Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MARÍA LÓPEZ-FERNÁNDEZ* CHRISTIAN LUBICH¶
ACHIM SCHÄDLE

Adaptive, Fast and Oblivious Convolution in Evolution Equations with Memory

*Departamento de Matemática Aplicada, Universidad de Valladolid.
¶Mathematisches Institut, Universität Tübingen.

ZIB-Report 06-45 (November 2006)

ADAPTIVE, FAST AND OBLIVIOUS CONVOLUTION IN EVOLUTION EQUATIONS WITH MEMORY

MARÍA LÓPEZ-FERNÁNDEZ*, CHRISTIAN LUBICH¶, AND ACHIM SCHÄDLE**

Abstract. To approximate convolutions which occur in evolution equations with memory terms, a variable-stepsize algorithm is presented for which advancing N steps requires only $O(N \log N)$ operations and $O(\log N)$ active memory, in place of $O(N^2)$ operations and $O(N)$ memory for a direct implementation. A basic feature of the fast algorithm is the reduction, via contour integral representations, to differential equations which are solved numerically with adaptive step sizes. Rather than the kernel itself, its Laplace transform is used in the algorithm. The algorithm is illustrated on three examples: a blow-up example originating from a Schrödinger equation with concentrated nonlinearity, chemical reactions with inhibited diffusion, and viscoelasticity with a fractional order constitutive law.

Key words. convolution quadrature, adaptivity, Volterra integral equations, numerical inverse Laplace transform, anomalous diffusion, fractional order viscoelasticity,

AMS subject classifications. 65R20, 65M99.

1. Introduction.

We consider the problems of computing the convolution

$$\int_0^t f(t-\tau)\, g(\tau)\, d\tau\,, \qquad 0 \le t \le T, \tag{1.1}$$

possibly with matrix-valued kernel f and vector-valued function g, and of solving evolution equations with memory containing such convolution integrals where g is not a function known in advance, but $g(\tau)$ depends on the solution at time τ of the integral equation or integro-differential equation. In previous papers [16, 21] we have developed convolution algorithms that are *fast* and *oblivious*: to approximate (1.1) on a grid $t = nh$ $(n = 0, 1, \ldots, N)$ with constant step size h and $Nh = T$, the algorithm requires

- $O(N \log N)$ operations,
- $O(\log N)$ evaluations of the Laplace transform $F = \mathcal{L}f$, none of f, and
- $O(\log N)$ active memory.

In the nth time step, g is evaluated at $t_n = nh$, but the history $g(t_j)$ for $j < n$ is forgotten in this algorithm, and only logarithmically few linear combinations of the values of g are kept in memory. This is to be contrasted with the $O(N^2)$ operations, $O(N)$ evaluations of the kernel f, and $O(N)$ memory for a naive implementation of a quadrature formula for (1.1). Moreover, we note that in many applications the Laplace transform F (the transfer function), rather than the kernel f (the impulse response), is known *a priori*. A basic feature of the fast algorithm is the reduction, via contour integral representations, to differential equations of the form $y' = \lambda y + g$ for suitable complex values of λ, which are solved numerically. It is not necessary to solve these differential equations with constant time step h, as was done in [16, 21], but the step size may instead be adapted to the behavior of g. This observation opens the

*Departamento de Matemática Aplicada, Universidad de Valladolid, Valladolid, Spain. E-mail: `marial@mac.cie.uva.es`. Supported by DGI-MCYT under project MTM 2004-07194 cofinanced by FEDER funds.

¶Mathematisches Institut, Universität Tübingen, Auf der Morgenstelle 10, D–72076 Tübingen, Germany. E-mail: `lubich@na.uni-tuebingen.de`. Supported by DFG, SFB 382.

**ZIB Berlin, Takustr. 7, D-14195 Berlin, Germany. E-mail: `schaedle@zib.de`. Supported by the DFG Research Center MATHEON "Mathematics for key technologies", Berlin.

way to an *adaptive* fast and oblivious convolution algorithm. Turning this simple idea into an efficient algorithm is, however, not as simple and the development of such an algorithm is precisely the topic of the present paper. The need to use adaptive time steps in solving evolutionary integro-differential equations in applications has been addressed at various places in the literature, e.g., by Adolfsson, Enelund & Larsson [4], Cao, Burrage & Abdullah [6], and Diethelm & Freed [8]. None of the adaptive algorithms proposed there, however, can make use of the convolution structure to reduce the $O(N^2)$ operation count and $O(N)$ memory requirements for N steps. The convolution algorithm proposed here works in the situation of a *sectorial* Laplace transform F:

$$\begin{aligned} &F \text{ is analytic in a sector } |\arg(s-\sigma)| < \pi - \varphi \text{ with } \varphi < \tfrac{1}{2}\pi, \\ &\text{and in this sector, } \ |F(s)| \le M\,|s|^{-\nu} \quad \text{for some real } M \text{ and } \nu > 0. \end{aligned} \tag{1.2}$$

An equivalent condition is that f is analytic in a complex sector containing the positive real half-axis $t > 0$, and is bounded by $|f(t)| \le C\,t^{\nu-1}e^{\sigma t}$ in this sector. A typical example is the fractional-power kernel $f(t) = t^{\nu-1}/\Gamma(\nu)$, which has the Laplace transform $F(s) = s^{-\nu}$. An essential ingredient of the algorithm is the discretization of the inversion formula for the Laplace transform, given by

$$f(t) = \frac{1}{2\pi i}\int_\Gamma e^{t\lambda}\,F(\lambda)\,d\lambda, \qquad t > 0, \tag{1.3}$$

with Γ a contour in the sector of analyticity oriented with increasing imaginary part and going to infinity with an acute angle to the negative real half-axis, so that $e^{t\lambda}$ decays fast for growing $|\lambda|$ along Γ. We will choose the contour as a hyperbola. Since we cannot obtain a uniformly good approximation for all $t \in (0,T]$ with a single contour Γ, we use different hyperbolas Γ_ℓ corresponding to geometrically growing intervals of uniform approximation, $t \in [B^{\ell-1}h_*, B^{\ell+1}h_*]$ with an integer $B \ge 1$, e.g., $B = 5$, and with a minimum step size h_*. The required number of contours is thus bounded by $\log_B(T/h_*)$. This logarithm shows up in the complexity estimates in place of $\log_B(N)$ for the fixed-stepsize algorithm. As in that case, it appears multiplied with the number of quadrature points on each hyperbola, which is $O(\log\frac{1}{\varepsilon})$ for an accuracy ε in the approximation of (1.3). In §2 we briefly review recent results from [14, 15] on the approximation of inverse Laplace transforms by discretized contour integrals. In §3 we describe the fast and oblivious algorithm for computing convolutions with variable time steps. The algorithm is then illustrated on various problems where adaptive time steps are important: a blow-up problem for a nonlinear Abel integral equation resulting from a nonlinear Schrödinger equation with concentrated nonlinearity (§4), a fractional diffusion-reaction system from chemical reaction kinetics (§5), and visco-elasticity with a weakly singular memory kernel in the constitutive equations, under applied forces that are switched on and off (§6).

2. Preparation: Numerical inversion of the Laplace transform. In the inversion formula (1.3) we choose Γ as the left branch of a hyperbola parameterized by

$$\mathbb{R} \to \Gamma : \ x \mapsto \gamma(x) = \mu(1 - \sin(a + ix)) + \sigma, \tag{2.1}$$

where $\mu > 0$ is a scale parameter, σ is the shift in (1.2), and $0 < a < \pi/2 - \varphi$. Thus, Γ is the left branch of the hyperbola with center at $(\mu, 0)$, foci at $(0,0)$, $(2\mu, 0)$, and with asymptotes forming angles $\pm(\pi/2 + a)$ with the real axis, so that Γ remains in the sector (1.2) of analyticity of F. After parameterizing (1.3), the function f is approximated by applying the truncated trapezoidal rule to the resulting integral along the real axis, i.e.,

$$f(t) = \frac{1}{2\pi i}\int_\Gamma e^{t\lambda}\, F(\lambda)\, d\lambda \approx \sum_{k=-K}^{K} w_k\, e^{t\lambda_k}\, F(\lambda_k), \tag{2.2}$$

with weights w_k and quadrature nodes λ_k given by

$$w_k = \frac{\tau}{2\pi i}\,\gamma'(k\tau)\ , \quad \lambda_k = \gamma(k\tau)$$

and $\tau > 0$ a suitable step length parameter. Different choices of contours Γ and parameterizations have been studied for the numerical inversion of sectorial Laplace transforms in the last years. The choice of a hyperbola has been studied in [14, 15, 17, 22, 9, 10, 28], and actually we follow here the approach in [14, 15]. The choice of Γ as a parabola has also been considered recently in [9, 10, 28]. Finally we refer to Talbot's method [26, 19, 27], which could also be used with the present algorithm; cf. [20, 16, 21, 13]. The good behavior of this quadrature formula to approximate (1.3) is due to the good properties of the trapezoidal rule when the integrand can be analytically extended to a horizontal strip around the real axis [24, 25]. We refer to [14, 15] for details and only give here the following error bound, which decays exponentially in the number of quadrature nodes.

THEOREM 2.1. *[15] Suppose that the Laplace transform F satisfies the sectorial condition* (1.2)*. For fixed $T > 0$, $\Lambda \geq 1$, $0 < a < \pi/2-\varphi$, and $K \geq 1$ there are positive numbers C_1, C_2, C, c depending on a and Λ (C depends additionally on T unless $\sigma < 0$ in* (1.2)*) such that the choice of parameters $\tau = C_1/K$ and $\mu = C_2K/(\Lambda t_0)$ yields a quadrature error in* (2.2) *bounded by*

$$|E_K(t)| \leq C\, t^{\nu-1}\left(\epsilon + e^{-cK}\right),$$

uniformly for t in intervals $[t_0, \Lambda t_0]$ with arbitrary $0 < t_0 \leq T/\Lambda$, where ν is the exponent of (1.2) *and ϵ is the precision in the evaluations of the Laplace transform F and the elementary operations in* (2.2)*.*

Hence, $K = O(\log \frac{1}{\varepsilon})$ quadrature points are sufficient to obtain an accuracy $O(\varepsilon)$ in the approximation of the contour integral. In practice, we choose $a \approx \frac{1}{2}(\frac{\pi}{2} - \varphi)$ and compute the values C_1 and C_2 following the optimization process described in [15].

