can monitor the choice of the accuracy parameters *Atol* and *Rtol* as explained in Section 3.4. The introduction of the variable $I_P(t)$ in the system has the further advantage that during the numerical integration only the back values of the scalar function $I_P(t)$ have to be stored. Note that the augmented system has additional breaking points at $i\tau + j\beta$ with non-negative integers i, j.

• *Remark.* For small positive α (e.g., $\alpha = 0.15$ and smaller, see [16, Table 1]) the condition $T^\alpha \ge \beta^\alpha/\varepsilon$ can lead to a very large T, which then results in a large negative truncation index M. To increase efficiency, we propose to introduce the new variable

$$y_{d+1}(t) = \alpha\beta^\alpha \int_0^{t-\beta} (t-s)^{-\alpha-1} g\big(s, y(s)\big)\,\mathrm{d}s,$$

which is the argument $I(y)(t)$ in (2.1), and to differentiate it with respect to t. This yields the delay differential equation

$$\dot{y}_{d+1}(t) = \alpha\beta^{-1} g\big(t-\beta, y(t-\beta)\big) - \alpha(\alpha+1)\beta^{\alpha} \int_0^{t-\beta} (t-s)^{-\alpha-2} g\big(s, y(s)\big)\,\mathrm{d}s,$$

which can be added to the system for $y(t)$. The integral in the right-hand side is again of Pareto type, but with α replaced by $\alpha+1$. Replacing the new kernel by a sum of exponentials gives a more efficient algorithm. Of course, this procedure can be repeated to increase the parameter α even more.

6. Numerical experiments.

This section provides numerical evidence of the proposed algorithm. The first two examples are scalar test equations (one ordinary differential equation with a Gamma-distributed delay term, the other a delay differential equation with a Pareto-distributed delay term). The third example is taken from applications in chemotherapy-induced myelo-suppression. For the time integration we make use of the code RADAU5 (see[9]), if the augmented system is an ordinary differential equation, and of the code RADAR5 (see [6, 7]), if the augmented system is a delay differential equation.

6.1. Example 1: ordinary differential equation with a Gamma-distributed delay term.

As a first test example we consider a linear differential equation

$$\dot{y}(t) = \big(1-y(t)\big)\,\mathrm{erf}\left(\frac{\sqrt{t}}{2}\right) - \frac{\mathrm{e}^{-t/4}\sqrt{t}}{\sqrt{\pi}} + I\big(t, y(t)\big) + \frac{1}{2}, \qquad y(0) = 0, \tag{6.1}$$

where $y(t)$ is a real-valued function, erf denotes the error function, and the distributed delay term is

$$I(y)(t) = \int_0^t k(t-s)y(s)\,\mathrm{d}s, \qquad k(t) = \frac{\mathrm{e}^{-t/4}}{2\sqrt{\pi t}} \tag{6.2}$$

Here, $k(t)$ is the Gamma-distribution (5.1) with parameters $\kappa = 1/4$ and $\alpha = 1/2$. Note that the kernel is weakly singular, but that there is no singularity in the augmented system (2.6). The inhomogeneity of the equation is chosen such that

$$y(t) = t/2$$

is the exact solution. This follows from the fact that with $y(t) = t/2$ we have

$$I(y)(t) = \frac{t-2}{2}\,\mathrm{erf}\left(\frac{\sqrt{t}}{2}\right) + \frac{e^{-t/4}\sqrt{t}}{\sqrt{\pi}}.$$

First experiment (connection between Tol and ε). We fix the tolerance $Tol = 10^{-8}$ for the time integrator, and we vary the value of ε, which determines the accuracy of the approximation by the sum of exponentials. We consider the integration interval $[0, t_f]$ with $t_f = 50$, so that the exact solution at the final point is $y(t_f) = 25$.

