Adv Comput Math (2021) 47: 81
https://doi.org/10.1007/s10444-021-09902-6

A fast and oblivious matrix compression algorithm for Volterra integral operators

J. Dölz[1] · H. Egger[2] · V. Shashkov[2]

Received: 24 March 2021 / Accepted: 14 September 2021 / Published online: 26 October 2021

Abstract

The numerical solution of dynamical systems with memory requires the efficient evaluation of Volterra integral operators in an evolutionary manner. After appropriate discretization, the basic problem can be represented as a matrix-vector product with a lower diagonal but densely populated matrix. For typical applications, like fractional diffusion or large-scale dynamical systems with delay, the memory cost for storing the matrix approximations and complete history of the data then becomes prohibitive for an accurate numerical approximation. For Volterra integral operators of convolution type, the *fast and oblivious convolution quadrature* method of Schädle, Lopez-Fernandez, and Lubich resolves this issue and allows to compute the discretized evaluation with N time steps in $O(N \log N)$ complexity and only requires $O(\log N)$ active memory to store a compressed version of the complete history of the data. We will show that this algorithm can be interpreted as an $\mathcal{H}$-matrix approximation of the underlying integral operator. A further improvement can thus be achieved, in principle, by resorting to $\mathcal{H}^2$-matrix compression techniques. Following this idea, we formulate a variant of the $\mathcal{H}^2$-matrix-vector product for discretized Volterra integral operators that can be performed in an evolutionary and oblivious manner and requires only $O(N)$ operations and $O(\log N)$ active memory. In addition to the acceleration, more general asymptotically smooth kernels can be treated and the algorithm does not require a priori knowledge of the number of time steps. The efficiency of the proposed method is demonstrated by application to some typical test problems.

Keywords Volterra integral operators · Convolution quadrature · $\mathcal{H}^2$-matrices · Matrix compression

Communicated by: Michael O'Neil

This article belongs to the Topical Collection: *Advances in Computational Integral Equations*
Guest Editors: Stephanie Chaillat, Adrianna Gillman, Per-Gunnar Martinsson, Michael O'Neil, Mary-Catherine Kropinski, Timo Betcke, Alex Barnett

✉ J. Dölz
doelz@ins.uni-bonn.de

Extended author information available on the last page of the article.

Mathematics Subject Classification (2010) 65D20 · 45D05

1 Introduction

We study the numerical solution of dynamical systems with memory which can be modelled by abstract Volterra integro-differential equations of the form

$$\alpha(t)y'(t) + A(t)y(t) = \int_0^t k(t,s)f(s,y(s))\,\mathrm{d}s, \qquad 0 \le t \le T. \tag{1}$$

Such problems arise in a variety of applications, e.g., in anomalous diffusion [32], neural sciences [2], transparent boundary conditions [1, 20, 22, 23], wave propagation [1, 12, 20], field circuit coupling [13], and many more, see also [7, 8, 29, 34] and the references therein. The simplest model problem which already shares the essential difficulties stemming from non-locality of the right-hand side in (1) is the evaluation of the integral operator

$$y(t) = \int_0^t k(t,s)f(s)\ \mathrm{d}s, \qquad 0 \le t \le T, \tag{2}$$

with kernel function k, data f, and result function y. Let us emphasize that, in order to allow the application in the context of integro-differential problems (1), the parameter-dependent integrals (2) have to be evaluated in an *evolutionary* manner, i.e., for successively increasing time. The results obtained for (2) then quite naturally extend to (1). We hence focus on the evaluation of Volterra integral operators (2) in the following and return to more general problems in Section 5.

1.1 Discretization and related work

After applying some appropriate discretization procedure, see [7] for a survey, problem (2) can be phrased as a simple matrix-vector multiplication

$$\mathtt{y}_n = (\mathtt{K}\mathtt{f})_n, \qquad 1 \le n \le N. \tag{3}$$

The evolutionary character and the nonlocal interactions are reflected by the fact that the matrix $\mathtt{K} \in \mathbb{R}^{N\times N}$ is lower block triangular but densely populated. The straight-forward computation of the result vector $\mathtt{y}$ requires $O(N^2)$ algebraic operations. The evolutionary character of problem (2) can be preserved by computing the entries $\mathtt{y}_n$ for $n = 1, \dots, N$ recursively, i.e., by traversing the matrix $\mathtt{K}$ from top to bottom. If, on the other hand, the matrix $\mathtt{K}$ is traversed from left to right, then the algorithm becomes *oblivious*, i.e., the data $\mathtt{f}_n$ is only required in the nth step of the algorithm, but the execution of (3) then requires $O(N)$ active memory to store the partial sums for every row. Although the evaluation can then still be organized in an evolutionary manner, see Section 2.2, the number of time steps N needs to be fixed a priori in order to store the intermediate results.