3. The variable-step-size, fast and oblivious convolution algorithm.

3.1. Local reduction to differential equations. We want to approximate

$$u(t) = \int_0^t f(t-\tau)\, g(\tau)\, d\tau \tag{3.1}$$

on a sequence of times $0 < t_1 < \cdots < t_N$, spaced arbitrarily. For the moment we assume that g is a known function, though we will see later how to use the algorithm for solving integral and integro-differential equations. For a given t_n, we can insert the Laplace inversion formula in (3.1) and write

$$\int_0^{t_n} f(t_n-\tau)\, g(\tau)\, d\tau = \int_0^{t_n} \frac{1}{2\pi i}\int_\Gamma e^{(t_n-\tau)\lambda} F(\lambda)\, d\lambda\, g(\tau)\, d\tau\,. \tag{3.2}$$

The numerical inversion of the Laplace transform is performed very efficiently by the quadrature rule (2.2). In Section 2 we have seen that the same contour Γ used in this quadrature can be used to approximate f at different values of t, ranging over intervals of the form $[t_0, \Lambda t_0]$, for a given ratio $\Lambda \geq 1$. Since in (3.2) we need to approximate $f(t_n - \tau)$ for $t_n - \tau \in [0, t_n]$, we cannot use a unique contour Γ and we need to split the integral in (3.1) into several pieces. For suitable intermediate times $0 < t^- < t^+ < t_n$, with $(t_n - t^-)/(t_n - t^+) \leq \Lambda$, we select a suitable contour Γ for the time interval $[t_n - t^-, t_n - t^+]$ and approximate

$$\begin{aligned}
\int_{t^-}^{t^+} f(t_n - \tau) g(\tau)\, d\tau &= \int_{t^-}^{t^+} \frac{1}{2\pi i} \int_\Gamma e^{(t_n-\tau)\lambda} F(\lambda)\, d\lambda\, g(\tau)\, d\tau \\
&\approx \int_{t^-}^{t^+} \sum_{k=-K}^{K} w_k\, e^{(t_n-\tau)\lambda_k}\, F(\lambda_k)\, g(\tau)\, d\tau \\
&= \sum_{k=-K}^{K} w_k\, F(\lambda_k)\, e^{(t_n-t^+)\lambda_k} \int_{t^-}^{t^+} e^{(t^+-\tau)\lambda_k} g(\tau)\, d\tau \\
&= \sum_{k=-K}^{K} w_k\, F(\lambda_k)\, e^{(t_n-t^+)\lambda_k}\, y(t^+, t^-, \lambda_k),
\end{aligned} \tag{3.3}$$

where $y(t^+, t^-, \lambda_k)$ is the solution at t^+ to the linear inhomogeneous ODE

$$y' = \lambda_k y + g, \quad y(t^-) = 0, \qquad -K \leq k \leq K. \tag{3.4}$$

We now approximate $y(t^+, t^-, \lambda_k)$ by interpolating g linearly on each interval $[t_j, t_{j+1}]$ for $j = 0, \ldots, N-1$, and integrating exactly. (More elaborate integration methods could be used instead, cf. [7, 21], but for simplicity of presentation we will just consider this particular integration scheme.) We denote by $\bar{g}$ the interpolant of g and by $\bar{y}(t^+, t^-, \lambda_k)$ the resulting approximation to $y(t^+, t^-, \lambda_k)$, i.e., $\bar{y}(t^+, t^-, \lambda_k)$ is the exact solution at t^+ to

$$y' = \lambda_k y + \bar{g}, \quad y(t^-) = 0, \qquad -K \leq k \leq K. \tag{3.5}$$

Thus, we approximate

$$\int_{t^-}^{t^+} f(t_n - \tau)\, g(\tau)\, d\tau \approx \sum_{k=-K}^{K} w_k\, F(\lambda_k)\, e^{(t_n-t^+)\lambda_k}\, \bar{y}(t^+, t^-, \lambda_k)\,. \tag{3.6}$$

3.2. Filling the mosaic. The key to the algorithm is the way the splitting times $t^\pm$ for the integral in (3.1) are selected for every t_n with $1 \leq n \leq N$. This is done following the mosaic in the triangle $\{(t, \tau) : 0 \leq \tau \leq t \leq T\}$ shown in Fig. 3.1, where patches grow geometrically with increasing distance from the diagonal. For the moment, we fix a minimum size of the patches closest to the diagonal, corresponding to a minimum step size h_*. If along the vertical line at t_n joining 0 with the diagonal value t_n we have L different patches of the tessellation, then we obtain the values t_ℓ^- and t_ℓ^+ for $1 \leq \ell \leq L$ as the smallest and largest grid points, resp., within the ℓth patch along the vertical line at t_n. All those ℓs are collected in an index set J. In case a patch does not contain any grid points its value ℓ is not contained in J. The times $t_\ell^\pm = t_{n,\ell}^\pm$ depend on n, though for simplicity we omit this dependence in the notation. Each class of patches of the same size in the mosaic represents a distance

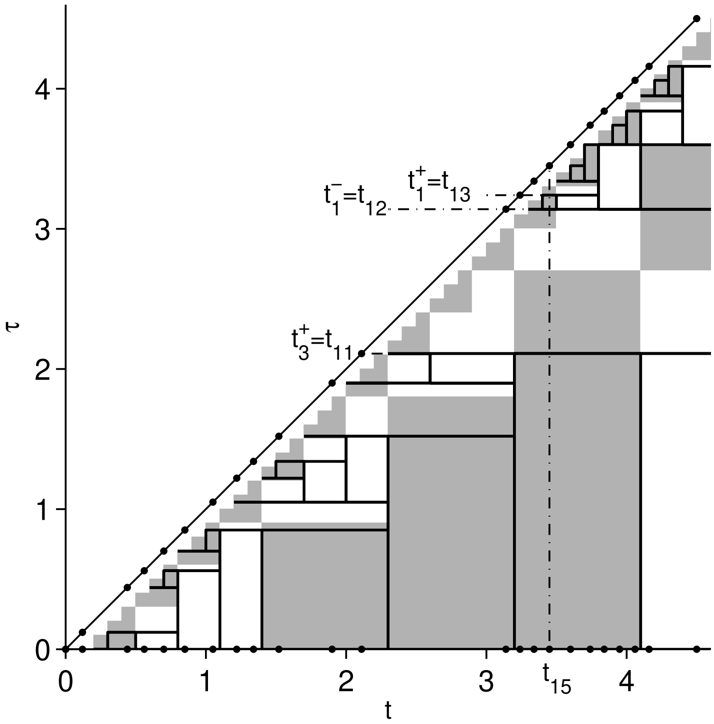

FIG. 3.1. *Mosaic in the triangle $\tau \le t$ for $B = 3$ with times t_j indicated by points. Each monochromatic rectangle fully enclosed by black lines corresponds to a solution value $y(t_\ell^+, t_\ell^-, \lambda)$ of a linear differential equation (3.4).*

class to the diagonal in the mosaic, and thus corresponds to a different approximation interval and a different contour to perform the inversion of the Laplace transform and to a different set of $2K + 1$ scalar differential equations. The approximation intervals for the values $t_n - t_\ell^\pm$ are of the form $I_\ell = [B^{\ell-1}h_*, B^{\ell+1}h_*]$, $2 \le \ell \le L$, so that the ratio Λ is B^2. Since we consider a non-equidistant sequence of times t_j, in this splitting there likely appear "gaps" in between the $t_{\ell+m}^+$ and t_ℓ^-, which in Figure 3.1 correspond to pairs of horizontal lines with any boundary of m patches in between them. For example, at the time point $t_{15} = 3.45$ we have $L = 3$, $J = \{1, 3\}$, $t_3^- = 0$, $t_3^+ = t_{11} = 2.11$, $t_1^- = t_{12} = 3.14$ and $t_1^+ = t_{13} = 3.24$.

We split (3.1) into $2|J| + 1$ parts

$$u(t_n) = \tilde{u}^{(0)}(t_n) + \sum_{\ell \in J} u^{(\ell)}(t_n) + \sum_{\ell \in J} \tilde{u}^{(\ell)}(t_n) \tag{3.7}$$

where

$$u^{(\ell)}(t_n) = \int_{t_\ell^-}^{t_\ell^+} f(t_n - \tau) g(\tau)\, d\tau \tag{3.8}$$

is computed using (3.3) and

$$\tilde{u}^{(0)}(t_n) = \int_{t_{n-1}}^{t_n} f(t_n - \tau) g(\tau)\, d\tau \quad \text{and} \quad \tilde{u}^{(\ell)}(t_n) = \int_{t_{\ell+m}^+}^{t_\ell^-} f(t_n - \tau) g(\tau)\, d\tau \tag{3.9}$$

correspond respectively to the step from t_{n-1} to t_n near the diagonal and to the gaps between $t^+_{\ell+m}$ and t^-_ℓ; see Fig. 3.1. These parts are computed by "direct steps" explained in the next subsection. Thus, for $t_{15} = 3.45$, $u(t_{15})$ is calculated using:

- one "ode step", $u^{(3)}$, from $t^-_3 = t_0 = 0$ to $t^+_3 = t_{11}$.
- one "direct step", $\tilde{u}^{(3)}$, from $t^+_3 = t_{11}$ to $t^-_1 = t_{12}$.
- one "ode step", $u^{(1)}$, from $t^-_1 = t_{12}$ to $t^+_1 = t_{13}$.
- one "direct step", $\tilde{u}^{(1)}$, from $t^+_1 = t_{13}$ to t_{14}.
- one "direct step", $\tilde{u}^{(0)}$, from t_{14} to t_{15}.