According to Algorithm 1 of Section 5.1 we compute, for given $\varepsilon > 0$ and for $\delta_{\min} = 0$, the parameters h, T, M, and N. For $\varepsilon = Tol = 10^{-8}$, we compute $T = \min\{50, 65.79\} = 50$ and $\delta = \pi \cdot 10^{-16}$. The suggested parameter values are $h = 0.4638$, $M = -89$, $N = 84$ which are indicated in bold in Table 6.1. There, the values of h, T, M, N are also given for further values of ε. The value of δ is $\delta = \pi\varepsilon^2$.

ε	h	T	M	N	err
10^{-4}	0.84	30.49	-27	24	$2.45 \cdot 10^{-4}$
10^{-5}	0.70	39.20	-39	35	$2.75 \cdot 10^{-5}$
10^{-6}	0.60	48.00	-54	49	$2.35 \cdot 10^{-6}$
10^{-7}	0.52	50.00	-70	65	$2.40 \cdot 10^{-7}$
$\mathbf{10^{-8}}$	**0.46**	**50.00**	$\mathbf{-89}$	**84**	$\mathbf{1.71 \cdot 10^{-8}}$
10^{-9}	0.42	50.00	-110	104	$4.72 \cdot 10^{-10}$
10^{-10}	0.38	50.00	-133	127	$2.14 \cdot 10^{-9}$
10^{-11}	0.35	50.00	-158	152	$2.08 \cdot 10^{-9}$

TABLE 6.1
Error behavior obtained by applying RADAU5 *to the test problem* (6.1) *with final point* $t_f = 50$ *and* $Tol = 10^{-8}$ *for different computed values of* h, M, *and* N *(corresponding to* ε*).*

For the numerical integration we apply the code RADAU5 (the code RADAR5 would give similar results) with accuracy requirement $Atol = Rtol = Tol = 10^{-8}$ to the augmented system (2.6) with parameters h, M, N, determined by different values of ε. The initial step size is $h = \varepsilon$. The last column of Table 6.1 reports the relative error err $:= |y(t_f) - \bar{y}|/|y(t_f)|$ at the final point. Here, $\bar{y}$ denotes the numerical approximation computed at time t_f. We can observe that for $\varepsilon \le Tol$ the error is essentially proportional to ε, which corresponds to the error of the kernel approximation. For $\varepsilon \ge Tol$ the error remains close to Tol, which shows the error of the time integration.

	$\omega = 1$			$\omega = 10$			$\omega = 100$		
ε	err	#fe	cpu	err	#fe	cpu	err	#fe	cpu
10^{-4}	2.5*e*-4	81	2.5*e*-4	2.5*e*-4	66	2.2*e*-4	2.5*e*-4	66	2.2*e*-4
10^{-6}	2.4*e*-6	162	7.9*e*-4	2.3*e*-6	132	6.6*e*-4	2.3*e*-6	117	6.0*e*-4
10^{-8}	1.8*e*-8	365	2.7*e*-3	1.6*e*-8	279	2.1*e*-3	1.5*e*-8	243	1.8*e*-3
10^{-10}	5.8*e*-11	773	7.6*e*-3	1.1*e*-11	587	6.2*e*-3	1.2*e*-10	482	5.1*e*-3

TABLE 6.2
Error behavior of RADAU5 *applied to the test problem* (6.1) *with* $t_f = 50$, *initial step size* $h = 0.1$, *and* $Atol = Rtol = \varepsilon$ *for* y *and* y_{d+1} *(see* (3.16)*), and* $Atol = Rtol = \omega\varepsilon$ *for the auxiliary variables* $z_{i,j}$. *Here,* err, #fe, cpu *indicate the relative error of* $y(t)$, *the number of function evaluations and the cpu time.*

