SIAM J. Sci. Comput.
Vol. 28, No. 2, pp. 421–438

FAST AND OBLIVIOUS CONVOLUTION QUADRATURE*

ACHIM SCHÄDLE†, MARÍA LÓPEZ-FERNÁNDEZ‡, and CHRISTIAN LUBICH§

Abstract. We give an algorithm to compute N steps of a convolution quadrature approximation to a continuous temporal convolution using only $O(N \log N)$ multiplications and $O(\log N)$ active memory. The method does not require evaluations of the convolution kernel but instead uses $O(\log N)$ evaluations of its Laplace transform, which is assumed sectorial. The algorithm can be used for the stable numerical solution with quasi-optimal complexity of linear and nonlinear integral and integro-differential equations of convolution type. In a numerical example we apply it to solve a subdiffusion equation with transparent boundary conditions.

Key words. convolution, numerical integration, Runge–Kutta methods, Volterra integral equation, anomalous diffusion

AMS subject classification. 65R20

DOI. 10.1137/050623139

1. Introduction.

In this paper we give a fast and memory-saving algorithm for computing the approximation of a continuous convolution (possibly matrix × vector)

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

by a convolution quadrature with a step size $h > 0$,

$$\sum_{j=0}^{n} \omega_{n-j}\, g(jh)\ , \qquad n = 1, \dots, N. \tag{1.2}$$

Here, the convolution quadrature weights ω_n are the coefficients of the generating power series (see [12])

$$\sum_{n=0}^{\infty} \omega_n \zeta^n = F\Big(\frac{\delta(\zeta)}{h}\Big), \tag{1.3}$$

where $F(s)$ is the Laplace transform of the (possibly matrix-valued) convolution kernel $f(t)$, and $\delta(\zeta) = 1 - \zeta$ or $\delta(\zeta) = (1-\zeta) + \frac{1}{2}(1-\zeta)^2$ for the methods based on the first- or second-order backward difference formula, respectively. We will also consider closely related convolution quadrature formulas that are based on implicit Runge–Kutta methods, such as the Radau IIA schemes [13].

Convolution quadratures (1.2)–(1.3) were introduced in [10, 11] and have since been used in a variety of applications. A recent review is given in [12], which contains an extensive, annotated bibliography. Attractive features of these convolution quadratures are that they work well for singular kernels $f(t)$, for kernels with multiple time

*Received by the editors January 24, 2005; accepted for publication (in revised form) October 28, 2005; published electronically March 31, 2006.
http://www.siam.org/journals/sisc/28-2/62313.html

†ZIB Berlin, Takustr. 7, D-14195 Berlin, Germany (schaedle@zib.de). This author's research was supported by the DFG Research Center Matheon, Berlin.

‡Departamento de Matemática Aplicada, Universidad de Valladolid, Valladolid, Spain (marial@mac.cie.uva.es). This author's research was 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. (lubich@na.uni-tuebingen.de). This author's research was supported by DFG, SFB 382.

scales, and in the common situation where the Laplace transform $F(s)$ (the transfer function) but not the convolution kernel $f(t)$ (the impulse response) is known analytically. Perhaps most important, these convolution quadratures enjoy excellent stability properties in the discretization of integral equations or integrodifferential equations of convolution type, in a way often strikingly opposed to discretizations with more straightforward quadrature formulas. This comes about because the stability of a discrete convolution equation depends on the range of the generating function of the weights for $|\zeta| < 1$. In (1.3), the generating function is directly related to the Laplace transform $F(s)$ which determines the stability properties of the continuous convolution equation.

The direct way to compute (1.2) is to first compute and store the (possibly matrix-valued) weights $\omega_0, \dots, \omega_N$, which can be done accurately with $O(N)$ evaluations of the Laplace transform $F(s)$ [11], and then to compute the discrete convolution. Done naively, this requires $O(N^2)$ multiplications (possibly matrix $\times$ vector) and $O(N)$ active memory for the values $g(jh)$ and for the weights. Using FFT, the number of multiplications can be reduced to $O(N \log N)$ and to $O(N (\log N)^2)$ in the case of integral equations where the values of $g(t)$ are not known beforehand, but where $g(nh)$ is computed only in the nth time step [3]. However, that approach does not reduce the number of F-evaluations and the memory requirements.

Here we give an algorithm, also applicable in the case of linear and nonlinear integral equations, which computes (1.2) in a way that requires

- $O(N \log N)$ multiplications,
- $O(\log N)$ evaluations of the Laplace transform $F(s)$, and
- $O(\log N)$ active memory.

The history $g(jh)$ for $j = 0, \dots, N$ is forgotten in this algorithm, and only logarithmically few linear combinations of the g-values are kept in memory. These are obtained by solving numerically, with step size h, initial value problems of the form $y' = \lambda y + g$ with complex λ. The weights ω_n $(n \leq N)$ are not computed explicitly, except the first few, e.g., the first 10 weights.

The algorithm presented here takes up ideas of the fast convolution algorithm of [14], which instead of (1.2)–(1.3) makes a different approximation to the continuous convolution. The stability properties of the second-order method of [14] in integro-partial differential equations such as those of section 5 are, however, extremely difficult to analyze (cf. also [17]) and remain entirely unclear for higher-order extensions, whereas the stability of convolution quadratures (1.2)–(1.3) for such problems is well understood; see [1]. Here we show how the convolution quadratures (1.2)–(1.3) with all their known favorable properties can be implemented in an equally fast and memory-saving way as the method in [14].

Following the error analysis of [7, 8] we give exponentially convergent error bounds for the contour integral approximations that are employed in this algorithm. They ensure that the constants hidden in the O-symbols of the above work estimates depend only logarithmically on the error tolerance for these contour integral approximations.

We assume a *sectorial* Laplace transform $F(s)$:

(1.4)
$F(s)$ is analytic in a sector $|\arg(s - \sigma)| < \pi - \varphi$ with $\varphi < \frac{1}{2}\pi$, and in this sector

$$|F(s)| \leq M\,|s|^{-\nu} \quad \text{for some real } M \text{ and } \nu > 0.$$

The inverse Laplace transform is then given by

$$f(t) = \frac{1}{2\pi i} \int_\Gamma e^{t\lambda} F(\lambda)\, d\lambda, \qquad t > 0, \tag{1.5}$$

with Γ a contour in the sector of analyticity, going to infinity with an acute angle to the negative real half-axis and oriented with increasing imaginary part. The function $f(t)$ is analytic in $t > 0$ and satisfies

$$|f(t)| \le C\, t^{\nu-1} e^{ct}, \qquad t > 0, \tag{1.6}$$

and is therefore locally integrable. (The absolute values on the left-hand sides of the bounds (1.4) and (1.6) are to be interpreted as matrix norms for matrix-valued convolution kernels.) A prototypical example, which will actually be used in our numerical experiments, is the fractional-power kernel $f(t) = t^{\nu-1}/\Gamma(\nu)$ with $\nu > 0$, which has the Laplace transform $F(s) = s^{-\nu}$.

In section 2 we review convolution quadrature based on multistep and Runge–Kutta methods. We give a contour integral representation of the convolution quadrature weights whose discretization along hyperbolas or Talbot contours is discussed in section 3. The fast and oblivious convolution algorithm is formulated in section 4. Finally, in section 5 we give the results of numerical experiments with integral and integrodifferential equations originating from regular and anomalous diffusion problems.

2. Convolution quadrature.

In this section we review briefly convolution quadrature and give a contour integral representation of the convolution quadrature weights on which the fast algorithm of this paper is based.