Translating the above splitting from the picture into a formal procedure, we thus proceed as follows: given a minimum step size h_* and a basis B, for each t_n we take L as small as possible so that we can represent $\lceil t_n/h_* \rceil = 2 + \sum_{\ell=1}^{L} b_\ell B^{\ell-1}$ with $b_\ell \in \{1, 2, \ldots, B\}$, where $\lceil x \rceil$ denotes the smallest integer greater than x. t^+_ℓ is the largest and t^-_ℓ is the smallest value in $\{t_j : j = 0, \ldots, n\}$, such that

$$h_* \sum_{k=\ell+1}^{L} b_k B^{k-1} \leq t^-_\ell \leq t^+_\ell \leq h_* \sum_{k=\ell}^{L} b_k B^{k-1}.$$

We remark that with this definition, there always exists some $j \in \{0, \ldots, n-1\}$ and some integer $m > 0$ such that $t^+_{\ell+m} = t_j$ and $t^-_\ell = t_{j+1}$. The $t^\pm_\ell$ are such that the integration limits of $u^{(\ell)}(t_n)$ fit into the approximation interval: $t_n - t^\pm_\ell \in I_\ell$. In Figure 3.1 we show how the solutions to the linear ODEs $\bar{y}(t^-_\ell, t^+_\ell, \lambda^\ell_k)$ fill the mosaic. In this example we have $B = 3$ and the t_j are given by the non-equidistant sequence of time points indicated both along the horizontal axis and the diagonal of the triangle. We can see here that, for instance, when times greater than $t = 3.2$ are reached, all the "past" from 0 to 2.11 is stored in the solution values $y(2.1, 0, \lambda^{(\ell)}_k)$, represented in Figure 3.1 by the tall dark rectangle with basis $[3.2, 4.1]$ for $\ell = 3$, and in the adjacent incomplete white rectangle to the right for $\ell = 4$. In our example the values $y(2.1, 0, \lambda^{(3)}_k)$ are used 8 times to evaluate $u(t)$ for $t \in [3.2, 4.1]$. The filling of the mosaic is done bottom up in the algorithm, advancing all the differential equations in every time step, so that the algorithm can forget all the past values of the function g, with the exception of those at $t^\pm_\ell$, which are needed for the direct steps described below. In addition to the current solution values $y(t, t^-_\ell, \lambda^{(\ell)}_k)$ of the differential equations at time $t = t_n$, also their values at a splitting point t^+_ℓ need to be stored until t^+_ℓ is increased at a later step. Actually the algorithms stores three copies of $y(t^+_\ell, t^-_\ell, \lambda^{(\ell)}_k)$. Pseudocodes for the organization of the decomposition are given in the appendix.

3.3. Direct steps. The gaps $[t^+_{\ell+m}, t^-_\ell]$ between the enclosed blocks in Figure 3.1 are bridged using the values $\tilde{u}^{(\ell)}(t_n)$ whose computation we describe next. These direct steps compute

$$\tilde{u}^{(\ell)}(t_n) = \int_{t^+_{\ell+m}}^{t^-_\ell} f(t_n - \tau) g(\tau)\, d\tau = \int_{t_j}^{t_{j+1}} f(t_n - \tau) g(\tau)\, d\tau \tag{3.10}$$

for some $j \in \{0, 1, \ldots, N-1\}$. On the interval $[t_j, t_{j+1}]$ we approximate $g(t)$ by a linear function:

$$g(t) \approx \bar{g}(t) = g_j + \frac{g_{j+1} - g_j}{h_{j+1}}(t - t_j), \qquad h_{j+1} = t_{j+1} - t_j,$$

with $g_j = g(t_j)$, $j = 0, 1, \dots, N$. (Here again, the approach would extend to polynomials of higher degree.) Extending $\bar{g}(t)$ to $[0, t_n]$ we split (3.10) in two terms

$$\int_{t_j}^{t_{j+1}} f(t_n - \tau)\, \bar{g}(\tau)\, d\tau = \int_{t_j}^{t_n} f(t_n - \tau)\, \bar{g}(\tau)\, d\tau - \int_{t_{j+1}}^{t_n} f(t_n - \tau)\, \bar{g}(\tau)\, d\tau$$
$$= \mathcal{L}^{-1}[F \cdot \mathcal{L}g(\cdot + t_j)](t_n - t_j) - \mathcal{L}^{-1}[F \cdot \mathcal{L}g(\cdot + t_{j+1})](t_n - t_{j+1})$$
$$= \mathcal{L}^{-1}\left[F_1 g_j + F_2 \frac{g_{j+1} - g_j}{h_{j+1}}\right](t_n - t_j) - \mathcal{L}^{-1}\left[F_1\, g_{j+1} + F_2 \frac{g_{j+1} - g_j}{h_{j+1}}\right](t_n - t_{j+1}),$$

where $F_1(s) = F(s)/s$ and $F_2(s) = F(s)/s^2$. We approximate the inverse Laplace transforms

$$f_1(t) = \big(\mathcal{L}^{-1} F_1\big)(t), \qquad f_2(t) = \big(\mathcal{L}^{-1} F_2\big)(t)$$

at $t = t_n - t_j$ and $t = t_n - t_{j+1}$ using the numerical integration of the Laplace inversion formula along the integration contours corresponding to the approximation intervals I_{ℓ_1} and I_{ℓ_2} such that $t_n - t_{j+1} \in I_{\ell_2}$ and $t_n - t_j \in I_{\ell_1}$. The result of the direct step is then calculated forming linear combinations

$$\begin{aligned}\int_{t_j}^{t_{j+1}} f(t_n - \tau)\, \bar{g}(\tau)\, d\tau = f_1(t_n - t_j)\, g_j + f_2(t_n - t_j)\, \frac{g_{j+1} - g_j}{h_{j+1}} \\ - f_1(t_n - t_{j+1})\, g_{j+1} - f_2(t_n - t_{j+1})\, \frac{g_{j+1} - g_j}{h_{j+1}}\ .\end{aligned} \tag{3.11}$$

This is also used for the terms closest to the diagonal, $t_{j+1} = t_n$, where we note in addition that $f_1(0) = f_2(0) = 0$.

3.4. Complexity. Given an arbitrary sequence of time points $0 < t_1 < \ldots < t_N = T$ with the minimum step size $h_* = \min_j (t_{j+1} - t_j)$, the above algorithm computes

$$\int_0^t f(t - \tau)\, \bar{g}(\tau)\, d\tau, \qquad \text{for} \quad t = t_1, \dots, t_N, \tag{3.12}$$

(with $\bar{g}$ the piecewise linear interpolant of g) up to an error ε using

- $L = O(\log \frac{T}{h_*})$ hyperbolas with
- $2K + 1 = O(\log \frac{1}{\varepsilon})$ quadrature points on each hyperbola.

The algorithm thus requires $(2K + 1)L$ evaluations of the Laplace transform $F(s)$ at the quadrature points and solves $(2K + 1)L$ differential equations $y' = \lambda_k^{(\ell)} y + \bar{g}$. As the algorithm proceeds, only three solution values need to be stored for each of these differential equations. In addition, at most $2L$ values of g need to be kept in memory for the direct steps. In total, the active memory requirements are $O(LK) = O(\log \frac{T}{h_*} \log \frac{1}{\varepsilon})$ vectors of the dimension of g. The total operation count is $O(NLK) = O(N \log \frac{T}{h_*} \log \frac{1}{\varepsilon})$. For the variable-step-size algorithm we thus obtain the complexity characteristics as stated in the introduction for the fast and oblivious fixed-step-size algorithm, with $\log N$ now replaced by $\log \frac{T}{h_*}$.

3.5. Adaptivity based on controlling the interpolation error. There are two sources of error in the algorithm. The first one is the discretization of the contour integral, which is well controlled. The second one is the piecewise linear interpolation of g by $\bar{g}$. Ignoring the error from discretizing the contour integrals, the algorithm thus computes (3.12) instead of (1.1). We control the error in g, which is bounded by

$$\|\bar{g}(t) - g(t)\| \leq \tfrac{1}{8}\, h_n^2 \max_{t_{n-1} \leq \tau \leq t_n} \|g''(\tau)\| \qquad \text{for} \quad t_{n-1} \leq t \leq t_n.$$

Given a tolerance *Tol*, we propose the new step-size h_{n+1} according to the criterion

$$Ch_{n+1}^2 \gamma_n'' = 0.8 \cdot \mathit{Tol}\,, \tag{3.13}$$

where $\gamma_n'' = \|\widetilde{g}''(t_n)\|$ with the quadratic interpolant $\widetilde{g}$ to g at t_{n-2}, t_{n-1}, t_n. The constant C is chosen as $C \approx \frac{1}{8}\int_0^T |f(t)|\, dt$. Additionally the step-size is restricted to fulfill $1/2h_n < h_{n+1} < 2h_n$. The proposed step size h_{n+1} is tested by

$$Ch_{n+1}^2 \gamma_{n+1}'' \leq \mathit{Tol}\,, \tag{3.14}$$

where the new value $g(t_n + h_{n+1})$ is used in the computation of γ_{n+1}''. If this condition is satisfied, then h_{n+1} is accepted and we set $t_{n+1} = t_n + h_{n+1}$, else we repeat the test with a reduced step h_{n+1} determined from (3.13) with γ_{n+1}'' in place of γ_n''. If necessary, this procedure is repeated until (3.14) is satisfied.

In the following sections we give three examples to show the performance of the algorithm, using also other strategies for controlling the step size in the time integration of integro-differential equations. However, we point out that the above fast algorithm is independent of the particular step size selection strategy. The step size control is just the way we generate the sequence of time points. The minimum step size need not be specified *a priori*, and at time t_n the future time points $t_{n+1}, t_{n+2}, \dots$ need not yet be known in the algorithm.

For the three examples provided, we ran the algorithm with basis $B = 5$, which gives approximation intervals of the type $[t_0, 25t_0]$.