For the particular case that the integral kernel in (2) is of convolution type

$$k(t,s) = k(t-s), \tag{4}$$

a careful discretization of (5) gives rise to an algebraic system (3) with block Toeplitz matrix $\mathtt{K}$, and the discrete solution $\mathtt{y}$ can be computed in $O(N \log N)$ operations using fast Fourier transforms. As shown in [21], an evolutionary version of the matrix vector product can be realized in $O(N \log^2 N)$ complexity and requiring $O(N)$ active memory. The convolution quadrature methods of [27, 28, 30] treat the case that only the Laplace transform $\hat{k}(s)$ of the convolution kernel (4) is available. The *fast and oblivious convolution quadrature* method introduced in [26, 31] allows the efficient evaluation of Volterra integrals with convolution kernel in an *evolutionary* and *oblivious* manner with $O(N \log N)$ operations and only $O(\log N)$ active memory and $O(\log N)$ evaluations of the Laplace transform $\widehat{k}(s)$. This method is close to optimal concerning complexity and memory requirements and has been applied successfully to the numerical solution of partial differential equations with transparent boundary conditions [20], the efficient realization of boundary element methods for the wave equation [34], or fractional diffusion [9].

For integral operators (2) with general kernels $k(t, s)$, the abovementioned methods cannot be applied directly. Alternative approaches, like the fast multipole method [14, 16, 33], the panel clustering technique [19], $\mathcal{H}$- and $\mathcal{H}^2$-matrices [5, 17], multilevel techniques [6, 15], or wavelet algorithms [10], which were developed and applied successfully in the context of molecular dynamics and boundary integral equations, are however still applicable. These methods are based on certain hierarchical approximations for the kernel functions $k(t, s)$, whose error can be controlled under appropriate smoothness assumptions, e.g., if the kernel is *asymptotically smooth*; see (27) for details. If the data f is independent of the solution y, the numerical evaluation of the Volterra integral operator (2) can then be realized with $O(N \log^\alpha N)$ computational cost with some $\alpha \geq 0$ and N again denoting the dimension of the underlying discretization. Moreover, data-sparse approximations of the matrix $\mathtt{K}$ for asymptotically smooth kernels $k(t, s)$ can be stored efficiently with only $O(N \log^\alpha N)$ memory and for convolution kernels $k(t-s)$ even with $O(\log N)$ memory; we refer to [5, 18] for details and an extensive list of references. Unfortunately, the algorithms mentioned in literature are not evolutionary and, therefore, cannot be applied to more complex problems like (1) directly.

1.2 A fast and oblivious evolutionary algorithm

In this paper, we propose an algorithm for the efficient evaluation of Volterra integrals (2) or corresponding matrix-vector products (3) which shares the benefits and overcomes the drawbacks of the approaches mentioned above, i.e., it is

- *Evolutionary*: the approximations $\mathtt{y}_n$ can be computed one after another and the number of time steps N does not need to be known in advance,
- *Oblivious*: the entry $\mathtt{f}_n$ of the right-hand side is only required in the nth step,
- *Fast*: the evaluation of all $\mathtt{y}_n$, $1 \leq n \leq N$ requires only $\mathcal{O}(N)$ operations, and
- *Memory efficient*: the storage of the convolution matrix requires only $\mathcal{O}(N)$ memory for general and $\mathcal{O}(\log N)$ memory for convolution kernels. The matrix entities can also be computed on the fly, such that only $\mathcal{O}(\log N)$ storage is required to store a compressed history of the data f.

Our strategy is based on the ideas of polynomial-based $\mathcal{H}^2$-compression algorithms for finding hierarchical low-rank approximations of the kernel function $k(t, s)$ leading to a block-structured hierarchical approximation of the matrix $\mathtt{K}$ in (3). The accuracy of the underlying approximation can thus be guaranteed by well-known approximation results; see [5, 18] for instance. A key ingredient for our considerations is the one-dimensional nature of the integration domain which allows to characterize the block structure of the approximating hierarchical matrix explicitly. This allows us to formulate an algorithm which traverses the compressed matrix $\mathtt{K}$ from top to bottom in accordance with the evolutionary structure of the underlying problem. The hierarchical approximation of the convolution kernel also yields a compression strategy for the history of the data f. In this sense, our algorithm can be considered a generalization of [1, 4, 22, 23], where a fast multipole expansion was employed to accelerate the *sum of exponentials approach*, or to [24], where a polynomial on growing time steps was employed for the compression of the data, as well as to [25], where an evolutionary $\mathcal{H}$-matrix approximation with a special low-rank structure was constructed. As a further result, we show that our algorithm seamlessly integrates into the convolution quadrature framework of [27, 28], when the kernel $k(t - s)$ is of convolution type and only accessible via its Laplace transform. In analogy to the treatment of nearfield contributions in the fast boundary element method, we utilize standard convolution quadrature to compute the entries of the convolution matrix close to the diagonal, while numerical inverse Laplace transforms [11] are used to set up an $\mathcal{H}^2$-approximation of the remaining off-diagonal parts of the convolution matrix in the time domain. This approach has some strong similarities to the fast and oblivious convolution quadrature method [31, 35], but we will reveal some important differences. In particular, we illustrate that the methods of [31, 35] can actually be interpreted as $\mathcal{H}$-matrix approximations with a particular organization of the matrix-vector product in (3), which shows that the $\mathcal{O}(N \log N)$ complexity cannot be further improved. Moreover, the convolution matrix must be applied from left to right to allow for an oblivious evaluation and the number of time steps N must be known in advance. In contrast to that, our new algorithm is based on an $\mathcal{H}^2$-approximation of the matrix $\mathtt{K}$ and the evolutionary, fast, and oblivious evaluation of the matrix-vector product can be realized by traversing through the matrix from top to bottom in $\mathcal{O}(N)$ complexity and without needing to know the number of time steps N in advance. Finally, our algorithm naturally extends to general integral kernels $k(t, s)$ increasing the field of applications substantially.