2.1. Convolution quadrature based on multistep methods.

We consider the convolution quadrature (1.2) with weights (1.3). By (1.4) and Cauchy's integral formula we have, with a contour Γ as in (1.5),

$$\sum_{n=0}^{\infty} \omega_n \zeta^n = F\Big(\frac{\delta(\zeta)}{h}\Big) = \frac{1}{2\pi i} \int_\Gamma \Big(\frac{\delta(\zeta)}{h} - \lambda\Big)^{-1} F(\lambda)\, d\lambda.$$

Hence, with $e_n(z)$ defined by

$$(\delta(\zeta) - z)^{-1} = \sum_{n=0}^{\infty} e_n(z)\, \zeta^n, \tag{2.1}$$

we have the integral formula

$$\omega_n = \frac{h}{2\pi i} \int_\Gamma e_n(h\lambda)\, F(\lambda)\, d\lambda, \tag{2.2}$$

which can be viewed as the discrete analogue of (1.5). For the backward Euler discretization $\delta(\zeta) = 1 - \zeta$ we note the explicit formula

$$e_n(z) = (1 - z)^{-n-1}, \tag{2.3}$$

which is of the form $e_n(z) = q(z) r(z)^n$ with $r(z) = \frac{1}{1-z}$ and $q(z) = \frac{1}{1-z}$.

For the second-order backward differentiation formula (BDF), where $\delta(\zeta) = \sum_{k=1}^{p} \frac{1}{k}(1 - \zeta)^k$ with $p = 2$, we obtain from a partial fraction decomposition of $(\delta(\zeta) - z)^{-1}$ that

$$e_n(z) = \frac{1}{\sqrt{1+2z}} \Big((2 - \sqrt{1+2z})^{-n-1} - (2 + \sqrt{1+2z})^{-n-1} \Big), \tag{2.4}$$

which is of the form $e_n(z) = q_1(z)r_1(z)^n + q_2(z)r_2(z)^n$. Connoisseurs of Cardano's formulas find analogous formulas to (2.4) also for the BDF methods of orders 3 and 4.

2.2. Convolution quadrature based on Runge–Kutta methods. We consider an implicit Runge–Kutta method with coefficients a_{ij}, b_j, c_i for $i, j = 1, \dots, m$. We denote the Runge–Kutta matrix by $\mathcal{O} = (a_{ij})$, the row vector of the weights by $b^T = (b_j)$, and the stability function by

$$r(z) = 1 + zb^T(I - z\mathcal{O})^{-1}1,$$

where $1 = (1, \dots, 1)^T$. We assume that all eigenvalues of the Runge–Kutta matrix $\mathcal{O}$ have positive real part and that the method is A-stable and the row vector of the weights equals the last line of the Runge–Kutta matrix,

$$b_j = a_{mj} \quad \text{for } \; j = 1, \dots, m,$$

and correspondingly $c_m = 1$. Notice that this implies that the method is L-stable, i.e.,

$$|r(z)| \le 1 \text{ for } \mathrm{Re} z \le 0 \qquad \text{and} \;\; r(\infty) = 0.$$

These conditions are in particular satisfied by the Radau IIA family of Runge–Kutta methods [4]. From such a Runge–Kutta method, a convolution quadrature is constructed as follows [13]: let

$$\Delta(\zeta) = \Big(\mathcal{O} + \frac{\zeta}{1-\zeta} 1 b^T\Big)^{-1} \tag{2.5}$$

and define weight matrices W_n by

$$\sum_{n=0}^{\infty} W_n \zeta^n = F\Big(\frac{\Delta(\zeta)}{h}\Big). \tag{2.6}$$

Let $\omega_n = (\omega_n^1, \dots, \omega_n^m)$ denote the last row of W_n. Then an approximation to the convolution integral (1.1) at time $t_{n+1} = (n+1)h$ is given by

$$u_{n+1} = \sum_{j=0}^{n} \sum_{i=1}^{m} \omega_{n-j}^i \, g(t_j + c_i h) = \sum_{j=0}^{n} \omega_{n-j} \, g_j \tag{2.7}$$

with the column vector $g_j = \big(g(t_j + c_i h)\big)_{i=1}^m$. For a Runge–Kutta method of classical order p and stage order q, this approximation is known to be convergent of the order $\min(p, q+1+\nu)$ with ν of (1.4) [13, Theorem 2.2].

With the row vector $e_n(z) = (e_n^1(z), \dots, e_n^m(z))$ defined as the last row of the $m \times m$ matrix $E_n(z)$ given by

$$(\Delta(\zeta) - zI_m)^{-1} = \sum_{n=0}^{\infty} E_n(z)\, \zeta^n, \tag{2.8}$$

we obtain an integral formula like in (2.2),

$$\omega_n = \frac{h}{2\pi i} \int_\Gamma e_n(h\lambda) \otimes F(\lambda)\, d\lambda. \tag{2.9}$$

For $n \geq 0,\ e_n(z)$ is given as

$$e_n(z) = r(z)^n q(z) \tag{2.10}$$

with the row vector $q(z) = b^T(I - z\mathcal{O}\!\!\mathcal{l})^{-1}$; cf. [13, Lemma 2.4]. We note that

$$y_{n+1}^{(\lambda)} = h \sum_{j=0}^{n} e_{n-j}(h\lambda)\, g_j \tag{2.11}$$

is the Runge–Kutta approximation at time t_{n+1} of the linear initial value problem

$$y' = \lambda y + g(t), \quad y(0) = 0\,. \tag{2.12}$$

The convolution quadrature (2.7) is thus interpreted as

$$u_{n+1} = \frac{1}{2\pi i} \int_\Gamma F(\lambda)\, y_{n+1}^{(\lambda)}\, d\lambda\ ;$$

see [13, Proposition 2.1].

3. Approximation of the contour integrals.

The fast convolution algorithm will be based on discretizing the integrals in (2.2) and (2.9) along suitable complex contours. This approximation is discussed in the present section.

3.1. Quadrature on hyperbolas and Talbot contours.

The fast algorithm approximates the quadrature weights ω_n by linear combinations of the exponential approximations $e_n(h\lambda)$, locally on a sequence of fast-growing time intervals $nh \in I_\ell$:

$$I_\ell = [B^{\ell-1}h, 2B^\ell h), \tag{3.1}$$

where the base $B > 1$ is an integer. For example, $B = 10$ was found a good choice in our numerical experiments. The approximation on I_ℓ results from applying the trapezoidal rule to a parametrization of the contour integral for the convolution quadrature weights,

$$\omega_n = \frac{h}{2\pi i} \int_{\Gamma_\ell} e_n(h\lambda) \otimes F(\lambda)\, d\lambda \approx h \sum_{k=-K}^{K} w_k^{(\ell)} e_n(h\lambda_k^{(\ell)}) \otimes F(\lambda_k^{(\ell)})\,, \qquad nh \in I_\ell, \tag{3.2}$$

with an appropriately chosen complex contour Γ_ℓ. The number of quadrature points on Γ_ℓ, $2K + 1$, is chosen independent of ℓ. It is much smaller than what would be required for a uniform approximation of the contour integral on the whole interval $[0, T]$. The integration contour is chosen as a hyperbola [8] or a Talbot contour [20, 16].