4. A blow-up example originating from a Schrödinger equation with concentrated nonlinearity. Adami & Teta [1, 2] consider a nonlinear Schrödinger equation with nonlinearity concentrated at $x = 0$,

$$i\frac{\partial \psi}{\partial t} = -\Delta\psi + \gamma|\psi|^{2\sigma}\psi \cdot \delta_{x=0}\,, \qquad x \in \mathbb{R},\ t > 0, \tag{4.1}$$

with initial data $\psi(x, 0) = \psi_0(x)$ for $x \in \mathbb{R}$. The equation can be given a rigorous formulation as an integral equation. With Duhamel's principle the wave function $\psi(x, t)$ can be expressed as the sum of $\big(U(t)\psi_0\big)(x)$, that is, the solution at position x and time t of the free Schrödinger equation with initial data ψ_0, and of a convolution with $|z|^{2\sigma}z$, where $z(\tau) := \psi(0, \tau)$ for $0 \leq \tau \leq t$. The function z is the solution of a nonlinear, complex Abel-type integral equation

$$z(t) + \gamma\,\frac{\sqrt{i}}{2}\int_0^t \frac{1}{\sqrt{\pi(t-\tau)}}\,|z(\tau)|^{2\sigma} z(\tau)\, d\tau = a(t)\,, \qquad t \geq 0, \tag{4.2}$$

where $a(t) = \big(U(t)\psi_0\big)(0)$ is the solution at $x = 0$ of the free Schrödinger equation. We present the results of numerical experiments in a situation where the solution is known to have finite-time blow-up. We choose $\sigma = 1$, and

$$a(t) = \frac{1}{\pi^{1/4}}\frac{1}{\sqrt{1 + i2t}}\,,$$

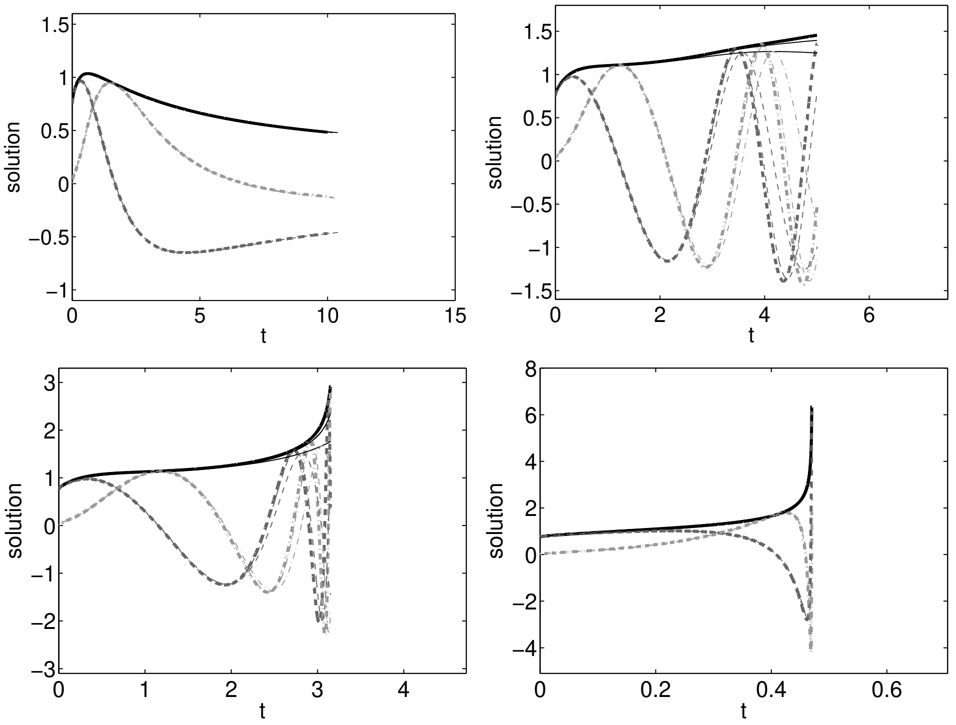

FIG. 4.1. *Real and imaginary part (light and dark gray) and modulus (black) of the solution z for $\gamma = -2, -2.05, -2.06 - 2.5$ and for different tolerances.*

which corresponds to a Gaussian wave-packet $\psi_0(x) = \pi^{-1/4}e^{-x^2/2}$ as initial data.

For four different values of γ, $\gamma = -2, -2.05, -2.06, -2.5$ the evolution of the solution is shown in Fig 4.1 for different tolerances. The thick lines correspond to solutions obtained with a tolerance of 10^{-7}. The other two lines correspond to tolerances of 10^{-3} and $2 \cdot 10^{-4}$. Whereas for $\gamma = -2$, -2.5 one cannot distinguish the different tolerances in the plot, for $\gamma = -2.05$, -2.06 it is clearly visible, that choosing a too low tolerance will produce a wrong result.

In Fig. 4.2 the step size is plotted over t. Clearly the adaptivity pays off to resolve the blow-up. In case $\gamma = -2.5$ we observe for a tolerance of 10^{-7} step-sizes ranging from 10^{-8} to 10^{-4}.

Fig. 4.3 displays the absolute error at the final time, obtained with tolerances $1 \cdot 10^{-3}, 5 \cdot 10^{-4}, 2 \cdot 10^{-4}, 1 \cdot 10^{-4}, 5 \cdot 10^{-5}, 2 \cdot 10^{-5}, 1 \cdot 10^{-5}, 5 \cdot 10^{-6}, 2 \cdot 10^{-6}, 1 \cdot 10^{-6}$ versus the total number of steps. The error is measured against a reference solution obtained with tolerance 10^{-7}. The numerical inversion of the Laplace transform is performed as explained in Section 2, with $a = 0.8$, $d = 0.7$, and $K = 50$, which gives $C_1 = 6.567$ and $C_2 = 0.066$ (see Theorem 2.1). For larger tolerances, good results can be obtained with a smaller K, say $K = 25$. Taking $K = 40$, we only get small oscillations in the stepsize for the smallest tolerance, `tol` $= 10^{-7}$, in Figure 4.2, and no visible changes for less stringent tolerances.

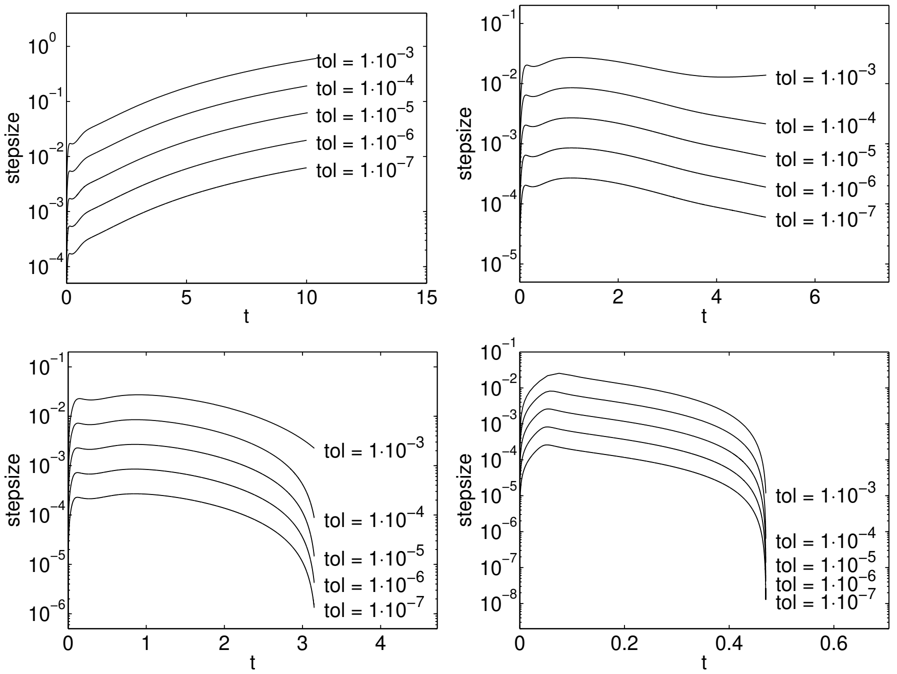

Fig. 4.2. *Evolution of the step-size for* $\gamma = -2, -2.05, -2.06, -2.5$ *and different tolerances*

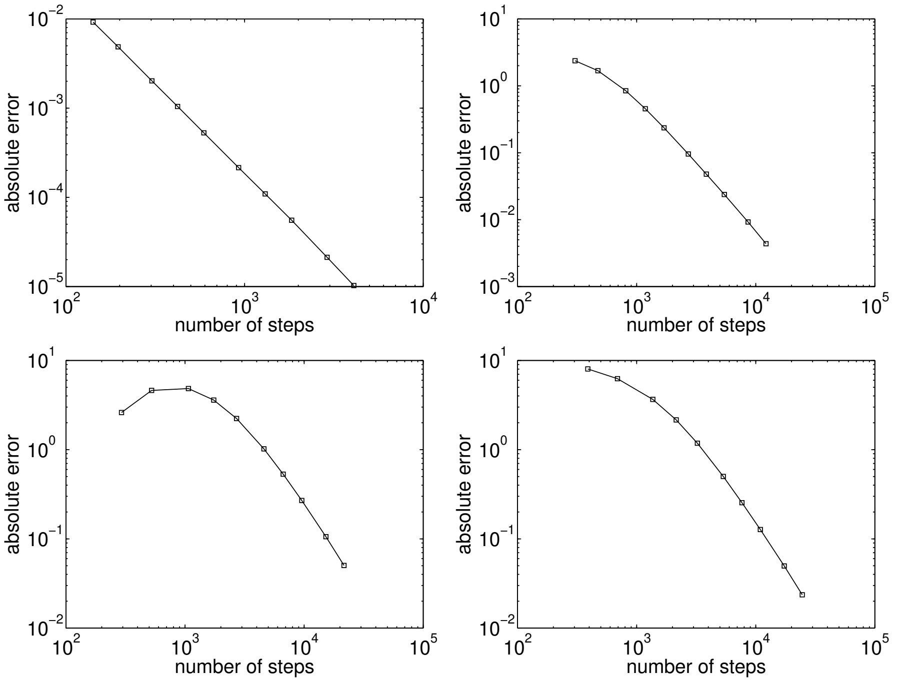

Fig. 4.3. *Error versus the number of steps, at* $t = 10$ *for* $\gamma = -2$*, at* $t = 5$ *for* $\gamma = -2.05$*, at* $t = 3.15$ *for* $\gamma = -2.06$*, and at* $t = 0.47$ *for* $\gamma = -2.5$

5. Chemical reaction kinetics with inhibited diffusion. We consider three molecular species A, B and C, reacting as

$$\begin{aligned} A+B &\xrightarrow{k_1} C \\ C &\xrightarrow{k_2} A+B \\ C &\xrightarrow{k_3} A+P\ , \end{aligned} \tag{5.1}$$

P being the resulting product. The diffusion of each of the species is anomalous. So we obtain a reaction diffusion equation with a memory term. A model like this, with three species and three reactions was considered in [6]. However we have chosen to follow [29] and to associate a memory with the reaction term. Thus to model this process the following system of integro-differential equations is considered:

$$\begin{aligned} \dot u_1 =& \partial_t \partial_t^{-\alpha}\big(K\Delta u_1 - k_1 u_1 u_2 + (k_2+k_3)u_3\big) \\ \dot u_2 =& \partial_t \partial_t^{-\alpha}\big(K\Delta u_2 - k_1 u_1 u_2 + k_2 u_3\big) \\ \dot u_3 =& \partial_t \partial_t^{-\alpha}\big(K\Delta u_3 + k_1 u_1 u_2 - (k_2+k_3)u_3\big), \end{aligned} \tag{5.2}$$

where $\Delta = \partial_{xx}$ is the 1D Laplacian with periodic boundary condition on $[-5,5]$ and $\partial_t^{-\alpha}$ denotes the fractional integral of order $0<\alpha<1$, given by the Riemann Liouville operator

$$\partial_t^{-\alpha} g(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} g(\tau)\, d\tau, \tag{5.3}$$

for $0<\alpha<1$. Integrating in time system (5.2), we get the integro-differential equation

$$\begin{aligned} u_1(t)-u_1(0) =& \partial_t^{-\alpha}\big[K\Delta u_1(t) - k_1 u_1(t)u_2(t) + (k_2+k_3)u_3(t)\big] \\ u_2(t)-u_2(0) =& \partial_t^{-\alpha}\big[K\Delta u_2(t) - k_1 u_1(t)u_2(t) + k_2 u_3(t)\big] \\ u_3(t)-u_3(0) =& \partial_t^{-\alpha}\big[K\Delta u_3(t) + k_1 u_1(t)u_2(t) - (k_2+k_3)u_3(t)\big]. \end{aligned} \tag{5.4}$$