1.3 Outline

In Section 2, we recall some general approximation results, introduce our basic notation, and state a slightly modified algorithm for the dense evaluation of the Volterra integral operators to illustrate some basic principles that we exploit later on. Section 3 is concerned with a geometric partitioning on the domain of integration, the multilevel hierarchy used for the $\mathcal{H}^2$-compression, and the description and analysis of our new algorithm. In Section 4, we consider convolution kernels $\hat{k}(s)$ and discuss the relation of our algorithm to Lubich's convolution quadrature and the connections to the fast and oblivious algorithm of [31, 35]. To support our theoretical considerations, some numerical results are provided in Section 5.

2 Preliminary results

Let us start with summarizing some basic results about typical discretization strategies for Volterra integral operators

$$y(t) = \int_0^t k(t,s) f(s) ds \tag{5}$$

which are the basis for the efficient and reliable numerical evaluation later on. For simplicity, all functions y, f, and k are assumed to be scalar valued. We will demonstrate the application to more general problems of the form (1) in Section 5.

2.1 A general approximation result

For the discretization of the integral operator (5), we consider methods of the form

$$\widetilde{y}_h(t) = \int_0^t k_h(t,s) f_h(s) ds, \tag{6}$$

where k_h and f_h are suitable approximations for k and f. The subscript h will be used to designate approximations throughout. The following result may serve as a theoretical justification for a wide variety of particular discretization schemes.

Lemma 1 *Let $T > 0$, kernels $k, k_h \in L^\infty(0,T; L^r(0,T))$, and $f, f_h \in L^{r'}(0,T)$ be given with $1 \le r, r' \le \infty$ with $1/r + 1/r' = 1$. Further assume that*

$$\|k - k_h\|_{L^\infty(0,T;L^r(0,T))} \le \varepsilon \qquad \text{and} \qquad \|f - f_h\|_{L^{r'}(0,T)} \le \varepsilon. \tag{7}$$

Then, the functions y, $\widetilde{y}_h$ defined by (5) and (6) satisfy

$$\|y - \widetilde{y}_h\|_{L^\infty(0,T)} \le C \left(\|k\|_{L^r(0,T)} + \|f\|_{L^{r'}(0,T)} + \varepsilon \right) \varepsilon, \tag{8}$$

i.e., the error in the results can be bounded uniformly by the perturbation in the data.

Proof From Hölder's inequality, we can deduce that

$$\begin{aligned} |y(t) - \widetilde{y}_h(t)| &\le \int_0^t |k(t,s)||f(s) - f_h(s)| + |k(t,s) - k_h(t,s)||f_h(s)| ds \\ &\le \|k(t,\cdot)\|_{L^r(0,T)} \|f - f_h\|_{L^{r'}(0,T)} + \|k(t,\cdot) - k_h(t,\cdot)\|_{L^r(0,T)} \|f_h\|_{L^{r'}(0,T)}. \end{aligned}$$

The result then follows by estimating $\|f_h\| \le \|f\| + \|f - f_h\|$, using the estimates for the differences in the data, and taking the supremum over all $0 < t < T$. □

Remark 1 The constant C in the estimate (8) depends on the kernel k, but is independent of T. The result can therefore be applied to time intervals of arbitrary size.

Without substantially changing the argument, it is possible to obtain similar estimates also in other norms. In many cases, $\widetilde{y}_h$ only serves as an intermediate result and the final approximation is given by $y_h(t) = (P_h \widetilde{y}_h)(t)$, where P_h is some projection or interpolation operator; a particular case will be discussed in more detail below. Estimates for the error $\|y - y_h\|$ can then be obtained by additionally taking into account the projection errors.

2.2 Piecewise polynomial approximations

Many discretization methods for integral or integro-differential equations, e.g., collocation or Galerkin methods [7], are based on piecewise polynomial approximations and fit into the abstract form mentioned above. As a particular example and for later reference, we consider such approximations in a bit more detail now.