Second experiment (choice of the accuracy parameters). As proposed in Section 3.4 we add a new variable $y_{1+1}(t)$ to the equation (6.1), and we apply the code RADAU5 to the augmented system with accuracy parameters $Atol = Rtol = \varepsilon$ for the y-components and $Atol = Rtol = \omega\varepsilon$ for the z-components. We first put $\omega = 1$, and then study the behaviour for increasing ω. The relative error of $y(t)$, the number of function evaluations, and the cpu time are given in Table 6.2. We observe that increasing ω does not affect the precision of the numerical approximation for $y(t)$, but significantly reduces the number of function evaluations and the cpu time. This is due to the fact that the numerical integrator can take larger step sizes. For every ε there seems to exist a critical value of ω, such that the numerical result does not change any more when ω is further increased. Numerical computations show that this happens for $\varepsilon = 10^{-4}$ when $\omega = 10$, for $\varepsilon = 10^{-6}$ when $\omega = 100$, for $\varepsilon = 10^{-8}$ when $\omega = 1000$, etc. We thus propose to use for this example $\omega = 0.1\varepsilon^{-1/2}$, i.e., $Rtol = \sqrt{\varepsilon}/10$.

6.2. Example 2: delay differential equation with a Pareto-distributed delay term. We next consider a test example with Pareto-distributed delay term. Since such a kernel leads in any case to a delay equation, we add a discrete delay term and consider

$$\dot{y}(t) = -5\, I(y)(t) - \frac{y\big(t-\tau\big)-2}{y(t)+1}, \qquad y(t) = t \;\text{ for }\; t \le 0, \tag{6.3}$$

where $y(t)$ is a scalar real function, the discrete delay term is $\tau = \pi/4$, and the distributed delay term is

$$I(y)(t) = \frac{\sqrt{\beta}}{2} \int_0^{t-\beta} (t-s)^{-3/2}\, y(s)\, \mathrm{d}s, \tag{6.4}$$

where the kernel is a Pareto distribution (5.3) with $\alpha = 1/2$. Since $t - s \ge \beta > 0$, there is no singularity in the integrand. Note that the system (5.4) depends on $y(t-\tau)$ and also $I_P(t-\beta)$, which gives rise to breaking points at integral multiples of τ and β and also their integer linear combinations.

ε	h	M	N	err
10^{-1}	1.662	-3	1	$7.69 \cdot 10^{-2}$
10^{-2}	1.116	-6	2	$8.97 \cdot 10^{-4}$
10^{-3}	0.851	-11	3	$2.31 \cdot 10^{-4}$
10^{-4}	0.692	-17	4	$2.81 \cdot 10^{-5}$
10^{-5}	0.586	-24	5	$1.37 \cdot 10^{-6}$
10^{-6}	0.509	-32	6	$3.46 \cdot 10^{-7}$
10^{-7}	0.451	-41	7	$1.90 \cdot 10^{-7}$
10^{-8}	**0.405**	$\mathbf{-51}$	**8**	$\mathbf{9.83 \cdot 10^{-8}}$
10^{-9}	0.368	-62	9	$5.95 \cdot 10^{-8}$
10^{-10}	0.337	-75	10	$1.75 \cdot 10^{-7}$
10^{-11}	0.311	-88	11	$2.40 \cdot 10^{-7}$

Table 6.3

Error behavior obtained by applying Radar5 *to the test problem* (6.3). *for different values of h, M and N, computed according Algorithm 2, with $Tol = 10^{-8}$*

In our experiments we set $\beta = 1$ and we consider the integration interval $[0, t_f]$ with $t_f = 10$. With this choice we compute a reference solution to high precision,

$$y(t_f) \approx 0.570525788119.$$

As explained in Section 5.2, an approximation of the kernel by a sum of exponentials gives rise to a delay differential equation. For an approximation error ε Algorithm 2 yields the parameters h, M, and N that are needed for the description of the augmented system.

For $\varepsilon = 10^{-8}$, the values are $h = 0.405$, $M = -51$ and $N = 8$, which are indicated in bold in the right side of Table 6.3. In contrast to the situation of Example 1, the resulting augmented system is not very stiff.