In this situation we have in the convolution terms the weakly singular kernel $f(t) = t^{\alpha-1}/\Gamma(\alpha)$, with Laplace transform $F(s) = s^{-\alpha}$. We approximate the solution to (5.4) by using an adaptive strategy similar to the one explained in Section 3.5 but replacing criterion (3.13) by

$$Ch_{n+1}^2 \gamma_n' = 0.8 \cdot Tol,$$

and the test (3.14) by

$$Ch_{n+1}^2 \gamma_{n+1}' \le Tol\,,$$

where $\gamma_n' = \|\widetilde{g}'(t_n)\|$, with $\widetilde{g}$ the linear interpolant of g at t_{n-1} and t_n. Our choice for the different parameters is $K=0.5$, $k_1=1$, $k_2=2$ and $k_3=3$, and we integrate up to $T=30$. We fix $\alpha=0.5$ and consider smoothed step-like functions as the initial data.

Setting $\mathbf{u} = [u_1,u_2,u_3]^T$, I_3 the 3×3 identity matrix, and following the notation introduced in Section 3.3 for the direct steps of the algorithm, the discrete equation approximating (5.4) is

$$
\begin{aligned}
&\Big(I_3 \otimes I - \frac{f_2(h_n)}{h_n}\,(K\,I_3 \otimes S + R)\Big)\mathbf{u}^n \\
&= \Big(f_1(h_n) - \frac{f_2(h_n)}{h_n}\Big)\big(K\,I_3 \otimes S + R\big)\mathbf{u}^{n-1} + \; k_1 f_1(h_n)\,\mathbf{e} \otimes u_1^{n-1} u_2^{n-1} \\
&+ \; \frac{1}{\Gamma(\alpha)} \int_0^{t_{n-1}} \frac{g(\bar{\mathbf{u}}(\tau))}{(t-\tau)^{1-\alpha}}\, d\tau + \mathbf{u}_0,
\end{aligned}
\tag{5.5}
$$

where $\bar{\mathbf{u}}$ denotes the piecewise linear interpolant of $\mathbf{u}$ at times $t_0, t_1, \ldots, t_n$ and, for M nodes in the spatial discretization and $\mathbf{v}$ a column vector of length $3M$, we define

$$
g(\mathbf{v}) = \big(K\,I_3 \otimes S + R\big)\mathbf{v} + k_1\big(\mathbf{e} \otimes v_1 v_2\big), \qquad \mathbf{e} = [-1, -1, 1]^T,
$$

with S the second order finite difference approximation to ∂_{xx} with periodic boundary conditions and R the $3M \times 3M$ matrix

$$
R = \begin{pmatrix} 0 & 0 & (k_2 + k_3) I_M \\ 0 & 0 & k_2 I_M \\ 0 & 0 & -(k_2 + k_3) I_M \end{pmatrix},
$$

where I_M is the $M \times M$ identity matrix.

Fig. 5.1 shows numerical results and performance characteristics of the algorithm. Here we inverted the Laplace transform taking $a = 1$, $d = 0.5$, and $K = 40$ quadrature nodes on the hyperbolas, giving $C_1 = 6.036$ and $C_2 = 0.0739$. Again, less stringent accuracy requirements demand fewer quadrature nodes.

6. Dynamic fractional order viscoelasticity.

6.1. Model. The fractional order linear viscoelastic constitutive equation for the stress $\boldsymbol{\sigma}$ considered in [4, 3] reads

$$
\boldsymbol{\sigma} = \boldsymbol{\sigma}_0(t) - \gamma \int_0^t f(t-\tau)\boldsymbol{\sigma}_0(\tau)\, d\tau, \tag{6.1}
$$

with stress tensor $\boldsymbol{\sigma}_0$ and strain tensor $\boldsymbol{\epsilon}$ given by

$$
\boldsymbol{\sigma}_0(t) = 2\mu\boldsymbol{\epsilon}(t) + \lambda \mathrm{tr}(\boldsymbol{\epsilon}(t))\boldsymbol{I}\,; \quad \boldsymbol{\epsilon}(t) = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right), \tag{6.2}
$$

where μ and λ are the Lamé constants and $0 < \gamma < 1$ is a given parameter. We refer to [4] for more details about the model.

The basic equations for the displacement field $\boldsymbol{u}$ are

$$
\begin{aligned}
&\rho\ddot{\boldsymbol{u}}(\boldsymbol{x}, t) - \nabla \cdot \boldsymbol{\sigma}_0(\boldsymbol{u}; \boldsymbol{x}, t) \\
&+ \gamma \int_0^t f(t-\tau)\nabla \cdot \boldsymbol{\sigma}_0(\boldsymbol{u}; \boldsymbol{x}, \tau)\, d\tau = 0 \quad \text{for } \boldsymbol{x} \in \Omega;\ t \geq 0 \\
&\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \Omega \\
&\dot{\boldsymbol{u}}(\boldsymbol{x}, 0) = \boldsymbol{v}_0(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \Omega \\
&\boldsymbol{u}(\boldsymbol{x}, t) = 0 \quad \text{for } \boldsymbol{x} \in \Gamma_D;\ t \geq 0 \\
&\boldsymbol{\sigma}_0(\boldsymbol{u}; \boldsymbol{x}, t) \cdot \boldsymbol{n}(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x}, t) \quad \text{for} \;\; \boldsymbol{x} \in \Gamma_N;\ t \geq 0].
\end{aligned}
\tag{6.3}
$$

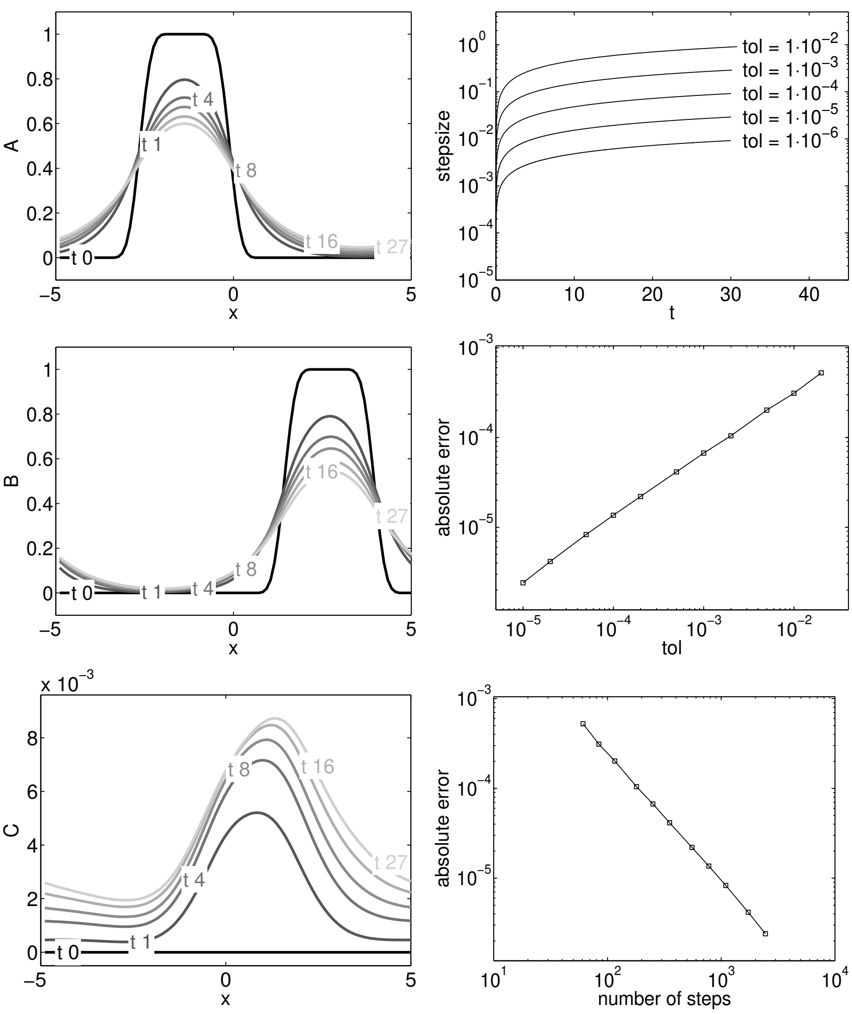

FIG. 5.1. *Left: Solutions for the three species A, B, C, at different times (lighter lines for larger times). Right: Step size versus time t, error versus tolerance, and error versus number of steps. All for $T = 30$, $K = 0.5$, $k_1 = 1$, $k_2 = 2$, and $k_3 = 3$.*

Γ_D denotes the Dirichlet boundary of the Ω and Γ_N the Neumann boundary, where the boundary force $\boldsymbol{b}$ is applied.