The hyperbola is given by

$$\mathbb{R} \to \Gamma : \theta \mapsto \gamma(\theta) = \mu(1 - \sin(\alpha + i\theta)) + \sigma \tag{3.3}$$

with a positive scale parameter μ, a positive angle $\alpha < \pi/2 - \varphi$, and the shift σ of (1.4). The contour thus remains in the sector (1.4) of analyticity of $F(s)$. Additionally the contour is chosen such that the singularities of $e_n(hs)$ lie to the right of the contour. See the left part of Figure 3.1 for $\alpha = \pi/2 - 1/2$. The parameter μ will depend on ℓ, which yields a contour Γ_ℓ depending on the approximation interval I_ℓ. The weights and quadrature points in (3.2) are given by (omitting ℓ in the notation)

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

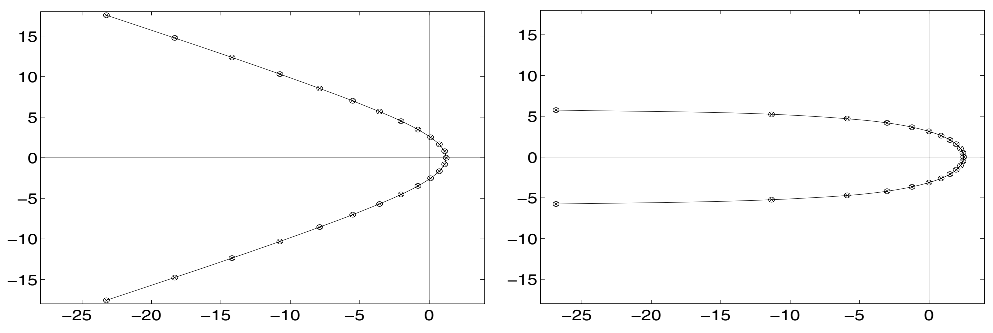

FIG. 3.1. *Hyperbola (left) and Talbot contour (right).*

where τ is a step length parameter.

Alternatively, the Talbot contour is given by

$$(-\pi,\pi)\to\Gamma:\theta\mapsto\gamma(\theta)=\mu\left(\theta\cot(\theta)+i\kappa\theta\right)+\sigma \tag{3.4}$$

where the parameters μ, κ, and σ are such that the singularities of $F(s)$ lie to the left of the contour and that the singularities of $e_n(hs)$ lie to the right of the contour. The parameter μ will again depend on ℓ. See the right part of Figure 3.1 for $\sigma=0$. The weights and quadrature points in (3.2) are given by (omitting ℓ in the notation)

$$w_k=-\frac{i}{2(K+1)}\,\gamma'(\theta_k)\ ,\qquad \lambda_k=\gamma(\theta_k)\qquad\text{with}\qquad \theta_k=\frac{k\pi}{K+1}\ .$$

3.2. Theoretical error bounds of the contour integral approximations. For the case of the hyperbola, we obtain with the proof of [7, Theorem 3] the following error bound, which shows exponential convergence.

THEOREM 3.1. *There are positive constants C, d, $c_0,\dots,c_4$, and c such that at $t=nh\le T$ the quadrature error in* (3.2) *for a hyperbola* (3.3) *with $1\le c\mu t\le n$ is bounded by*

$$\|E(\tau,K,h,n)\|\le C\,h\,t^{\nu-1}\,(\mu t)^{1-\nu}\left(\frac{e^{c_0\mu t}}{e^{2\pi d/\tau}-1}+e^{(c_1-c_2\cosh(K\tau))\mu t}\right.$$
$$\left.+\ e^{c_3\mu t}\Big(1+\frac{c_4\cosh(K\tau)\mu t}{n/2}\Big)^{-n/2}\right),$$

where ν is the exponent of (1.4).

The first summand in this estimate corresponds to the error in the discretization of the integral (3.2), parametrized over the real line with an integrand holomorphic in a strip $|\mathrm{Im}\,\theta|<d$, with step size τ and with the infinite quadrature sum over all integers k [18, 19]. The other two terms result from truncating the infinite sum to the terms $-K\le k\le K$, using the strong decay of the integrand along the integration contour.

Given an error tolerance ε, the first term in the error bound becomes $O(\varepsilon\,h\,t^{\nu-1})$ if τ is chosen so small that $c_0\mu t-2\pi d/\tau\le\log\varepsilon$, which requires an asymptotic proportionality $\frac{1}{\tau}\sim\log\frac{1}{\varepsilon}+\mu t$. For μ chosen such that $\frac{a_1}{B}\log\frac{1}{\varepsilon}\le\mu t\le a_1\log\frac{1}{\varepsilon}$ with an arbitrary positive constant a_1 and with $B>1$, we obtain that the second term is $O(\varepsilon\,h\,t^{\nu-1})$ if $c_1-c_2\cosh(K\tau)\le -B/a_1$, i.e., with $\cosh(K\tau)=a_2$ for a sufficiently large constant a_2. With the above choice of τ, this yields $K \sim \log \frac{1}{\varepsilon}$. The third term then becomes smaller than $\varepsilon\, h\, t^{\nu-1}$ for $n \geq c \log \frac{1}{\varepsilon}$ with a sufficiently large constant c. In summary, this gives the following bound for the required number of quadrature points on the hyperbola.

THEOREM 3.2. *In* (3.2)*, a quadrature error bounded in norm by $\varepsilon\, h\, t^{\nu-1}$ for $nh \in I_\ell$ is obtained with $K = O(\log \frac{1}{\varepsilon})$. This holds for $n \geq c \log \frac{1}{\varepsilon}$ (with some constant $c > 0$) with K independent of ℓ and of n and h with $nh \leq T$.*

No small error bound is, however, obtained for the first few $n < c \log \frac{1}{\varepsilon}$. The approximation is indeed poor for small n, as will be seen in the numerical experiments in section 3.4. Therefore the first few ω_n are not computed using the integral representation (3.2) but are computed using the method of section 3.5.

We expect that a similar result to Theorem 3.2 holds also for the Talbot contours, if the Laplace transform has an analytic continuation beyond the negative real axis from above and below, as is the case for the fractional powers considered below.

3.3. Choice of parameters for the hyperbolas. The estimate in Theorem 3.1 is valid for the discretization of contour integrals of the type considered in (3.2), where $e_n(h\lambda)$ is an approximation to $\exp(nh\lambda)$. Replacing $e_n(h\lambda)$ by $\exp(nh\lambda)$ in (3.2) we obtain the inverse Laplace transform at nh of F. For the numerical inversion of the Laplace transform there is a spectral order method developed in [9], where the error estimate is explicit in all constants involved. Getting an estimate explicit in all constants for the discretization of (3.2) is much more difficult and, in practice, we choose the parameters following the strategy proposed in [9] for the numerical inversion of the Laplace transform. This makes sense as for large n and small h with $nh \leq T$ the estimate in Theorem 3.1 tends to an expression of the same type as the error estimate for the approximation of the inverse Laplace transform in [9].

Given the approximation interval $[B^{\ell-1}h, (2B^\ell - 2)h]$ and K the number of nodes on the hyperbola, the strategy for choosing the parameters is as follows:

1. Choose $\alpha = d = (\pi/2 - \varphi)/2$, with φ of (1.4).
2. Minimize for $0 < \rho < 1$ the expression
$$\mathrm{eps}\, \epsilon_K(\rho)^{\rho-1} + \epsilon_K(\rho)^{\rho},$$
where
$$\epsilon_K(\rho) = \exp\Big(-\frac{2\pi d}{a(\rho)}K\Big), \qquad a(\rho) = \operatorname{acosh}\Big(\frac{2B}{(1-\rho)\sin\alpha}\Big)$$
and eps is the machine precision.
3. Take
$$\tau = \frac{1}{K}a(\rho_{opt}), \qquad \mu = \frac{2\pi d K(1-\rho_{opt})}{(2B^\ell - 2)ha(\rho_{opt})}.$$

This strategy for choosing the hyperbola contours is used in the numerical experiments below.