For the numerical solution of this problem we use the code Radar5. We consider a fixed tolerance $Tol = 10^{-8}$, and we apply the code with $Atol = Rtol = Tol$. We include in the mesh the first 10 breaking points,

$$\tau,\ \beta,\ 2\tau,\ \tau+\beta,\ 2\beta,\ 3\tau,\ 2\tau+\beta,\ 2\beta+\tau,\ 3\beta,\ 4\tau,$$

and let the code possibly compute further breaking points.

As initial step size we choose $h = Atol$. Similar to the first experiment for Example 1, we report the relative error in Table 6.3 at the final point. We get 120 steps (with no rejections), 854 function evaluations, 72 Jacobian evaluations, 99 LU decomposition of the matrix in the linear system and 244 solutions of triangular systems.

The result is very similar to that of Table 6.1. For a fixed tolerance Tol, the error is proportional to ε as long as $\varepsilon > Tol$. It stagnates at Tol for values of ε that are smaller than Tol.

6.3. Example 3: a distributed delay model of chemotherapy-induced myelosuppression. We consider a model system that is proposed and studied in [10, p. 56]. It uses ideas of the publication [5], where transit compartments in a semi-physiological model by Friberg are replaced by a convolution integral with a Gamma-distribution (see also [11]). The equations of the model are

$$\begin{aligned}
\dot y(t) &= \Big(\kappa\Big(\frac{w_0}{w(t)}\Big)^{\gamma} - k_s C(t) - \kappa\Big) y(t) \\
\dot w(t) &= -\kappa w(t) + \kappa I\big(t, y(t)\big) \\
\dot A(t) &= -\frac{V_{\max} A(t)}{K_m + C(t)}, \qquad C(t) = \frac{A(t)}{V},
\end{aligned} \tag{6.5}$$

where $y(t)$ gives the proliferating precursor cells for granulocytes, $w(t)$ the circulating granulocytes, and $A(t)$ is the amount of drug in the plasma. The distributed delay term, with a Gamma-distribution kernel, is given by

$$I(y)(t) = \int_0^t k(t-s) y(s)\,\mathrm{d}s, \qquad k(t) = \frac{\kappa^{1-\alpha}}{\Gamma(1-\alpha)}\, t^{-\alpha} \mathrm{e}^{-\kappa t}. \tag{6.6}$$

Table 6.4 gives two sets of parameters, which are taken from [10, Table 2]. Concerning the units of the parameters we refer to [10]. Time is measured in hours.

$\nu = 1-\alpha$	κ	w_0	γ	k_s	$V_{\max}$	K_m	V
0.964	$\nu/47.5$	14.4	0.664	0.0328	77.2	16.9	1.35
1.46	$\nu/55.6$	14.4	0.507	0.0213	100	22	1.03

Table 6.4
Values of the parameters for the system (6.5)

We note that the third equation of (6.5) is a scalar differential equation for $A(t)$, which is independent of the other two variables of the system. It can be solved analytically by separation of variables. In this way we find that $A(t)$ is solution of the nonlinear equation

$$\frac{K_m}{V_{\max}}\Big(\ln A(t) - \ln A_0\Big) + \frac{A(t) - A_0}{V_{\max} V} = -(t - t_0).$$

To get a better conditioning for its numerical treatment we write it as

$$A(t) = A_0 \exp\Big(-\frac{1}{K_m V}\big(A(t) - A_0\big) - \frac{V_{\max}}{K_m}(t - t_0)\Big). \tag{6.7}$$