On the Sobolev space $V := \{\boldsymbol{v} \in H^1(\Omega)^2 : \boldsymbol{v} = \mathbf{0} \text{ on } \Gamma_D\}$ the variational formulation reads as follows: Find $\boldsymbol{u}(t) \in V$ such that

$$\begin{aligned}
&\int_\Omega \rho \ddot{\boldsymbol{u}}(\boldsymbol{x},t)\cdot\boldsymbol{\psi}(\boldsymbol{x})d\boldsymbol{x} + \int_\Omega \boldsymbol{\epsilon}(\boldsymbol{u};\boldsymbol{x},t) : C\boldsymbol{\epsilon}(\boldsymbol{\psi}(\boldsymbol{x}))d\boldsymbol{x} - \int_{\Gamma_N} \boldsymbol{b}(\boldsymbol{x},t)\cdot\boldsymbol{\psi}(\boldsymbol{x})d\sigma(\boldsymbol{x}) \\
&-\gamma\int_0^t f(t-\tau)\left(\int_\Omega \boldsymbol{\epsilon}(\boldsymbol{u};\boldsymbol{x},\tau) : C\boldsymbol{\epsilon}(\boldsymbol{\psi}(\boldsymbol{x}))d\boldsymbol{x} - \int_{\Gamma_N}\boldsymbol{b}(\boldsymbol{x},\tau)\cdot\boldsymbol{\psi}(\boldsymbol{x})d\sigma(\boldsymbol{x})\right)\, d\tau \qquad (6.4)\\
&= 0\,, \qquad \forall \boldsymbol{\psi} \in V \\
&\boldsymbol{u}(0) = \boldsymbol{u}_0\ ; \quad \dot{\boldsymbol{u}}(0) = \boldsymbol{v}_0(\boldsymbol{x}),
\end{aligned}$$

where the tensor product is given by

$$\boldsymbol{\epsilon}(\boldsymbol{u}) : C\boldsymbol{\epsilon}(\boldsymbol{\psi}) = \sum_{i,j=1}^{2} 2\mu_0\epsilon_{ij}(\boldsymbol{u})\epsilon_{ij}(\boldsymbol{\psi}) + \lambda_0\epsilon_{jj}(\boldsymbol{u})\epsilon_{ii}(\boldsymbol{\psi}) \ .$$

Equation (6.4) is discretized in space using linear finite elements. The mesh is generated using Triangle [23] and the assembly of the mass and stiffness matrices M and A and the boundary force vector b is done following [5]. In contrast to [5] we have chosen not to use Lagrange multipliers to enforce the Dirichlet data, but to incorporate the Dirichlet data directly. Thus (6.4) results in the abstract integro-differential equation

$$\begin{aligned} M\ddot{u}(t) + Au(t) - b(t) &= \gamma \int_0^t f(t-\tau)(Au(\tau) - b(\tau)) \\ u(0) = u_0 \ ; \qquad & \dot{u}(0) = v_0. \end{aligned}$$

The kernel f in (6.1) is given by

$$f(t) = -\frac{d}{dt}E_\alpha\left(-t^\alpha\right), \qquad 0 < \alpha < 1, \tag{6.5}$$

where E_α denotes the Mittag-Leffler function of order α, defined by

$$E_\alpha(x) = \sum_{j=0}^{\infty} \frac{x^j}{\Gamma(1+\alpha j)}.$$

The Laplace transform F of f is given by

$$F(s) = \frac{1}{1+s^\alpha}.$$

6.2. Adaptive step size control. The discretization of the fractional order viscoelastic problem yields a Volterra integro-differential equation of second order of convolution type,

$$M\ddot{u}(t) + Au(t) = \gamma \int_0^t f(t-\tau)(Au(\tau - b(\tau))\, d\tau + b(t) =: c(t).$$

This is equivalent to

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & M^{-1} \\ -A & 0 \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix} + \begin{pmatrix} 0 \\ c \end{pmatrix}. \tag{6.6}$$

Applying the transformations $u \to \hat{u} = M^{1/2}u$, $v \to \hat{v} = M^{-1/2}v$, $A \to \hat{A} = M^{-1/2}AM^{-1/2}$ and $c \to \hat{c} = M^{-1/2}c$, we get

$$\begin{pmatrix} \dot{\hat{u}} \\ \dot{\hat{v}} \end{pmatrix} = \begin{pmatrix} \hat{v} \\ -A\hat{u} + \hat{c} \end{pmatrix}.$$

In what follows we drop the $\hat{}s$. The time discretization is done using the Störmer–Verlet scheme, which is explicit and symmetric and has good properties for the part

$\ddot{u}(t) = -Au(t) + b(t)$ (without the memory term). The Verlet scheme for the above equation reads

$$\begin{aligned} v_{n+1/2} &= v_n + \frac{h}{2}(-Au_n + c_n) \\ u_{n+1} &= u_n + hv_{n+1/2} \\ v_{n+1} &= v_{n+1/2} + \frac{h}{2}(-Au_{n+1} + c_{n+1}), \end{aligned} \tag{6.7}$$

where $c_n \approx c(t_n)$ is computed using the adaptive convolution algorithm explained in Section 3. Note that u_n is already known before we evaluate the c_n and thus scheme is explicit. In order not to lose the good properties of the Störmer-Verlet scheme a special step-size control is used, following [11, 12] . For the *integrating* controller we fix an accuracy parameter ε (which can roughly be viewed as the square root of a local error tolerance). Our step-size density function should control $\ddot{v}$ and $\ddot{u}$, therefore we take

$$\begin{aligned} \sigma(u,v,t) &= \tilde{\sigma}(u,v,t)^{-1/4} = \left(\|\ddot{v}\|^2 + \|\ddot{u}\|_A^2\right)^{-1/4} \\ &= \left((-Av+\dot{c})^T(-Av+\dot{c}) + (-Au+c)^T A(-Au+c)\right)^{-1/4}. \end{aligned} \tag{6.8}$$

Assuming that A is symmetric, the partial derivatives of $\tilde{\sigma}$ are

$$\begin{aligned} \tilde{\sigma}_u(u,v,t) &= 2(Au-c)^T AA \\ \tilde{\sigma}_v(u,v,t) &= 2(Av-\dot{c})^T A \\ \tilde{\sigma}_t(u,v,t) &= -2(Av-\dot{c})^T\ddot{c} - 2(Au-c)^T A\dot{c}. \end{aligned}$$

With this choice the step-size becomes approximately proportional to $1/\sqrt{\|\ddot{u}\| + c\|\ddot{v}\|}$. We have to take the A norm of u so that $\|\ddot{v}\|^2$ and $\|\ddot{u}\|_A^2$ are in the same units. We use the integrating controller of [12],[11, (VIII.3.2)] and set

$$G(u,v,t) = -\frac{1}{\sigma(u,v,t)}\nabla\sigma(u,v,t)^T \begin{pmatrix} v \\ -Au + c(t) \\ 1 \end{pmatrix} = -\frac{1}{4\tilde{\sigma}}\left(2(Av-\dot{c})^T\ddot{c}\right), \tag{6.9}$$

where for an evaluation, $\dot{c}$ and $\ddot{c}$ at t_n are approximated by divided differences using c_n, c_{n-1}, c_{n-2} (set to zero for negative subscripts).

Transforming back to "non-hat" quantities and again assuming that M and A are symmetric one obtains

$$\tilde{\sigma}(u,v,t) = (AM^{-1}v-\dot{c})^T M^{-1}(AM^{-1}v-\dot{c}) + (Au-c)^T M^{-1}AM^{-1}(Au-c) \tag{6.10}$$

and

$$G = -\frac{1}{4\tilde{\sigma}}\left(2(AM^{-1}v-\dot{c})^T M^{-1}\ddot{c}\right). \tag{6.11}$$

With an accuracy parameter ε, and starting with $c_{-1} = c_{-2} = c_0 = b_0$, $z_{-1/2} =$

$1/\sigma(u_0, v_0, t_0) - \varepsilon G(u_0, v_0, t_0)/2$, we compute for $n = 0, \ldots$

$$
\begin{aligned}
z_{n+1/2} &= z_{n-1/2} + \varepsilon G(u_n, v_n, t_n) \\
h_{n+1/2} &= \frac{\varepsilon}{z_{n+1/2}} \\
t_{n+1} &= t_n + h_{n+1/2} \\
v_{n+1/2} &= v_n + \frac{h_{n+1/2}}{2}(-Au_n + c_n) \\
u_{n+1} &= u_n + h_{n+1/2} M^{-1} v_{n+1/2} \\
c_{n+1} &= \gamma \left(\frac{f_2(h_{n+1/2})}{h_{n+1/2}}(Au_{n+1} - b_{n+1}) + \left(f_1(h_{n+1/2}) - \frac{f_2(h_{n+1/2})}{h_{n+1/2}} \right)(Au_n - b_n) \right. \\
&\quad \left. + \int_0^{t_n} f(t_{n+1} - \tau)(Au(\tau) + b(\tau))d\tau \right) + b_{n+1} \\
v_{n+1} &= v_{n+1/2} + \frac{h_{n+1/2}}{2}(-Au_{n+1} + c_{n+1}),
\end{aligned}
\tag{6.12}
$$

where $h_{n+1/2}$ is the new step-size proposed by the integrating controller.

6.3. Numerical example. In the example the domain Ω has the form of a cantilever as shown in Fig 6.1. The Dirichlet boundary Γ_D – the left vertical boundary of Ω – is indicated by squares. The time-dependent boundary force $\boldsymbol{b}$ is applied to the right vertical boundary Γ_{N_1} of Ω – indicated by circles in Fig. 6.1. On Γ_{N_2} (the upper and lower part of the boundary of Ω) homogenous Neumann boundary condition is assumed.

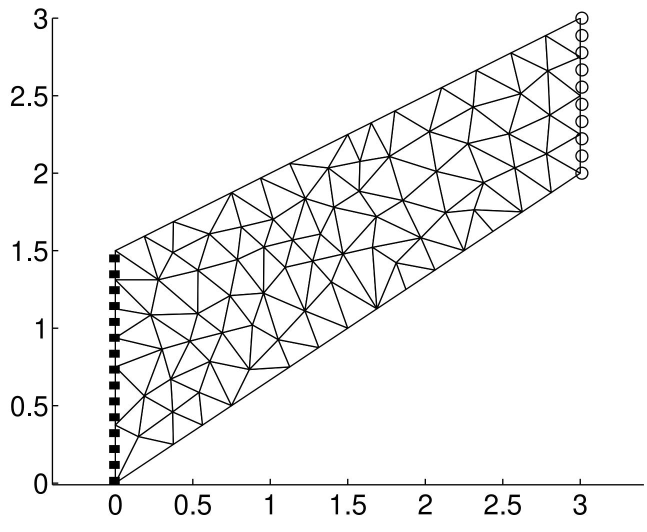

FIG. 6.1. *Cantilever with finite element mesh.* Γ_D *is indicated by small squares and* Γ_{N_1} *by circles.*

As the initial data $\boldsymbol{u}(\boldsymbol{x}, 0)$, $\boldsymbol{v}(\boldsymbol{x}, 0)$ we take the lowest mode of the static semi-discrete problem corresponding to the first order equation (6.6), i.e., the eigenvector of the matrix there corresponding to the smallest eigenvalue. The boundary force is