3.4. Numerical experiments. In view of the examples of section 5 we present here numerical experiments with
$$f(t) = \frac{1}{\sqrt{\pi t}} \quad \text{for which} \quad F(s) = s^{-1/2}.$$

The error is calculated with respect to a reference solution, obtained for a discretization of the contour integral with a large number of integration points. For the

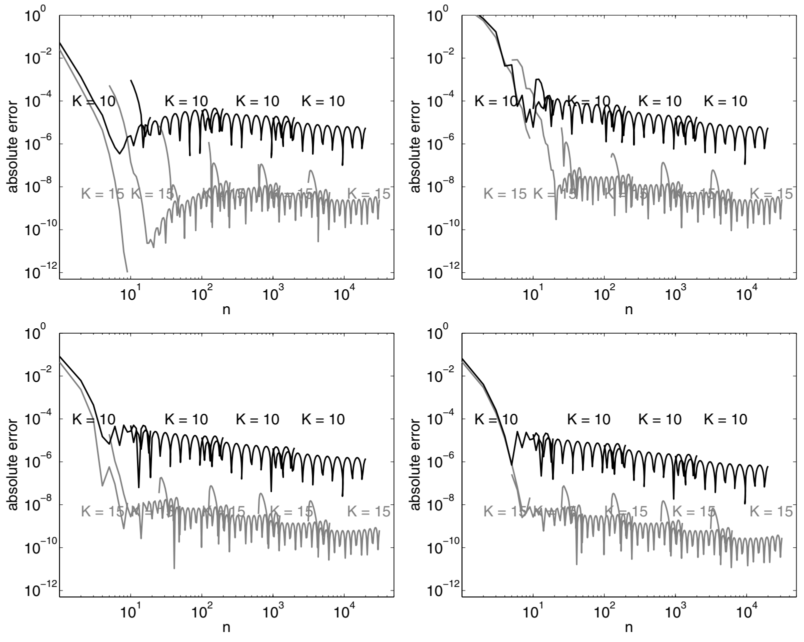

FIG. 3.2. *Hyperbola quadrature errors versus time in logarithmic scale for $K = 15$, $B = 5$ and $K = 10$, $B = 10$ for different Integrators. (Implicit Euler, BDF(2), RadauIIA(3), and RadauIIA(5) in clockwise order starting from the upper left).*

Radau IIA methods of order 3 and 5, where the ω_n are row vectors of dimension 2 and 3, respectively, we plot the error of the last entry.

Using the hyperbola contours, for $B = 10$ and $K = 10$ we obtain an absolute error of about 10^{-4} on the interval I_ℓ for $\ell \geq 2$ taking $\alpha = 1$, μ and τ as described above. For an absolute approximation error of $3 \cdot 10^{-8}$, we take $B = 5$, $K = 15$, and the other parameters as before; cf. Figure 3.2. For $n > 20$ there is no substantial difference between the quadrature errors corresponding to the different Runge–Kutta methods.

Using the Talbot contours, the following choices of parameters were experimentally found to give good results. An absolute accuracy of about 10^{-3} on the interval I_ℓ for $\ell \geq 2$ with right end-point T_ℓ is obtained with $B = 10$, $K = 10$, $\mu = 8/T_\ell$, $\kappa = 0.6$. For an absolute approximation error of 10^{-6}, take $B = 5$, $K = 15$, and the other parameters as before; cf. Figure 3.3. For $n > 20$ there is again no essential difference between the different Runge–Kutta methods.

As we commented after Theorem 3.2, the approximations to the first few convolution quadrature weights are poor and so they will not be used in the algorithm.

Figure 3.4 shows the absolute errors on the interval $[10, 20000]$ (similar for any interval $[a, 2000a]$ with $a > 10$) for the RadauIIA(3) method with $K = 10, 20, 40, 80, 160, 320, 640$. For the implicit Euler, the BDF(2), and the RadauIIA(5) methods these error plots look similar. This behavior of the errors clearly demonstrates the advantage of using local approximations. With $B = 10$, we need three approximation intervals to cover the interval $[10, 20000]$, so that for a work of $3 \cdot K$ with $K = 10$ we obtain better accuracy than with $K = 640$ over the whole interval.

In this example the maximum quadrature errors using the hyperbolas are smaller than those for the Talbot contours. Moreover, the hyperbolas allow us to choose larger

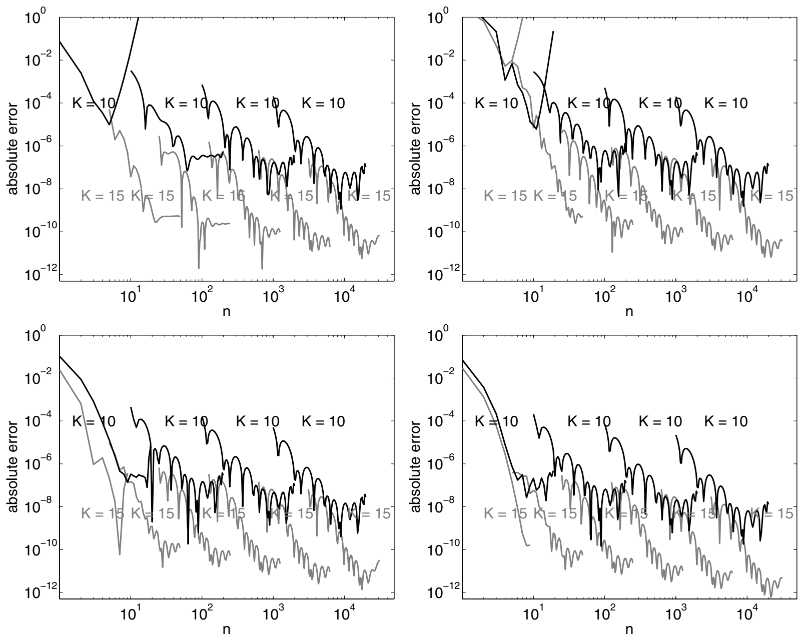

FIG. 3.3. *Talbot quadrature errors versus time in logarithmic scale for $K = 15$, $B = 5$ and $K = 10$, $B = 10$ for different Integrators. (Implicit Euler, BDF(2), Radau*IIA*(3), and Radau*IIA*(5) in clockwise order starting from the upper left).*

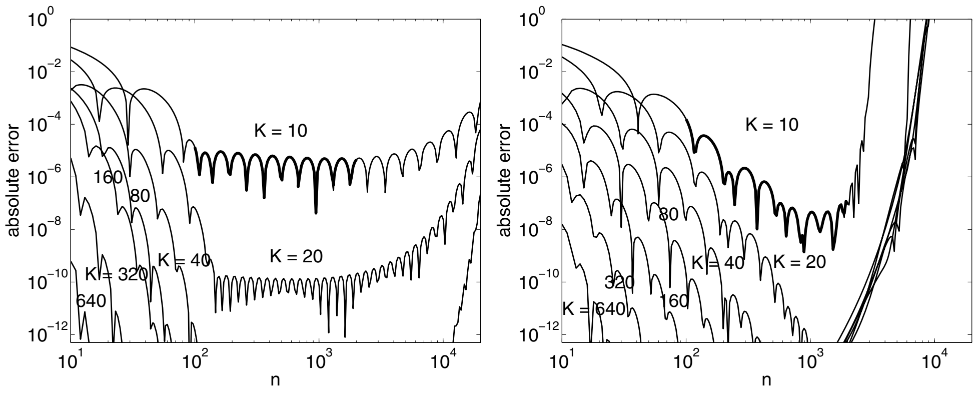

FIG. 3.4. *Hyperbola (left) and Talbot (right) quadrature error versus n in logarithmic scale for the Radau*IIA*(3) method with $K = 10, 20, 40, 80, 160, 320, 640$. The bold parts of the error curve correspond to the lower left parts of Figures* 3.3 *and* 3.2.

intervals. On the other hand, the Talbot contours turned out to be less sensitive to the choice of parameters and the Laplace transform functions than the hyperbolas.