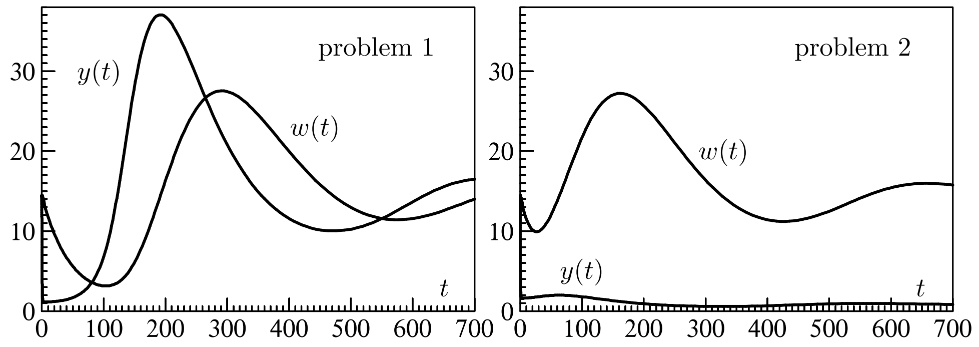

FIG. 6.1. *Solution components $y(t)$ and $w(t)$ of the system* (6.5) *for the parameter values of Table 6.4: upper row (problem 1) and lower row (problem 2).*

Without changing the solution of the problem we can replace in (6.5) the differential equation for $A(t)$ by the algebraic equation (6.7). This then leads to a differential-algebraic system of index 1, which can be solved efficiently by RADAU5 and by RADAR5. The formulation as a differential-algebraic system has the advantage that during the numerical integration there is no accumulation of errors in the variable $A(t)$. In our experience the DAE formulation is slightly more efficient than the ODE formulation (see Table 6.7).

As proposed in Section 3.4 we add a new variable $y_{3+1}(t)$ to the equation (6.5), and we apply the code RADAU5 to the augmented system. For various values of ε we use $Atol = Rtol = \varepsilon$ for the three components of (6.5), $Atol = Rtol = 10^{-2}\varepsilon$ for the y_{3+1}-component corresponding to the integral delay, and $Atol = Rtol = 10^{2}\varepsilon$ for the z-components. As initial step size we take $\max(\varepsilon, 10^{-5})$.

First experiment. We consider the problem (6.5) with parameters taken from the upper row of Table 6.4. Initial values are

$$y(0) = w_0, \qquad w(0) = w_0, \qquad A(0) = A_0 = 127. \tag{6.8}$$

The solution components $y(t)$ and $w(t)$ are plotted in the left picture of Figure 6.1. We do not include the graph of $A(t)$, because it monotonically decreases, rapidly approaches zero, and stays there for all t. We observe that the function $y(t)$ has an initial layer. It rapidly decreases from $y(0) = 14.4$ to a small value and then behaves smoothly.

The challenge of this problem is the fact that the value of $\alpha = 0.036$ is very small. According to Algorithm 1 this then leads to a very small x_*, and consequently to a large negative value of M, so that the dimension of the augmented system becomes large. The values of M, N (as well as h) are computed by Algorithm 1 of Section 5.1. They are presented for different values of ε in Table 6.5.

We apply the code RADAU5 for values of ε ranging between 10^{-3} and 10^{-10} on the interval $[0, 100]$. Since the Jacobian of the vector field is not banded, we apply the code in the standard way, where the linear algebra considers the Jacobian as a full matrix. The cpu time (measured in seconds) is given in the row of Table 6.5 marked "cpu (full)". We also have implemented the treatment of the arising linear systems according to the algorithm of Section 3.1, and the cpu times are listed in the row of Table 6.5 marked "cpu (sumexp)". The improvement is amazing. Already for $\varepsilon = 10^{-3}$ the cpu time is decreased by a factor of 100 and for $\varepsilon = 10^{-10}$, where the

ε	10^{-3}	10^{-4}	10^{-6}	10^{-7}	10^{-9}	10^{-10}
M	-157	-268	-582	-783	-1276	-1567
N	4	8	20	27	45	56
cpu (full)	$1.9 \cdot 10^{-1}$	$1.3 \cdot 10^{0}$	$2.6 \cdot 10^{1}$	$8.3 \cdot 10^{1}$	$5.1 \cdot 10^{2}$	$1.2 \cdot 10^{3}$
cpu (sumexp)	$9.6 \cdot 10^{-4}$	$2.1 \cdot 10^{-3}$	$8.9 \cdot 10^{-3}$	$1.6 \cdot 10^{-2}$	$5.3 \cdot 10^{-2}$	$9.5 \cdot 10^{-2}$

Table 6.5
Comparison of different linear algebra solvers for the system (6.5) *with parameters of the upper row of Table 6.4.*

dimension of the augmented system is very large, the cpu time is even decreased by a factor of 10^4.