Let $h > 0$ be given, define $t^n = nh$, $n \geq 0$, and set $T = t^N = Nh$. We set $I^n = [t^{n-1}, t^n]$ for $1 \leq n \leq N$, and denote by $\mathcal{T}_h = \{I^n : 1 \leq n \leq N\}$ the resulting uniform mesh of the interval $[0, T]$. We write $\mathcal{P}_p(a, b)$ for the space of polynomials of degree at most p over the interval (a, b), and $\mathcal{P}_{q,q}((a, b) \times (c, d)) = \mathcal{P}_q(a, b) \otimes \mathcal{P}_q(c, d)$ for the space of polynomials in two variables of degree at most q in each variable. We further define piecewise polynomial spaces

$$\mathcal{P}_p(\mathcal{T}_h) = \left\{ f \in L^1(0, T) : f|_{I^n} \in \mathcal{P}_p(I^n) \right\}, \tag{9}$$

$$\mathcal{P}_{q,q}(\mathcal{T}_h \times \mathcal{T}_h) = \left\{ k \in L^1((0, T) \times (0, T)) : k|_{I^m \times I^n} \in \mathcal{P}_{q,q}\left(I^m \times I^n\right) \right\}, \tag{10}$$

over the grid $\mathcal{T}_h$ and the tensor-product grid $\mathcal{T}_h \times \mathcal{T}_h$.

For sufficiently regular functions f and k over the mesh $\mathcal{T}_h$ and $\mathcal{T}_h \times \mathcal{T}_h$, piecewise polynomial approximations k_h, f_h satisfying (7) can be found by appropriate interpolation and choosing the mesh size h small enough. Without further structural assumptions on the data, it seems natural to use uniform grids $\mathcal{T}_h$, which can be obtained, e.g., by uniform refinement of some reference grid. In Fig. 1, we depict the resulting uniform partitions for approximation of the kernel function k.

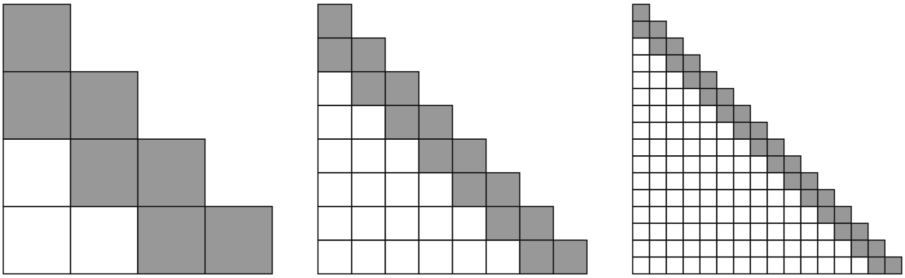

Fig. 1 Uniformly refined grids $\mathcal{T}_h \times \mathcal{T}_h$ for approximation of k. Only the elements required for approximating $k(t, s)$ for $s \leq t$ are depicted. The grid cells near the diagonal $t = s$, i.e, the *nearfield*, play a special role and are thus colored in gray

The evaluation of $\widetilde{y}_h$ defined by (6) can be split into two contributions

$$\widetilde{y}_h(t) = \widetilde{w}_h(t) + \widetilde{z}_h(t) \tag{11}$$

with w_h corresponding to the integrals over the *farfield* cells and z_h over the *nearfield* cells, which are depicted in white and gray in Fig. 1, respectively. As discussed in the following subsection, the numerical treatment of these contributions differs slightly.

2.3 Practical realization

From equation (6) and the choice of $f_h \in \mathcal{P}_p(\mathcal{T}_h)$ and $k_h \in \mathcal{P}_{q,q}(\mathcal{T}_h \times \mathcal{T}_h)$, one can see that $\widetilde{y}_h$ is a piecewise polynomial of degree $\leq p + q + 1$ over the grid $\mathcal{T}_h$. It is often convenient to replace $\tilde{y}_h$ by a piecewise polynomial $y_h \in \mathcal{P}_p(\mathcal{T}_h)$ with the same degree as the data. For this purpose, we simply choose a set of distinct points $0 \leq \gamma_j \leq 1$, $0 \leq j \leq p$, and define $y_h \in \mathcal{P}_p(\mathcal{T}_h)$ by collocation

$$y_h\left(t_j^n\right) = \widetilde{y}_h\left(t_j^n\right), \qquad 0 \leq j \leq p, \tag{12}$$

on every time interval $I^n \in \mathcal{T}_h$, with collocation points $t_j^n = t^{n-1} + \gamma_j h$, $j = 0, \ldots, p$. Now let ψ_j^n, $j = 0, \ldots, p$, denote the Lagrangian basis of $\mathcal{P}_p(I^n)$ with respect to the interpolation points t_j^n, i.e.,

$$\psi_i^n\left(t_j^n\right) = \delta_{i,j}, \qquad 0 \leq i \leq p, \quad 1 \leq n \leq N. \tag{13}$$

Then as a consequence of the uniformity of the mesh $\mathcal{T}_h$, one may deduce that

$$\psi_i^n\left(t - t^n\right) = \psi_i^m\left(t - t^m\right), \qquad 0 \leq i \leq p, \quad 1 \leq m, n \leq N, \tag{14}$$

i.e., the basis $\{\psi_i^n\}_{0 \leq i \leq p}$ is *invariant under translation*, which will become an important ingredient for our algorithm below. The approximate data f_h and the discrete solution y_h can now be expanded as

$$y_h(t) = \sum_{j=0}^{p} y_j^n \psi_j^n(t), \qquad f_h(t) = \sum_{j=0}^{p} f_j^n \psi_j^n(t), \qquad \text{for } t \in I^n. \tag{15}$$