3.5. Computation of the first few convolution quadrature weights. In view of the poor approximation properties of the discretized contour integral for the first few ω_n, the first N_0 (e.g., $N_0 = 10$ or 20, and asymptotically $N_0 = c \log \frac{1}{\varepsilon}$) convolution quadrature weights are approximated differently, using their representation as an integral over a circle:

$$\omega_n = \text{ last row of } \frac{1}{2\pi i}\int_{|\zeta|=\rho} \zeta^{-n-1}\, F\Big(\frac{\Delta(\zeta)}{h}\Big)\, d\zeta, \qquad n \le N_0. \tag{3.5}$$

This is approximated by the trapezoidal rule,

$$\omega_n \approx \text{ last row of } \frac{\rho^{-n}}{J}\sum_{j=1}^{J} F\Big(\frac{\Delta(\zeta_j)}{h}\Big)\, e^{-2\pi i n j/J}, \qquad n \le N_0, \tag{3.6}$$

with $\zeta_j = \rho e^{-2\pi i j/J}$. The choice of the parameters ρ and J follows [11]: assuming that the values of F are computed with precision $\widehat{\epsilon}$, the error in ω_n $(0 \le n \le N_0)$ is $O(\widehat{\epsilon}^{1/2})$ if $J = N_0 + 1$ and $\rho^J = \widehat{\epsilon}^{1/2}$, and the error is $O(\widehat{\epsilon})$ if $J \ge N_0 \log(1/\widehat{\epsilon})$ and $\rho = e^{-\gamma h}$, with $\gamma > \sigma$ of (1.4).

4. The fast and oblivious algorithm.

We now describe the convolution algorithm, concentrating on Runge–Kutta-based convolution quadrature. The algorithm differs slightly depending on whether we want to compute a convolution or to solve an integral or integrodifferential equation of convolution type.

4.1. The algorithm for computing convolutions.

The algorithm presented here uses the organization scheme of the fast convolution algorithm described in a step-by-step manner in [14]. A pseudocode for the algorithm developed in [14] can be found in [5]. We now describe the fast and memory-saving algorithm for computing (2.7) for $n \le N$.

For fixed integer $n \le N$ and a given base B we split the discrete convolution (1.2) or (2.7) into L sums, where L is the smallest integer such that $n < 2B^L$:

$$u_{n+1} = \sum_{j=0}^{n} \omega_{n-j}\, g_j = u_{n+1}^{(0)} + \cdots + u_{n+1}^{(L)}$$

$$\text{with} \quad u_{n+1}^{(0)} = \omega_0\, g_n \ \text{ and } \ u_{n+1}^{(\ell)} = \sum_{j=b_\ell}^{b_{\ell-1}-1} \omega_{n-j}\, g_j$$

for suitable $n = b_0 > b_1 > \cdots > b_{L-1} > b_L = 0$. In view of the approximation intervals (3.1), the splitting is done in such a way that for fixed ℓ in each sum from b_ℓ to $b_{\ell-1} - 1$, we have $n - j \in [B^{\ell-1}, 2B^\ell - 2]$. The $b_\ell = \mathtt{b}(\ell)$, $\ell = 1, \ldots, L-1$ are determined recursively by the following pseudocode:

```
 L = 1; q = 0;
 for n = 1 to N do
    if (2*B^L == n+1) then L = L+1; endif
    k = 1;
    while ((mod(n+1,B^k) == 0) & (k < L))
       q(k) = q(k)+1;
       k = k+1;
    endwhile
    for k = 1 to L-1 do b(k) = q(k)*B^k; endfor
endfor
```

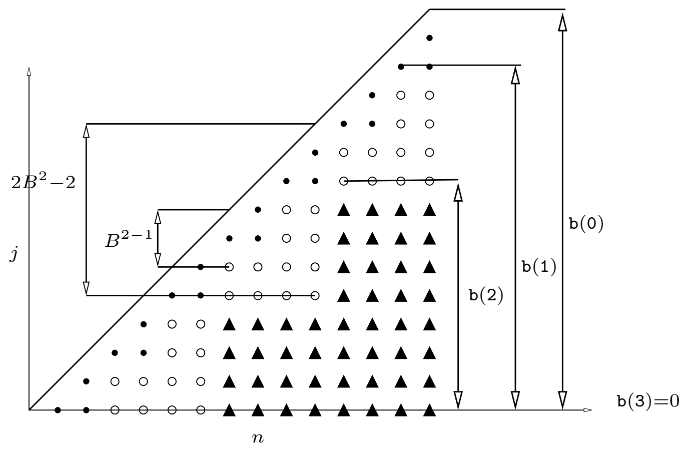

FIG. 4.1. *Visualization of the scheme for $B = 2$ to determine the values* $\mathtt{b(l)}$. *For $n = 14$, $L = 3$ and* $\mathtt{b} = [14\ 12\ 8\ 0]$. *Additionally the approximation interval for $\ell = 2$ is indicated.*

Note that for growing n, b_ℓ is augmented by B^ℓ every B^ℓ steps. The procedure is illustrated for $B = 2$ in Figure 4.1.

On inserting the integral representation (2.9) of the Runge–Kutta quadrature weights and the relation (2.10), i.e., $e_{n-j}(h\lambda) = r(h\lambda)^{n-j}q(h\lambda)$, we obtain

$$u^{(\ell)}_{n+1} = \sum_{j=b_\ell}^{b_{\ell-1}-1} \omega_{n-j}\, g_j = \sum_{j=b_\ell}^{b_{\ell-1}-1} \frac{h}{2\pi i}\int_{\Gamma_\ell} e_{n-j}(h\lambda)\otimes F(\lambda)\, d\lambda\, g_j \tag{4.1}$$
$$= \frac{1}{2\pi i}\int_{\Gamma_\ell} r(h\lambda)^{n-(b_{\ell-1}-1)}\, F(\lambda)\, y^{(\ell)}(h\lambda)\, d\lambda$$

with

$$y^{(\ell)}(h\lambda) = h \sum_{j=b_\ell}^{b_{\ell-1}-1} e_{(b_{\ell-1}-1)-j}(h\lambda)\, g_j\,.$$

Comparing this formula with (2.11), we see that $y^{(\ell)}(h\lambda)$ is the Runge–Kutta approximation $y(b_{\ell-1}h, b_\ell h, \lambda)$ to the solution at $t = b_{\ell-1}h$ of the linear initial-value problem

$$y' = \lambda y + g(t), \quad y(b_\ell h) = 0, \tag{4.2}$$

and hence $y^{(\ell)}(h\lambda)$ is computed as such, by Runge–Kutta time stepping. The integrals are discretized with the quadrature formula discussed in section 3:

$$u^{(\ell)}_{n+1} \doteq \sum_{k=-K}^{K} w^{(\ell)}_k\, r(h\lambda^{(\ell)}_k)^{n-b_{\ell-1}+1}\, F(\lambda^{(\ell)}_k)\, y^{(\ell)}(h\lambda^{(\ell)}_k). \tag{4.3}$$

In the nth time step, we thus compute $u^{(\ell)}_{n+1}$ and for subsequent use we update the Runge–Kutta solutions to the $(2K+1)L$ initial value problems (4.2) for the integration points $\lambda^{(\ell)}_k$ on the contours Γ_ℓ for $\ell = 1, \ldots, L$, doing one time step from t_n to t_{n+1} in each of these differential equations.