Second experiment. We next consider the set of parameters given in the lower row of Table 6.4. The solution is plotted in the right picture of Figure 6.1. Here, the value of $\alpha = -0.46$ is negative, and the trick explained in Section 5.1 has to be applied. This means that the factor $t^{-\alpha}$ in (6.6) is written as $t^{-\alpha} = t \cdot t^{-(\alpha+1)}$ with $\alpha + 1 = 0.54 \in (0, 1)$, and only the term $t^{-(\alpha+1)}$ is replaced by a sum of exponentials. This yields an approximation of the form (2.3) with $m_i = 1$.

As before we apply the code Radau5 with several different values of ε, and initial values (6.8). We only use the option (sumexp). This time we study the error of the numerical approximation. A reference solution is computed with a very high accuracy requirement. The exact solution of $A(100)$ is far below round-off. The row, indicated as "err" in Table 6.6 shows the relative error of $y(t)$ and $w(t)$ at time $t = 100$. The values of M and N are computed by Algorithm 1. They are of moderate size, even for very small ε. We realize that the error is nicely proportional to the accuracy requirement Tol.

ε	10^{-3}	10^{-5}	10^{-7}	10^{-9}	10^{-11}
M	-17	-38	-67	-105	-150
N	13	35	67	108	156
cpu (sumexp)	$4.7 \cdot 10^{-4}$	$1.7 \cdot 10^{-3}$	$5.1 \cdot 10^{-3}$	$1.5 \cdot 10^{-2}$	$3.9 \cdot 10^{-2}$
err	$5.3 \cdot 10^{-4}$	$1.4 \cdot 10^{-5}$	$1.5 \cdot 10^{-7}$	$3.3 \cdot 10^{-9}$	$1.0 \cdot 10^{-10}$

Table 6.6
Study of accuracy of Radau5 *for the system* (6.5) *with parameters of the lower row of Table 6.4.*

Third experiment. The equations (6.5) are an ODE formulation of the problem of Section 6.3. When the differential equation for $A(t)$ is replaced by the algebraic equation (6.7) we get a DAE formulation with two differential equations and one algebraic relation. The code Radau5 can be applied to each of these formulations, and it is interesting to know which one performs better. In Table 6.7 we present for various ε the error at the final point $t_f = 100$, as well as the required number of steps and function evaluations. We observe that, for the DAE formulation (the final three columns), not only the error is smaller (excepting $\varepsilon = 10^{-3}$), but also the number of steps and function evaluations (and consequently the cpu time) are smaller.

6.4. Implementation of the approach of Section 3.1 for solving the linear system. The code Radau5 [9] (and similarly the code Radar5 [6]) consists of three files: Radau5.f (Radar5.f) contains the time integrator, decsol.f the linear algebra subroutines, and dc_decsol.f (dc_decdel.f) contains the subroutines

ε	h	M	N	err_1	$\#\mathrm{st}_1$	$\#\mathrm{fe}_1$	err_2	$\#\mathrm{st}_2$	$\#\mathrm{fe}_2$
10^{-3}	1.04	-17	13	$5.34 \cdot 10^{-4}$	25	161	$9.54 \cdot 10^{-3}$	23	154
10^{-5}	0.69	-38	35	$1.37 \cdot 10^{-5}$	47	287	$9.06 \cdot 10^{-6}$	38	262
10^{-7}	0.52	-67	67	$1.52 \cdot 10^{-7}$	80	507	$5.47 \cdot 10^{-8}$	68	483
10^{-9}	0.42	-105	108	$3.33 \cdot 10^{-9}$	152	985	$4.98 \cdot 10^{-10}$	126	945
10^{-11}	0.35	-150	156	$1.05 \cdot 10^{-10}$	309	2036	$2.41 \cdot 10^{-11}$	256	1933