In a similar manner, the approximate kernel function k_h can be expanded with respect to a set of bases $\{\varphi_i^n\}_{i=0,\ldots,q}$ for the spaces $\mathcal{P}_q(I^n)$, which leads to

$$k_h(s, t) = \sum_{i=0}^{q} \sum_{j=0}^{q} k_{i,j}^{m,n} \varphi_i^m(s) \varphi_j^n(t), \qquad \text{for } s \in I^m,\ t \in I^n. \tag{16}$$

We will again assume translation invariance of this second basis, i.e.,

$$\varphi_i^n\left(t - t^n\right) = \varphi_i^m\left(t - t^m\right), \qquad 0 \leq i \leq q, \quad 1 \leq m, n \leq N. \tag{17}$$

Note that we allow for different polynomial degrees $q \neq p$ in the approximations y_h, f_h, and k_h, and hence two different sets of basis functions are required. Evaluating (6) at time $t = t_j^m$ and utilizing (12), we obtain

$$y_h(t_j^m) = \sum_{n=1}^{m-2} \int_{I^n} k_h\left(t_j^m, s\right) f_h(s)\,\mathrm{d}s + \int_{t_j^{m-2}}^{t_j^m} k_h\left(t_j^m, s\right) f_h(s)\,\mathrm{d}s, \tag{18}$$

where we used the splitting of the integration domain $(0, t_j^m)$ into subintervals of the mesh $\mathcal{T}_h$ and an additional separation of farfield and nearfield contributions; see Fig. 1. By inserting the basis representations (15) and (16) for y_h, f_h, and k_h, the integrals in the farfield contribution can be expressed as

$$\int_{I^n} k_h\left(t_j^m, s\right) f_h(s)\,\mathrm{d}s = \sum_{i=0}^{q} \varphi_i^m\left(t_j^m\right) \sum_{k=0}^{q} k_{i,k}^{m,n} \sum_{r=0}^{p} \int_{I^n} \varphi_k^n(s)\psi_r^n(s)\,\mathrm{d}s\, f_r^n. \tag{19}$$

For convenience of presentation, let us introduce the short-hand notations

$$P_{i,j} = \varphi_i^m\left(t_j^m\right), \qquad Q_{k,r} = \int_{I^n} \varphi_k^n(s)\psi_r^n(s)\,\mathrm{d}s, \tag{20}$$

and note that the corresponding matrices P and Q are independent of the time steps m, n, due to the translation invariance conditions (14) and (17). The result vector y^m containing entries $y_j^m = y_h\left(t_j^m\right)$ from (18) can then be expressed as

$$y^m = w^m + z^m \tag{21}$$

with farfield contribution w^m given by

$$w^m = Pu^m, \qquad u^m = \sum_{n=1}^{m-2} k^{m,n} g^n, \qquad g^n = Qf^n, \tag{22}$$

where $k^{m,n}$ is the matrix containing the entries $k_{i,j}^{m,n}$. Further introducing the symbols $K^{m,n} = Pk^{m,n}Q$, this may be stated equivalently as $w^m = \sum_{n=0}^{m-2} K^{m,n} f^n$. In a similar manner, we may represent the nearfield contribution z^m by

$$z^m = K^{m,m-1} f^{m-1} + K^{m,m} f^m, \tag{23}$$

with appropriate nearfield matrices $K^{m,m-1}$, $K^{m,m} \in \mathbb{R}^{(p+1)\times(p+1)}$.

We denote by $\mathtt{y}$, $\mathtt{f} \in \mathbb{R}^{N(p+1)}$ the global vectors that are obtained by stacking the element contributions y^m, f^n together. The computation of $\mathtt{y}$ can then be written as matrix-vector product $\mathtt{y} = \mathtt{K}\mathtt{f}$ with block-matrix $\mathtt{K} \in \mathbb{R}^{N(p+1)\times N(p+1)}$ consisting of blocks $K^{m,n}$ as defined above. A possible block-based implementation of this matrix-vector product can be realized as in Algorithm 1.

Remark 2 At first sight, this algorithm may seem more complicated than actually required. In fact, after generating the matrix blocks $K^{m,n} = Pk^{m,n}Q$, the mth component of the result vector could also be computed as $y^m = \sum_{n=1}^{m} K^{m,n} f^n$. The above version, however, is closer to the algorithm developed in the next section. Moreover, it is *evolutionary*, i.e., the entries of the vector $\mathtt{y}$ are computed one after another, and *oblivious* in the sense that only the blocks f^{m-1} and f^m are needed for

Algorithm 1 Evaluation of Volterra integral operators for uniform meshes.

```
for $m = 1, \dots, N$ do
    $u = 0$
    for $n = 1, \dots, m-2$ do
        $u = u + k^{m,n} g^n$
    end for
    $g^m = Qf^m$
    $w^m = Pu$
    $z^m = K^{m,m-1} f^{m-1} + K^{m,m} f^m$
    $y^m = w^m + z^m$
end for
```

the computation of y^m. Note, however, that all auxiliary values g^n, $n = 1, \dots, m-2$ are required to compute the block y^m and therefore have to be kept in memory. This will be substantially improved in Section 3.2.