This algorithm does not keep the history g_j ($j = 0, \dots, n$) in memory. Instead, for each $\ell = 1, \dots, L$, it stores the Runge–Kutta approximation `yRK(l)` to (4.2) at the current time step, the values $w_k^{(\ell)}$, $\lambda_k^{(\ell)}$, $r(h\lambda_k^{(\ell)})$, $F(\lambda_k^{(\ell)})$, the vector `y(l)` $= \big(y^{(\ell)}(h\lambda_k^{(\ell)})\big)$, and two auxiliary vectors `yA(l)` and `yM(l)` of the same dimension as `yRK(l)` and `y(l)` ($(2K+1) \times (\operatorname{size} g)$) needed for bookkeeping purposes.

Proceeding from t_n to t_{n+1} the Runge–Kutta solutions `yRK(l)` of (4.2) are advanced from $y(t_{b_\ell}, t_n, \lambda_k^{(\ell)})$ to $y(t_{b_\ell}, t_{n+1}, \lambda_k^{(\ell)})$ for $\ell = 2, \dots, L$ and all $k = -K \dots, K$.

The bookkeeping for the `yA` and `yM` is done using the following pseudocode:

```
l = 2;
while ((mod(n+1,Basis^(l-1)) == 0) & (l <= L))
  if (mod(n+1, Basis^l) == 0) then
    yA(l) = yRK(l);
    yM(l) = yRK(l);
    restart yRK(l);
  else
    yM(l) = yRK(l));
  endif
  l = l+1;
endwhile
```

Here "`restart yRK(l)`" means that `yRK(l)` is set to $y(t_{b_\ell}, t_{b_\ell}, \lambda_k^{(\ell)}) = 0$. This way the scheme in Figure 4.1 is build bottom up.

To retrieve the values `y(l)` $= y^{(\ell)}(h\lambda_k^{(\ell)})$ required at the current time from the `yA` and `yM` we use the `b(l)`. Again we give the pseudocode:

```
tvek = h*b;  % b is the vector of b(l); l = 0 , ... , L
for l = 2 to  length(tvek)-1 do
  y(l) = r(l) * y(l);
  if (tvek(l) == final time of yM(l)) then
    y(l) = yM(l);
    if (l == 2) then g = g(1:B-1);  endif
  endif
  if (tvek(l+1) ~= initial time of yM(l)) then
    y(l) = y(l) + yA(l)*r(l)^(B^(l-1));
  endif
endfor
```

Here `r(l)` denotes the stability function given in (2.10) and `g` is the vector containing the values $g_n, \dots, g_{b_1}$.

There are only $(2K+1)L$ evaluations of the Laplace transform $F(s)$. In the case of real functions $f(t)$ and $g(t)$ only the real parts of the above sums are used, since the imaginary ones cancel out, and hence the factor $2K+1$ can be replaced by $K+1$, since the quadrature points lie symmetric with respect to the real axis. We recall $L \le \log_B N$ and $K = O(\log \frac{1}{\varepsilon})$, where ε is the accuracy requirement in the discretization of the contour integrals.

In view of the poor approximation of the first convolution quadrature weights by the discretization of the contour integral, we evaluate $u_{n+1}^{(\ell)}$ directly for a few of the first ℓ, e.g., for $\ell = 0, 1$ with $B = 10$. For this we need to keep the $n - b_1 + 1 \le 2B$

values $g_{b_1}, \dots, g_n$ in memory, but none of the earlier history g_j for $j \le n - 2B$. We also need the few convolution quadrature weights $\omega_0, \dots, \omega_{2B-1}$, which are computed from (3.6) with $J = 2B$ evaluations of the Laplace transform $F(s)$, as described in section 3.5.

For the convolution quadrature based on the second-order BDF method a similar fast algorithm is obtained by inserting the formula (2.4) for $e_{n-j}(h\lambda)$ in (4.1).

4.2. The algorithm for solving integral equations. Consider now a Volterra integral equation

$$u(t) = a(t) + \int_0^t f(t-\tau)\, g(\tau, u(\tau))\, d\tau\,, \qquad t \ge 0, \tag{4.4}$$

with a kernel $f(t)$ whose Laplace transform satisfies the sectorial condition (1.4), with a smooth nonlinearity $g(t,u)$ and inhomogeneity $a(t)$. The adaptation of the above algorithm to such convolution equations is straightforward for the case of the convolution quadrature based on the implicit Euler method and the second-order BDF method, which use solution approximations only on the grid $t = nh$. The extension of the Runge–Kutta-based algorithm is, however, less immediate, because the integral approximation uses the internal stages of the Runge–Kutta method. Consider a Runge–Kutta-based convolution quadrature under the assumptions of section 2.2. With the column vector of internal stages $v_n = (v_{ni})_{i=1}^m$, the discretization of (4.4) reads

$$v_n = a_n + \sum_{j=0}^{n} W_{n-j}\, g_j\,, \qquad n \ge 0, \tag{4.5}$$

with $a_n = \big(a(t_n + c_i h)\big)_{i=1}^m$, with weight matrices W_n defined by (2.6), and with $g_j = \big(g(t_j + c_i h, v_{ji})\big)_{i=1}^m$ depending on the stages v_{ji}. The scheme is implicit in v_n. The solution at t_{n+1} is approximated by the last component of the stage vector v_n,

$$u_{n+1} = v_{nm}\,.$$

With the proof of [13, Theorem 4.1] we obtain that the error of this approximation over bounded time intervals is bounded by $O(h^k)$ with $k = \min(p, q+1)$, where p and q are the classical order and stage order, respectively, of the underlying Runge–Kutta method. This estimate holds under the assumption that the solution is sufficiently smooth. It gives orders 3 and 4 for the two- and three-stage Radau IIA methods, respectively. The precise approximation order for the three-stage method (of classical order 5) may become larger under appropriate conditions on the nonlinearity and the convolution kernel; cf. [13, Theorem 4.2].

The weight matrix W_n has the integral representation (cf. (2.9))

$$W_n = \frac{h}{2\pi i} \int_\Gamma E_n(h\lambda) \otimes F(\lambda)\, d\lambda\,,$$

where the $m \times m$ matrix $E_n(z)$ is defined by (2.8). By [13, Lemma 2.4], for $n \ge 1$, $E_n(z)$ is the rank-1 matrix given by

$$\begin{aligned} E_n(z) &= r(z)^{n-1} (I - z\mathcal{O}\!\iota)^{-1} \mathbb{1} b^T (I - z\mathcal{O}\!\iota)^{-1} \\ &= r(z)^{-1}\, (I - z\mathcal{O}\!\iota)^{-1} \mathbb{1}\, e_n(z). \end{aligned}$$

These relations permit us to proceed for the history term of (4.5) as we did for (2.7). We split the stage vector v_n as

$$v_n = a_n + v_n^{(0)} + \cdots + v_n^{(L)} \qquad \text{with} \qquad v_n^{(\ell)} = \sum_{j=b_\ell}^{b_{\ell-1}-1} W_{n-j}\, g_j$$

and obtain, like in (4.1),

$$v_n^{(\ell)} = \frac{1}{2\pi i} \int_{\Gamma_\ell} r(h\lambda)^{n-b_{\ell-1}} \, (I - h\lambda\mathcal{O\!\iota})^{-1} 1 \otimes F(\lambda)\, y^{(\ell)}(h\lambda)\, d\lambda\,,$$

where $y^{(\ell)}(h\lambda)$ is again the Runge–Kutta approximation at $t = b_{\ell-1}h$ to the initial-value problem (4.2), now for the inhomogeneity values $g_j = \big(g(t_j + c_i h, v_{ji})\big)_{i=1}^m$ in place of $g_j = \big(g(t_j + c_i h)\big)_{i=1}^m$. For $\ell \geq 2$ or 3, we thus approximate $v_n^{(\ell)}$ as

$$v_n^{(\ell)} \doteq \sum_{k=-K}^{K} w_k^{(\ell)}\, r(h\lambda_k^{(\ell)})^{n-b_{\ell-1}} \, (I - h\lambda_k^{(\ell)}\mathcal{O\!\iota})^{-1} 1 \otimes F(\lambda_k^{(\ell)})\, y^{(\ell)}(h\lambda_k^{(\ell)})\,.$$

The algorithm stores the same values as before. The memory requirements for the algorithm are thus independent of the number of stages m and remain essentially the same as in the pure convolution case.