TABLE 6.7

Performance of RADAU5 *for the problem* (6.5) *(coefficients from the lower row of Table 6.4) with* $t_f = 100$ *and* ε *ranging from* 10^{-3} *to* 10^{-11}. *Here,* err_1, $\#\mathrm{st}_1$, $\#\mathrm{fe}_1$ *indicate the relative error, the number of steps and the number of function evaluations for an implementation as ODE, while* err_2, $\#\mathrm{st}_2$, $\#\mathrm{fe}_2$ *are the numbers corresponding to an implementation as DAE.*

that link the time integrator with the linear algebra subroutines. When using the linear algebra approach of Section 3.1 one only has to replace the file DC_DECSOL.F (or DC_DECDEL.F) by DC_SUMEXP.F (or DC_SUMEXPDEL.F)[1]. The other files need not be touched. In the driver the Jacobian has to be defined as follows: first of all, if d is the dimension of y in the original system and n the number of summands in (2.3), we have to define the dimension of the system as

$$N = \begin{cases} d+1+n+2 & \text{if all } m_i = 0 \text{ in (2.3)} \\ d+1+2n+2 & \text{if all } m_i = 1 \text{ in (2.3).} \end{cases}$$

The first d components correspond to those of the original system (2.1), the $(d+1)$th component is for the integral delay (3.16), and the next n (or $2n$) components are for the variables $z_{i,j}$. The final 2 components are not used in the right-hand side function of the problem. They are used only for the Jacobian.

The Jacobian has to be declared as banded with upper bandwidth MUJAC $= d$ and lower bandwith MLJAC $= \max(2-d, 0)$. The array FJAC then has N columns and $\max(d+1, 3)$ rows. In the left upper $(d+1) \times (d+1)$ matrix, the derivative of the vector field f (augmented by (3.16)) with respect to (y, y_{d+1}) has to be stored, in column $d+2$ the derivative of f with respect to the integral term I, and in column $d+3$ the gradient of $g(t,y)$ with respect to y. For the case that all $m_i = 0$, the coefficients $\{c_{i0} \,|\, i = 1, \ldots n\}$ are stored in the first row starting at position $d+4$, the exponents $\{\gamma_i \,|\, i = 1, \ldots n\}$ in the second row, and zeros in the third row. Finally, for the case that all $m_i = 1$, the coefficients $\{(c_{i0}, c_{i1}) \,|\, i = 1, \ldots n\}$ are stored in the first row starting at position $d+4$, the exponents $\{(\gamma_i, \gamma_i) \,|\, i = 1, \ldots n\}$ in the second row, and the subdiagonal $(1, 0, 1, 0, \ldots, 1, 0)$ of (3.2) in the third row.

The codes RADAU5 and RADAR5 together with the files DECSOL.F, DC_DECSOL.F, DC_SUMEXP.F together with drivers for the examples of this article are available at the address `http://www.unige.ch/~hairer/software.html`.

Conclusions. In this article we have described an efficient methodology to deal with important classes of distributed delays, by using RADAU5 or RADAR5, which are codes for the numerical integration of a large class of stiff and differential-algebraic equations, and with discrete delays (for RADAR5).

The methodology is based on a suitable expansion of the kernels in terms of exponential functions, which allows to transform the distributed delay into a set of ODEs or DDEs. Our numerical experiments confirm the efficiency of this methodology

[1]The subroutines DC_DECDEL.F and DC_SUMEXPDEL.F are restricted to the case where M is a (possibly singular) diagonal matrix.

which allows to get a much more versatile code, able to deal with both discrete and distributed memory effects, a feature that makes it very appealing in disciplines where these kind of delays naturally arise, as pharmacodynamics and pharmacokinetics.