2.4 Computational complexity

As indicated above, the computation of y_h according to (18) can be phrased in algebraic form as a matrix-vector product

$$\mathtt{y} = \mathtt{K}\mathtt{f}, \tag{24}$$

with $\mathtt{y}$ and $\mathtt{f}$ denoting the coefficient vectors for y_h and f_h, and a lower block triangular matrix $\mathtt{K} \in \mathbb{R}^{N(p+1)\times N(p+1)}$. Note that the pattern of the matrix $\mathtt{K}$ is structurally the same as that of the tensor-product grid underlying the approximation of the kernel function k; see Fig. 1, with each cell corresponding to a block of size $(p+1)\times(p+1)$. Thus, the the computation of $\mathtt{y} = \mathtt{K}\mathtt{f}$ will in general require $O(p^2N^2)$ operations and $O(p^2N^2)$ memory to store the matrix $\mathtt{K}$. In addition, we require $O(pN)$ active memory to store two values of f^n and the history of g.

3 A fast and oblivious algorithm

The aim of this section is to introduce a novel algorithm which allows for a simultaneous compression of the matrix $\mathtt{K}$ used for the evaluation of (24) and the history of the data stored in the vectors g^n, $n \geq 1$. The underlying approximation is that of $\mathcal{H}^2$-matrix compression techniques [5, 18]. We first collect some results about these hierarchical approximations and then state and analyze our algorithm.

3.1 Multilevel partitioning

For ease of presentation, we will assume that the number of time steps is given as $N = 2^L$ with $L \in \mathbb{N}$ and define $h = T/N$. Now let $I^{(n;1)} = I^n$ and introduce a

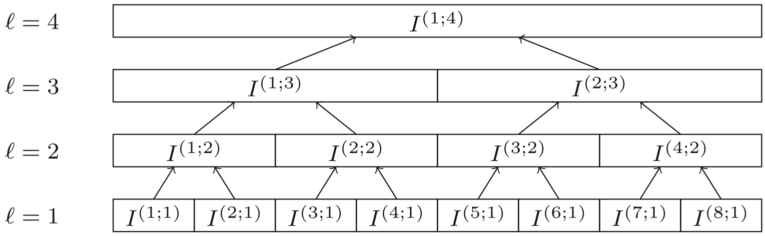

Fig. 2 Mesh hierarchy obtained by recursive coarsening of intervals $I^n = I^{(n;1)}$ with maximal coarsening level $L = 3$ and $N = 2^L = 8$ fine grid cells

hierarchy of nested partitions into subintervals

$$I^{(n;\ell)} = I^{(2n-1;\ell-1)} \cup I^{(2n;\ell-1)} = \left[t^{2^{\ell-1}(n-1)}, t^{2^{\ell-1}n}\right], \qquad \ell > 1,$$

of length $2^{\ell-1}h$ by recursive coarsening of the intervals; see Fig. 2 for a sketch. From the hierarchic construction, one can immediately see that

$$I^n \subset I^{(C(n;\ell);\ell)} \qquad \text{for} \quad C(n;\ell) := \left\lceil n/2^{\ell-1} \right\rceil, \tag{25}$$

where $\lceil r \rceil$ denotes the smallest integer larger or equal to r as usual.

In a similar spirit to [5, 14, 16, 18], we next introduce a multilevel partitioning of the support of the kernel k leading to adaptive hierarchical meshes

$$\begin{aligned} \mathcal{AT}_h = \Big\{ I^{(m;\ell)} \times I^{(n;\ell)} \colon \ell &= 1 \text{ with } n \in \{m-1, m\} \text{ or} \\ I^{(m;\ell)} \cap I^{(n;\ell)} &= \emptyset \text{ with } I^{(\lceil m/2 \rceil;\ell+1)} \cap I^{(\lceil n/2 \rceil;\ell+1)} \neq \emptyset \Big\}. \end{aligned} \tag{26}$$

Note that every element of $\mathcal{AT}_h$ is square and thus corresponds to one element of a, possibly coarser, uniform mesh $\mathcal{T}_{h'} \times \mathcal{T}_{h'}$. Moreover, any element in $\mathcal{AT}_h$ is the union of elements of the underlying uniform mesh $\mathcal{T}_h \times \mathcal{T}_h$ and can be constructed by recursive agglomeration or coarsening. As illustrated in Fig. 3, the hierarchic adaptive mesh $\mathcal{AT}_h$ can again be split into nearfield elements adjacent to the diagonal and the remaining farfield elements. Let us remark that the resulting partitioning and its splitting coincide with most of the classical partitioning strategies developed in