The effect of the (exponentially small in K) quadrature error of the contour integrals on the convolution quadrature approximation of the integral equation can be estimated using a discrete Gronwall inequality or by stability estimates of the convolution quadrature as, e.g., in [1]. Since the emphasis of the present paper is on the algorithmic aspects, we do not work out details of such estimates here.

5. Numerical experiments.

We give two examples to illustrate the application and behavior of the fast convolution algorithm.

5.1. A nonlinear Volterra equation.

We consider a nonlinear Volterra integral equation with weakly singular kernel from [6],

$$u(t) = -\int_0^t \frac{\big(u(\tau) - \sin(\tau)\big)^3}{\sqrt{\pi(t-\tau)}}\, d\tau\,. \tag{5.1}$$

The convolution quadrature based on the backward Euler method gives the implicit discretization

$$u_{n+1} = \sum_{j=0}^{n} \omega_{n-j} \big(u_{j+1} - \sin((j+1)h)\big)^3,$$

where ω_n is given by (1.3) with $F(s) = s^{-1/2}$ and $\delta(\zeta) = 1 - \zeta$. To solve the nonlinear equation in each time step we use Newton iterations. The history term is computed by the fast algorithm of the previous section.

We consider also the discretizations based on the backward differentiation method of order 2 (cf. section 2.1) and on the two- and three-stage Radau IIA implicit Runge–Kutta methods of orders 3 and 5, respectively; see sections 2.2 and 4.2.

In the numerical experiment we use the base $B = 5$ and the Talbot contours with $K = 15$ and $K = 30$ and the further parameters as in section 3.4. We choose

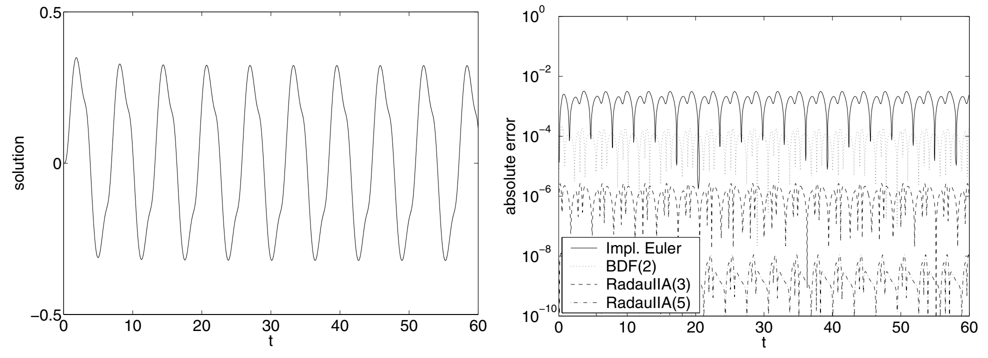

FIG. 5.1. *Evolution of the solution over the interval* $[0, 60]$ *(left) and logarithm of absolute error for different time integration methods (right) for* $h = 0.05$ *and* $K = 30$.

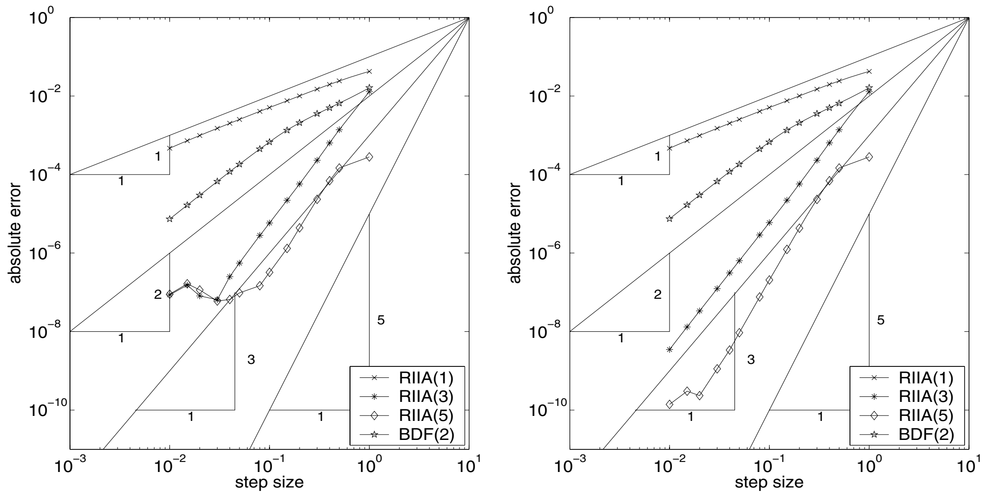

FIG. 5.2. *Absolute error versus step size h in logarithmic scale, for different integration methods, with* $K = 15$ *(left) and* $K = 30$ *(right).*

a tolerance of 10^{-12} in the Newton method. The error is calculated with respect to a reference solution, obtained with $h = 0.001$. Figure 5.1 shows the evolution of the absolute error and the oscillating solution u.

Figure 5.2 shows the errors $u_n - u(t_n)$ versus the step size h at time $t_n = 60$ for $K = 15$ and $K = 30$.

Figure 5.3 plots the cpu time versus the number of integration steps, up to 10^6 time steps. The near-linear growth of the computational work is clearly visible.

5.2. Fractional diffusion with transparent boundary conditions. Here we consider a fractional diffusion equation on the real line; see, e.g., [15] for applications of such equations in physics and for numerous references. The equation can be formulated as

$$u(x,t) - u_0(x) = \int_0^t \frac{(t-\tau)^{\alpha-1}}{\Gamma(\alpha)}\, \partial_{xx} u(x,\tau)\, d\tau + g(x,t) \qquad \text{for } x \in \ , \quad t > 0, \tag{5.2}$$

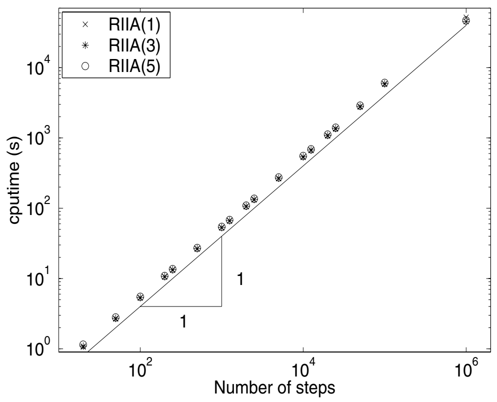

FIG. 5.3. *Cpu time in seconds versus the number of integration steps.*

with the asymptotic condition $u(x,t) \to 0$ for $x \to \pm\infty$, for an inhomogeneity g with $g(x,0) = 0$. To reduce the computation to a finite domain $x \in [-a,a]$ for initial data u_0 and inhomogeneity g with support in $[-a,a]$, we impose transparent boundary conditions at $x = \pm a$, which read