Acknowledgments. Nicola Guglielmi acknowledges that his research was supported by funds from the Italian MUR (Ministero dell'Università e della Ricerca) within the 'PRIN 2022 Project "Advanced numerical methods for time dependent parametric partial differential equations with applications" and the Pro3 joint project entitled "Calcolo scientifico per le scienze naturali, sociali e applicazioni: sviluppo metodologico e tecnologico". He is also affiliated to the INdAM-GNCS (Gruppo Nazionale di Calcolo Scientifico).

Ernst Hairer acknowledges the support of the Swiss National Science Foundation, grant No.200020 192129.

REFERENCES

[1] G. Beylkin and L. Monzón. Approximation by exponential sums revisited. *Appl. Comput. Harmon. Anal.*, 28:131–149, 2010.
[2] D. Braess and W. Hackbusch. On the efficient computation of high-dimensional integrals and the approximation by exponential sums. In A. Kunoth R. DeVore, editor, *Multiscale, Nonlinear and Adaptive Approximation*, volume 56, pages 39 – 74. Springer Verlag, 2009.
[3] H. Brunner. *Collocation methods for Volterra integral and related functional differential equations*, volume 15 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2004.
[4] H. Brunner and P. J. van der Houwen. *The numerical solution of Volterra equations*, volume 3 of *CWI Monographs*. North-Holland Publishing Co., Amsterdam, 1986.
[5] D. Câmara De Souza, M. Craig, T. Cassidy, J. Li, F. Nekka, J. Bélair, and A.R. Humphries. Transit and lifespan in neutrophil production: implications for drug intervention. *Journal of Pharmacokinetics and Pharmacodynamics*, 45(1):59 – 77, 2018.
[6] N. Guglielmi and E. Hairer. Implementing Radau IIA methods for stiff delay differential equations. *Computing*, 67(1):1–12, 2001.
[7] N. Guglielmi and E. Hairer. Computing breaking points in implicit delay differential equations. *Adv. Comput. Math.*, 29(3):229–247, 2008.
[8] E. Hairer and P. Maass. Numerical methods for singular nonlinear integro-differential equations. *Appl. Numer. Math.*, 3(3):243–256, 1987.
[9] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer Series in Computational Mathematics 14. Springer-Verlag, Berlin, 2nd edition, 1996.
[10] W. Krzyzanski. Ordinary differential equation approximation of gamma distributed delay model. *Journal of Pharmacokinetics and Pharmacodynamics*, 46(1):53 – 63, 2019.
[11] W. Krzyzanski, S. Hu, and M. Dunlavey. Evaluation of performance of distributed delay model for chemotherapy-induced myelosuppression. *Journal of Pharmacokinetics and Pharmacodynamics*, 45(2):329 – 337, 2018.
[12] M. López-Fernández, C. Lubich, and A. Schädle. Adaptive, fast, and oblivious convolution in evolution equations with memory. *SIAM J. Sci. Comput.*, 30(2):1015–1037, 2008.
[13] C. Lubich. Discretized fractional calculus. *SIAM J. Math. Anal.*, 17(3):704–719, 1986.
[14] A. Schädle, M. López-Fernández, and C. Lubich. Fast and oblivious convolution quadrature. *SIAM J. Sci. Comput.*, 28(2):421–438, 2006.
[15] L.N. Trefethen and J.A.C. Weideman. The Exponentially Convergent Trapezoidal Rule. *SIAM Review*, 56(3):385–458, 2014.
[16] C.A. Wesolowski, S.N. Wanasundara, P.S. Babyn, and J. Alcorn. Comparison of the gamma-pareto convolution with conventional methods of characterising metformin pharmacokinetics in dogs. *J. Pharmacokinet. Pharmacodyn.*, 47:19–45, 2020.