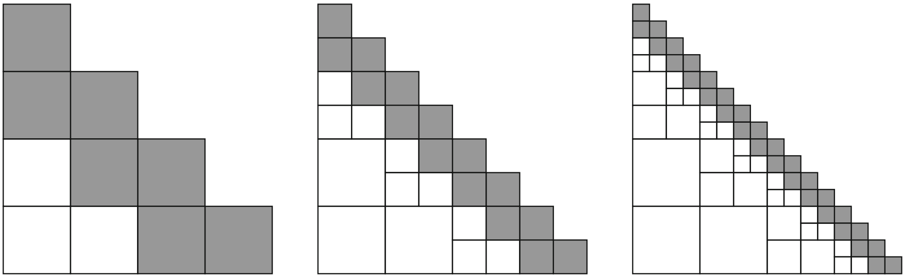

Fig. 3 Adaptive hierarchical meshes $\mathcal{AT}_h$ obtained by recursive coarsening of farfield cells in the corresponding uniformly refined meshes $\mathcal{T}_h \times \mathcal{T}_h$ in Fig. 1

the context of panel clustering and $\mathcal{H}$- and $\mathcal{H}^2$-matrices, see [18] and the references therein.

3.2 Adaptive data-sparse approximation

Let $\mathcal{P}_{q,q}(\mathcal{AT}_h)$ be the space of piecewise polynomials of degree $\leq q$ in each variable over the mesh $\mathcal{AT}_h$. Since the adaptive hierarchical mesh is obtained by coarsening of the underlying uniform grid $\mathcal{T}_h \times \mathcal{T}_h$, we certainly have

$$\mathcal{P}_{q,q}(\mathcal{AT}_h) \subset \mathcal{P}_{q,q}(\mathcal{T}_h \times \mathcal{T}_h).$$

Instead of a uniform approximation as used in Section 2.2, we now consider adaptive approximations $k_h \in \mathcal{P}_{q,q}(\mathcal{AT}_h)$ for the evaluation of (6) or (18).

Remark 3 Let us assume for the moment that the kernel k in (2) is *asymptotically smooth*, i.e., there exist constants $c_1, c_2 > 0$, $r \in \mathbb{R}$ such that

$$\left|\partial_t^\alpha \partial_s^\beta k(t,s)\right| \leq c_1 \frac{(\alpha+\beta)!}{c_2^{\alpha+\beta}} (t-s)^{r-\alpha-\beta} \tag{27}$$

for all α, $\beta \geq 0$ and all $t \neq s$. As shown in [5, 18], adaptive approximations $k_h \in \mathcal{P}_{q,q}(\mathcal{AT}_h)$ can be constructed for asymptotically smooth kernels, which converge exponentially in q in the farfield. As a consequence, the same level of accuracy required in Lemma 1 can be achieved by adaptive approximations with much less degrees of freedom than by uniform approximations.

Remark 4 It is not difficult to see that $\dim(\mathcal{P}_{q,q}(\mathcal{T}_h \times \mathcal{T}_h)) = \mathcal{O}(N^2q^2)$ while $\dim(\mathcal{P}_{q,q}(\mathcal{AT}_h)) = \mathcal{O}(Nq^2)$. The adaptive hierarchical approximation thus is *data-sparse* and leads to substantial savings in the memory required for storing the kernel approximation or its matrix representation (24), compare to Lemma 3 at the end of this section. In addition, an appropriate reorganization of the operations required for the matrix-vector product (24) leads to a substantial reduction of computational complexity. Moreover, the fast evaluation also induces an automatic compression of the history of the data.

3.3 Multilevel hierarchical basis

In order to obtain algorithms for the matrix-vector multiplication (3) of quasi-optimal complexity, we require the following second fundamental ingredient. Based on the multilevel hierarchy $I^{(n;\ell)}$ of time intervals and the translation invariance of the basis functions $\varphi_i^n =: \varphi_i^{(n;1)}$, we define a multilevel basis

$$\varphi_i^{(n;\ell)}(t) = \begin{cases} \sum_{j=0}^q A_{i,j}^{(1)} \varphi_j^{(2n-1;\ell-1)}(t), & t \in I^{(2n-1;\ell-1)}, \\ \sum_{j=0}^q A_{i,j}^{(2)} \varphi_j^{(2n;\ell-1)}(t), & t \in I^{(2n;\ell-1)}, \end{cases} \tag{28}$$

for the spaces $\mathcal{P}_q\left(I^{(n;\ell)}\right)$, $\ell > 1$ appearing in the farfield blocks of the approximation $k_h \in \mathcal{P}_{q,q}(\mathcal{AT}_h)$. Let us note that by translation invariance, the coefficients $A_{i,j}^{(1)}$ and $A_{i,j}^{(2)}$ are independent of n and ℓ.