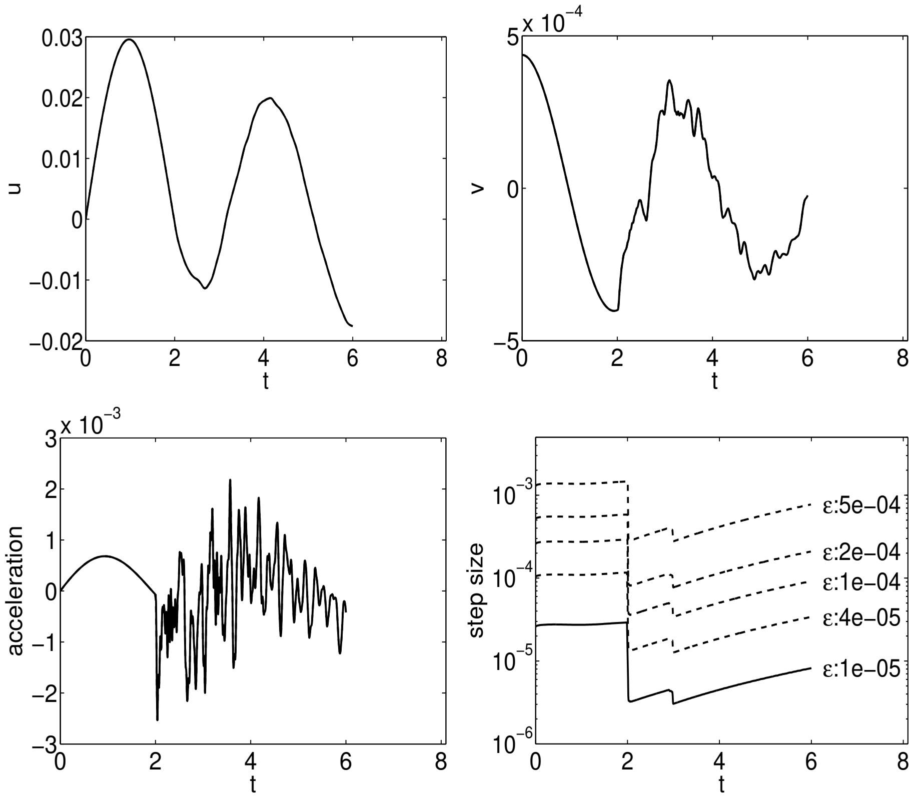

FIG. 6.2. $t = 6$. *Displacement, velocity, and acceleration recorded for one fixed spatial node. Also step size versus time for different* ε.

given by

$$\boldsymbol{b}(\boldsymbol{x}, t) = \begin{cases} 20e^{1/((2t-5)^8-1)}(1,1)^T & \text{for } 2 < t < 3; \boldsymbol{x} \in \Gamma_{N_1} \\ 0 & \text{else.} \end{cases}$$

In the numerical example the order of the Mittag-Leffler function is $\alpha = 1/2$. We set the density $\rho = 1$ and $\gamma = 0.3$. Youngs modulus and Poisson ratio are $E = 200$, $\nu = 0.3$. Equivalently the Lamé constants are $\mu = 76.9$, $\lambda = 115.4$. For the numerical inversion of the Laplace transform we took here $a = 0.8$, $d = 0.7$ and $K = 35$ quadrature nodes, giving $C_1 = 6.225$ and $C_2 = 0.097$.

The evolution from $t = 0$ to $t = 6$ of the horizontal component (x coordinate) of the displacement field $\boldsymbol{u}$, the velocity $\dot{\boldsymbol{u}}$, and the acceleration $\ddot{\boldsymbol{u}}$ recorded at the upper left corner of the cantilever is shown in Fig. 6.2. At $t = 2$ when the boundary force is applied, an abrupt change in the velocity and strong oscillations in the acceleration is observed. Furthermore Fig. 6.2 shows the evolution of the step-size $h_{n+1/2}$ for five different precision parameters $\varepsilon = 1 \cdot 10^{-5}, 4 \cdot 10^{-5}, 1 \cdot 10^{-4}, 2 \cdot 10^{-4}, 5 \cdot 10^{-4}$. The integrating controller reduces the step-size by roughly a factor of ten at $t = 2$.

The relative error at $t = 6$ in energy norm $\|u\|_A + \|v\|_M$ is calculated with respect to a reference solution obtained with $\varepsilon = 1 \cdot 10^{-5}$. The left plot in Fig 6.3 shows the relative error versus the number of steps. The different solutions were obtained for $\varepsilon = 4 \cdot 10^{-5}, 5 \cdot 10^{-5}, 7 \cdot 10^{-5}, 1 \cdot 10^{-4}, 1.5 \cdot 10^{-4}, 2 \cdot 10^{-4}, 3 \cdot 10^{-4}, 4 \cdot 10^{-4}, 5 \cdot 10^{-4}$.

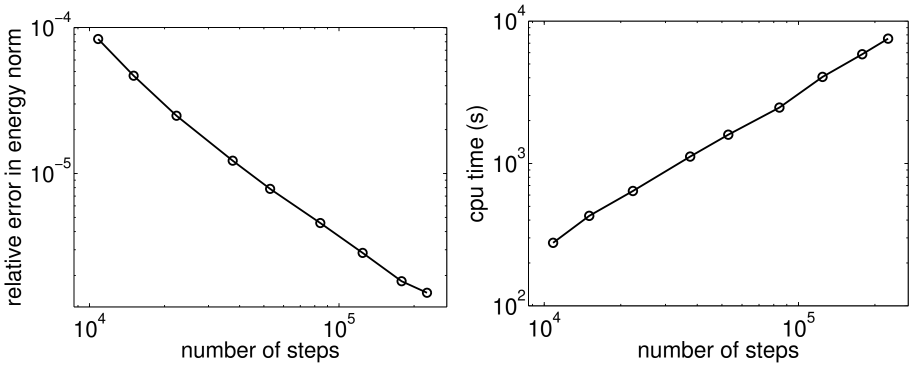

FIG. 6.3. *Error versus number of steps and error versus cpu time, for $t = 6$.*

The right plot in Fig 6.3 shows the cpu time in seconds versus the number of steps, illustrating the essentially linear growth.

Acknowledgment. We thank Roland Klose for kindly providing his elasticity finite-element code.

REFERENCES

[1] R. Adami, A. Teta, *A simple model of concentrated nonlinearity.* in Mathematical results in quantum mechanics, Prague, Czech Republic, 1998, Oper. Theory Adv. Appl., 108, Birkhäuser, Basel, (1999), pp. 183–189.

[2] R. Adami, A. Teta, *A class of nonlinear Schrödinger equations with concentrated nonlinearity*, J. Funct. Anal., 180 (2001), pp. 148–175.

[3] K. Adolfsson, M. Enelund, S. Larsson, and M. Racheva, *Discretization of integro-differential equations modeling dynamic fractional order viscoelasticity* in Proc. Large-Scale Scientific Computations, Sozopol, Bulgaria, 2005 LNCS vol. 3743, Springer, (2006), pp. 76-83.

[4] K. Adolfsson, M. Enelund, and S. Larsson, *Adaptive discretization of fractional order viscoelasticity using sparse time history*, Computer Meth. Appl. Mech. Eng., 193 (2004), pp. 4567–4590.

[5] J. Alberty, C. Carstensen, S. A. Funken, and R. Klose, *Matlab Implementation of the Finite Element Method in Elasticity*, Computing, 69 (2002), pp. 239–263.

[6] X. Cao, K. Burrage, and F. Abdullah, *A variable stepsize implementation for fractional differential equations*, Report, 2006.

[7] D. Conte and I. Del Prete, *Fast collocation methods for Volterra integral equation of convolution type*, J. Comput. Appl. Math., 196 (2006), pp. 652–663.

[8] K. Diethelm and A. D. Freed, *The FracPECE Subroutine for the Numerical Solution of Differential Equations of Fractional Order*, in Forschung und wissenschaftliches Rechnen: Beiträge zum Heinz-Billing-Preis 1998, Gesellschaft fr̈ wissenschaftliche Datenverarbeitung, Göttingen (1999), pp. 57–71

[9] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *Data-sparse approximation to the operator-valued functions of elliptic operators*, Math. Comp., 73 (2004), pp. 1297–1324.

[10] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *Data-sparse approximation to a class of operator-valued functions*, Math. Comp., 74 (2005), pp. 681–708.

[11] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations.* 2nd ed., Springer, Berlin, 2006.

[12] E. Hairer and G. Söderlind, *Explicit, time reversible, adaptive step size control*, SIAM J. Sci. Comput., 6 (2005), pp. 1838–1851.

[13] R. Hiptmair and A. Schädle, *Non-reflecting boundary conditions for Maxwell's equations*, Computing, 71 (2003), pp. 265–292.

[14] M. López-Fernández and C. Palencia, *On the numerical inversion of the Laplace transform of certain holomorphic mappings*, Appl. Numer. Math., 51 (2004), pp. 289–303.

[15] M. López-Fernández, C. Palencia, and A. Schädle, *A spectral order method for inverting sectorial Laplace transforms*, SIAM J. Numer. Anal., 44 (2006), pp. 1332–1350.
[16] C. Lubich and A. Schädle, *Fast convolution for nonreflecting boundary conditions*, SIAM J. Sci. Comput., 24 (2002), pp. 161–182.
[17] W. McLean and V. Thomée, *Time discretization of an evolution equation via Laplace transforms*, IMA J. Numer. Anal., 24 (2004), pp. 439–463.
[18] R. Metzler, J. Klafter, *The random walk's guide to anomalous diffusion: a fractional dynamics approach*, Physics Reports, 339 (2000), pp. 1–77.
[19] M. Rizzardi, *A modification of Talbot's method for the simultaneous approximation of several values of the inverse Laplace transform*, ACM Trans. Math. Software, 21 (1995), pp. 347–371 .
[20] A. Schädle, *Ein schneller Faltungsalgorithmus für nichtreflektierende Randbedingungen.* Doctoral Thesis, Universität Tübingen, Germany, 2002.
[21] A. Schädle, M. López-Fernández, and C. Lubich, *Fast and oblivious convolution quadrature*, SIAM J. Sci. Comput. 28 (2006), pp. 421–438.
[22] D. Sheen, I. H. Sloan, and V. Thomée, *A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature*, Math. Comp., 69 (2000), pp. 177–195.
[23] J. R. Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator* in Applied Computational Geometry: Towards Geometric Engineering, LNCS, Vol. 1148, Springer, (1996) pp. 203–222.
[24] F. Stenger, *Approximations via Whittaker's Cardinal Function*, J. Approx. Theory, 17 (1976), pp. 222–240.
[25] F. Stenger, *Numerical methods based on Whittaker Cardinal, or sinc Functions*, SIAM Rev., 23 (1981), pp. 165–224.
[26] A. Talbot, *The accurate numerical inversion of Laplace transforms*, J. Inst. Math. Appl., 23 (1979), pp. 97–120.
[27] J.A.C. Weideman, *Optimizing Talbot's contours for the inversion of the Laplace transform*, Preprint, 2005.
[28] J.A.C. Weideman, L.N. Trefethen, *Parabolic and hyperbolic contours for computing the Bromwich integral*, Preprint, 2005.
[29] S. B. Yuste, L. Acedo, K. Lindenberg, *Reaction front in an A+B → C reaction-subdiffusion process*, Phys. Rev. E 69, (2004), pp. 036126.