$$u(\pm a,t) = -\int_0^t \frac{(t-\tau)^{\alpha/2-1}}{\Gamma(\alpha/2)}\,\partial_\nu u(\pm a,\tau)\,d\tau, \tag{5.3}$$

with the outward derivative $\partial_\nu = \pm\partial_x$ at $x = \pm a$. These boundary conditions are derived with Laplace transform techniques in the same way as for the wave or the Schrödinger equation; see, e.g., [2]. Space discretization of (5.2) is done using second-order finite differences and a central finite difference to approximate the normal derivative. With the notation

$$\delta_{xx} u_l^n = \frac{1}{\Delta x^2}\left(u_{l-1}^n - 2u_l^n + u_{l+1}^n\right), \quad \delta_\nu u_{\pm(M-1)}^n = \frac{1}{2\Delta x}\left(u_{\pm M}^n - u_{\pm(M-2)}^n\right)$$

for $a = M\Delta x$, the discrete equation approximating (5.2) is

$$\begin{aligned} u_l^{n+1} - u_l^0 &= \sum_{j=0}^n \omega_{n-j}^{(\alpha)}\,\delta_{xx} u_l^{j+1} + g_l^n \qquad \text{for } l = -(M-1),\dots,M-1\ ;\ n>0, \\ u_{\pm(M-1)}^{n+1} &= -\sum_{j=0}^n \omega_{n-j}^{(\alpha/2)}\,\delta_\nu u_{\pm(M-1)}^{j+1}\ , \end{aligned} \tag{5.4}$$

where the weights $\omega_n^{(\beta)}$ are the convolution quadrature weights of the backward Euler method for the kernel $f(t) = t^{\beta-1}/\Gamma(\beta)$ with Laplace transform $F(s) = s^{-\beta}$. In this case $g_l^n = g(x_l, t_n + h)$.

In the numerical example we set $a = 5$ and $M = 450$. We consider the problem with $\alpha = 2/3$ and no inhomogeneity, i.e., $g \equiv 0$. The initial value is $u(x,0) = \exp(-x^2)$. Figure 5.4 shows the errors at $t = 2$ and $t = 10$ in dependence on the step size for the convolution quadratures based on the Radau IIA methods of orders 1, 3, 5, obtained with $B = 5$ and $K = 15$ in the fast convolution algorithm, by using the hyperbola contours. The reference solution is obtained with the convolution quadrature based on the Radau IIA method of order 5, with a small step size $h = 0.0002$

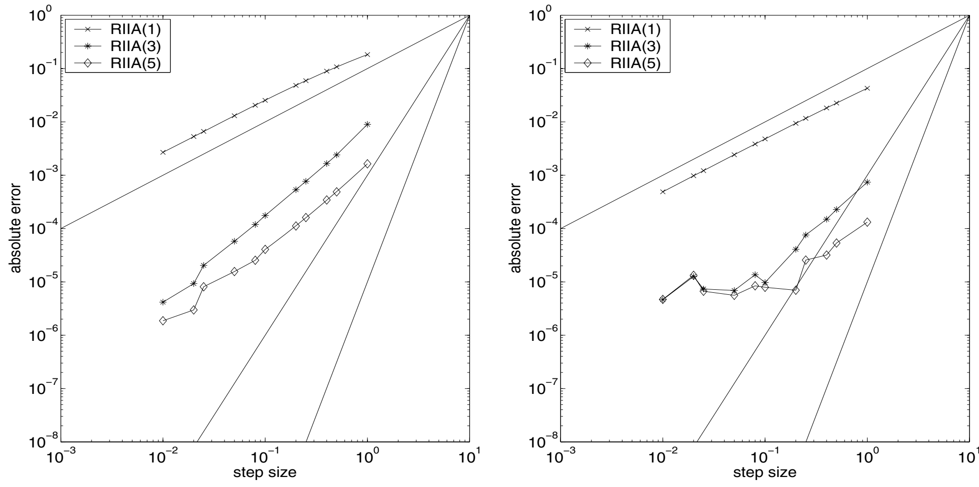

Fig. 5.4. *Absolute error versus time step in logarithmic scale, for different integration methods, with $K = 15$. Left $t = 2$, right $t = 10$.*

and large number $K = 40$ of nodes on the integration contour. We observe an order reduction for the higher-order methods, which is due to the temporal nonsmoothness of the solution at $t = 0$; cf. [1, sect. 8]. Nevertheless, the higher-order methods give much better accuracy.

The work diagram looks almost identical to Figure 5.3, showing practically linear dependence of the computational work on the number of time steps. The required memory is less than 200 entries per spatial grid point for up to $N \leq 10^4$ steps and less than 300 entries per grid point for $N \leq 10^6$ steps. These numbers are halved if we run the algorithm with $B = 10$, $K = 10$ instead of $B = 5$, $K = 15$, as is sufficient for less stringent accuracy requirements ($\sim 10^{-3}$).

Acknowledgment. We thank Cesar Palencia for helpful discussions during the preparation of this paper.

REFERENCES

[1] E. Cuesta, C. Lubich, and C. Palencia, *Convolution quadrature time discretization of fractional diffusion-wave equations*, Math. Comp., 75 (2006), pp. 673–696.

[2] T. Hagstrom, *Radiation boundary conditions for numerical simulation of waves*, Acta Numer., 8 (1999), pp. 47–106.

[3] E. Hairer, C. Lubich, and M. Schlichte, *Fast numerical solution of nonlinear Volterra convolution equations*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 532–541.

[4] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations.* II. *Stiff and Differential-Algebraic Problems*, 2nd ed., Springer, Berlin, 1996.

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

[6] N. Levinson, *A nonlinear Volterra equation arising in the theory of superfluidity*, J. Math. Anal. Appl., 1 (1960), pp. 1–11.

[7] M. López-Fernández, C. Lubich, C. Palencia, and A. Schädle, *Fast Runge-Kutta approximation of inhomogeneous parabolic equations*. Numer. Math., 102 (2005), pp. 277–291.

[8] 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.

[9] M. López-Fernández, C. Palencia, and A. Schädle, *A spectral order method for inverting sectorial Laplace transforms*, SIAM J. Numer. Anal., to appear.

[10] C. Lubich, *Convolution quadrature and discretized operational calculus.* I, Numer. Math., 52 (1988), pp. 129–145.

[11] C. Lubich, *Convolution quadrature and discretized operational calculus.* II, Numer. Math., 52 (1988), pp. 413–425.

[12] C. Lubich, *Convolution quadrature revisited*, BIT, 44 (2004), pp. 503–514.

[13] C. Lubich and A. Ostermann, *Runge-Kutta methods for parabolic equations and convolution quadrature*, Math. Comput., 60 (1993), pp. 105–131.

[14] C. Lubich and A. Schädle, *Fast convolution for nonreflecting boundary conditions*, SIAM J. Sci. Comp., 24 (2002), pp. 161–182.

[15] R. Metzler and J. Klafter, *The random walk's guide to anomalous diffusion: A fractional dynamics approach*. Phys. Rep., 339 (2000), pp. 1–77.

[16] 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.

[17] A. Schädle, *Ein Schneller Faltungsalgorithmus für Nichtreflektierende Randbedingungen*, Doctoral Thesis, University of Tübingen, Germany, 2002.

[18] F. Stenger, *Approximations via Whitaker's cardinal function*, J. Approx. Theory, 17 (1976), pp. 222–240.

[19] F. Stenger, *Numerical methods based on Whitaker cardinal, or sinc functions*, SIAM Rev., 23 (1981), pp. 165–224.

[20] A. Talbot, *The accurate numerical inversion of Laplace transforms.* J. Inst. Math. Appl., 23 (1979), pp. 97–120.