Appendix A. Pseudocodes for the algorithm. We describe one step of the algorithm from t_{n-1} to a given new time $t = t_n$.

For all $\ell = 1, \dots, L$ and $k = -K, \dots, K$ the ODEs (3.5) corresponding to the $\lambda_k^{(\ell)}$ with initial time t_ℓ^- are advanced to the new time t or restarted, depending on whether the horizontal line at height t fits the current patch or enters a new patch (see Section 3.2 and Figure 3.1), i.e., we compute $y^{(\ell)}(t_\ell^-, t, \lambda_k^{(\ell)})$, or we set $t_\ell^- = t$ and $y^{(\ell)}(t_\ell^-, t, \lambda_k^{(\ell)}) = 0$, using the pseudocode 1.

In the following pseudocodes `Y(l)` denotes a structure storing:

- `Y(l).data` $\leftarrow y$, the solution of the ODE corresponding to the ℓ-patch,
- `Y(l).tini` $\leftarrow t_\ell^-$, the initial time of the ODE in the ℓ-patch,
- `Y(l).tcur` $\leftarrow t_n$, the new time (new final time of the ODE),
- `Y(l).b` $\leftarrow b_\ell$, the number of the current step in the corresponding ℓ-patch of the mosaic, along the vertical line from $(t_n, 0)$ to (t_n, t_n) (see Figure 3.1). This value ranges between 1 and B. In the example of Section 3.2 we have for $t_n = t_{15}$, $b_3 = 3$, $b_2 = 1$ and $b_1 = 3$.
- `Y(l).tmin` $\leftarrow t_{min}^{(\ell)} = h_{min} \sum_{k=\ell+1}^{L} b_k B^{k-1}$, the baseline of the ℓ-patch of the mosaic, along the vertical line from $(t_n, 0)$ to (t_n, t_n).
- `Y(l).tmax` $\leftarrow t_{max}^{(\ell)} = h_{min} \sum_{k=\ell}^{L} b_k B^{k-1}$, the top line of the ℓ-patch of the mosaic, along the vertical line from $(t_n, 0)$ to (t_n, t_n) ($t_{max}^{(\ell+1)} = t_{min}^{(\ell)}$),
- `Y(l).ub` $\leftarrow h_{min} \left(1 + \sum_{k=0}^{\ell} B^k\right)$ upper bound of the approximation interval,
- `Y(l).lb` $\leftarrow h_{min} \left(1 + \sum_{k=0}^{\ell-2} B^k\right)$ lower bound of the approximation interval,

- `Y(l).gini` $\leftarrow g(t_\ell^-)$ and `Y(l).gcur` $\leftarrow g(t_n)$, the values of the inhomogeneity at the initial time in the ℓ-patch and at the current time.

In the rare case that t_n is exactly on one of the horizontal lines of the mosaic, the structure `Y(l)` is copied to `YA(l)` before restarting for bookkeeping purposes.

Algorithm 1 Advance and restart the scalar ODEs. `odesol`

for $l = 1$ to L **do**
  if $t_n \geq Y(l).tmin + Y(l).b * B^{(l-1)} * hmin$ **then**
    if $t_n \geq Y(l).tmax$ **then**
      if $t_n = Y.tmax$ **then**
        $Y(l) =$ `odesadvance`$(Y(l), t_n, g_{n-1}, g_n)$;
        $Y(l).b = B$;
        $YA(l) = Y(l)$;
      end if
      restart the ODE $Y(l)$ *c.f.* Algorithm 2 ;
    else
      $Y(l) =$ `odesadvance`$(Y(l), t_n, g_{n-1}, g_n)$;
      while $t > Y(l).tmin + Y(l).b * B^{(l-1)} * hmin$ **do**
        $Y(l).b = Y(l).b + 1$;
      end while
    end if
  else
    $Y(l) =$ `odesadvance`$(Y(l), t_n, g_{n-1}, g_n)$;
  end if
end for

Algorithm 2 restart the ODE

$Y(l).data = 0 * Y(l).data$; $Y(l).b = 1$;
while $t_n > Y(l).tmin + Y(l).b * B^{(l-1)} * hmin$ **do**
  $Y(l).b = Y(l).b + 1$;
end while
$Y(l).tini = t$; $Y(l).tcur = t$; $Y(l).gini = g_n$; $Y(l).gcur = g_n$;
while $t_n \geq Y(l).tmax$ **do**
  $Y(l).tmin = Y(l).tmax$;
  $Y(l).tmax = Y(l).tmin + B^l * hmin$;
end while

Algorithm 3 Advancing the ODE (exponential Euler method) `odesadvance`

for $k = -K, \ldots, K$ **do**
  $Y(\ell).data_k = Y(\ell).data_k + (\exp(dt * \lambda_k^{(\ell)}) - 1)/\lambda_k^{(\ell)}$
    $*(\lambda_k^{(\ell)} * Y(\ell).data_k + g_{n-1} + (g_n - g_{n-1})/dt/\lambda_k^{(\ell)}) + (g_{n-1} - g_n)/\lambda_k^{(\ell)}$;
end for
$Y(\ell).gcur = g_n$; $Y(\ell).tcur = t$;

To fill the mosaic botton-up, Algorithm 4 is used. There copying the structure `Y(l)` to `YM(l)` and `YA(l)` is done by checking if the distances to the diagonal $t_n -$

`Y(l)`.t_{min} and $t_n - $`Y(l)`.$t_{max}$ fit the approximation interval I_ℓ.

Algorithm 4 update routine `update`

> **for** ℓ from L downto 1 **do**
>   **if** $t_n - YM(\ell).tmin - hmin * YM(\ell).b * B^{(\ell-1)} \geq Y(\ell).lb$ & $t_n - YM.tini \leq Y(\ell).ub$ **then**
> $YT(\ell) = YM(\ell)$;
>   **end if**
>   **if** $t_n \geq Y(\ell).tmax$ **then**
> $YA(\ell) = Y(\ell)$;
> $YM(\ell) = Y(\ell)$;
>   **else if** $t_n \geq Y(\ell).tmin + hmin * YM.b * B^{(\ell-1)}$ **then**
> $YM(\ell) = Y(\ell)$;
>   **end if**
>   **if** $t_n - YM(\ell).tmin - hmin * YM(\ell).b * B^{(\ell-1)} \geq Y(\ell).lb$ & $t_n - YM.tini \leq Y(\ell).ub$ **then**
> $YT(\ell) = YM(\ell)$;
>   **end if**
>   **if** $t_n - YA(\ell).tmin - hmin * YA(\ell).b * B^{(\ell-1)} \geq Y(\ell).lb$ & $t_n - YA.tini \leq Y(\ell).ub$ & $YA(\ell).tini > YT(\ell).tini$ **then**
> $YT(\ell) = YA(\ell)$;
>   **end if**
> **end for**

`YT(l)` is updated by checking if either of the distances to the diagonal corresponding to `YM(l)` or `YA(l)` fit the approximation intervals.

Algorithm 5 update routine part 2 `update`

> $idv = [\,]$; $ito = 1$;
> **if** $t_n - YT(L).tcur \geq Y(L).lb$ & $t_n - YT(L).tini \leq Y(L).ub$ **then**
> $idv = [L,\ idv]$;
> **else**
> $YT(L).tini = -inf$; $YT(L).tcur = -inf$;
> **end if**
> **for** ℓ from $L-1$ downto 1 **do**
>   **if** $t_n - YT(\ell).tcur \geq Y(\ell).lb$ & $t_n - YT(\ell).tini \leq Y(\ell).ub$ & $YT(\ell).tini > YT(\ell+1).tini$ & $YT(\ell).tcur > YT(\ell+1).tcur$ **then**
> $idv = [\ell,\ idv]$;
>   **else**
> $YT(\ell).tini = -inf$; $YT(\ell).tcur = -inf$;
>   **end if**
>   **if** $t_n - t_{n-1} \geq Y(\ell).lb$ & $t_n - t_{n-1} \leq Y(\ell).ub$ **then**
> $ito = \ell$;
>   **end if**
> **end for**
> $idv = [ito\ idv]$;

Algorithm 6 puts together the necesary “direct” and “odes” steps. It uses the ode solutions `YT` and the vector `idv` from Algorithm 5. `idv` stores the orders ℓ of the

YT required at $t = t_n$.

Algorithm 6 Fast convolution evaluation, *cf.* (3.7)

```
out = 0 ;
if idv is not empty then
  if length(idv) ≥ 2 then
    if t_{n-1} > YT(idv(2)).tcur then
      out = out+
      directstep(t, YT(idv(2)).tcur, t_{n-1}, YT(idv(2)).gcur, g_{n-1}) ; (3.11)
    end if
    for ll = 2 : length(idv) − 1 do
      if YT(idv(ll)).tcur > YT(idv(ll)).tini then
        ℓ = idv(ll) ;
        out = out + Σ_{k=-K}^{K} w_k^{(ℓ)} F(λ_k^{(ℓ)}) exp(t − YT(ℓ).tcur) ∗ YT(ℓ).data_k ; (3.6)
      end if
      if YT(idv(ll)).tini > YT(idv(ll + 1)).tcur then
        out = out + directstep(t, YT(idv(ll + 1)).tcur, YT(idv(ll)).tini,
              YT(idv(ll + 1)).gcur, YT(idv(ll)).gini) ; (3.11)
      end if
    end for
    if YT(idv(end)).tcur > YT.(idv(end))tini then
      ℓ = idv(end) ;
      out = out + Σ_{k=-K}^{K} w_k^{(ℓ)} F(λ_k^{(ℓ)}) exp(t − YT(ℓ).tcur) ∗ YT(ℓ).data_k ; (3.6)
    end if
  end if
end if
```