For each of the elements of the adaptive partition $\mathcal{AT}_h$, we expand the kernel function as

$$k_h(s,t) = \sum_{i=0}^{q}\sum_{j=0}^{q} k_{i,j}^{(m,n;\ell)} \varphi_i^{(m;\ell)}(s)\varphi_j^{(n;\ell)}(t), \qquad (s,t) \in I^{(m;\ell)} \times I^{(n;\ell)}. \quad (29)$$

For the computation of the farfield contributions in (18), we can further split the integration domain into

$$\left[0, t^{m-2}\right] = \bigcup_{\ell=1}^{L(m)} \bigcup_{n=1}^{B(m;\ell)} I^{(P(m,n;\ell);\ell)} \quad (30)$$

with $L(m) = \lceil \log_2(m) \rceil - 1$, $B(m;\ell) = \mathrm{bin}(m)_\ell + 1$, and $P(m,n;\ell) = C(m;\ell) - n - 1$. Here $\mathrm{bin}(m)_\ell$ denotes the ℓth digit from behind of the binary representation of m obtained by MATLAB's `dec2bin` function. This partitioning of the integration domain exactly corresponds to the cells of a corresponding row in the adaptive mesh $\mathcal{AT}_h$; see Fig. 4 for an illustration. More precisely, $L(m) \in \mathbb{N}$ describes the number of different coarsening levels involved in the mth row, $B(m;\ell) \in \{1,2\}$ corresponds to the number of intervals on each level, and $P(m,n;\ell)$ yields the indices of these intervals on level ℓ. By inserting the splitting of the integration domain in (30) into (18), we obtain

$$y_h\left(t_j^m\right) = \sum_{\ell=1}^{L(m)} \sum_{n=1}^{B(m;\ell)} \int_{I^{(P(m,n;\ell);\ell)}} k_h\left(t_j^m, s\right) f_h(s)\,\mathrm{d}s + \int_{t^{m-2}}^{t_j^m} k_h\left(t_j^m, s\right) f_h(s)\,\mathrm{d}s.$$

We can now further insert the expansions (29) for the kernel k_h into the farfield integrals over $I^{(P(m,n;\ell);\ell)}$ to see that

$$\int_{I^{(P(m,n;\ell);\ell)}} k_h(t_j^m, s) f_h(s)\,\mathrm{d}s$$
$$= \sum_{i=0}^{q} \varphi_i^{(C(n;\ell);\ell)}(t_j^m) \sum_{k=0}^{q} k_{i,k}^{(C(n;\ell),P(m,n;\ell);\ell)} g_k^{(P(m,n;\ell);\ell)},$$

with auxiliary values

$$g_k^{(P(m,n;\ell);\ell)} = \int_{I^{(P(m,n;\ell);\ell)}} \varphi_k^{(P(m,n;\ell);\ell)}(s) f_h(s)\,\mathrm{d}s.$$

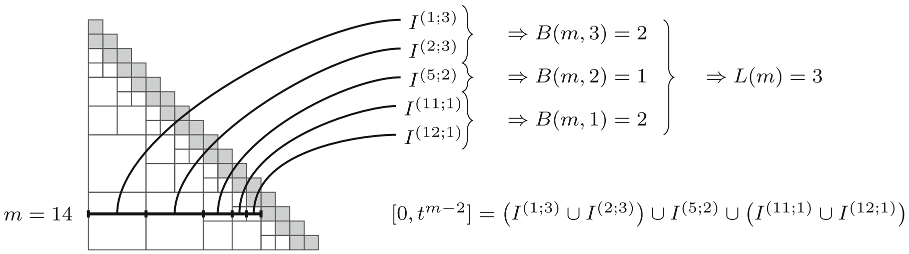

Fig. 4 Illustration of the meaning of the quantities $L(m)$, $B(m,\ell)$, and $P(m,n;\ell)$ arising in (30) for $m = 14$ time steps

By the recursive definition of $\varphi^{(n;\ell)}$, the latter expression can be rewritten as

$$g_k^{(i;\ell)} = \int_{I^{(i;\ell)}} \varphi_k^{(i;\ell)}(s) f_h(s)\,\mathrm{d}s = \sum_{j=0}^{q} \left(A_{i,j}^{(1)} g_k^{(2i-1;\ell-1)} + A_{i,j}^{(2)} g_k^{(2i;\ell-1)} \right)$$

for $\ell > 1$ complemented with $g_k^{(i;1)} = g_k^i = \sum_{r=0}^{p} Q_{k,r} f_r^i$ as defined on the uniform grid in Section 2.2. Evaluation of the recursion (28) at time t_j^m further yields

$$\varphi_i^{(C(n;\ell);\ell)}\left(t_j^m\right) = \sum_{k=0}^{q} A_{i,k}^{(B(m,\ell-1))} \varphi_k^{(C(n,\ell-1);\ell-1)}\left(t_j^m\right),$$

such that we may define intermediate values

$$u_j^{(C(n;\ell);\ell)} = \sum_{i=0}^{q} A_{i,j}^{(B(m;\ell))} u_i^{(C(m,\ell+1);\ell+1)} + \sum_{n=1}^{B(m;\ell)} \sum_{k=0}^{q} k_{i,k}^{(C(n;\ell),P(m,n;\ell);\ell)} g_k^{(P(m,n;\ell);\ell)}.$$

The result of the integral (18) then is finally obtained by

$$y_j^m = y_h(t_j^m) = \sum_{k=0}^{q} P_{j,k} u_k^{(m;1)} + z^m$$

with nearfield contributions z^m and projections $P_{j,k}$ as given in (20) and (22). The above derivations can be summarized as in Algorithm 2.

Remark 5 The MATLAB function `bitxor`(a, b) returns the integer generated by a bit-wise xor comparison of the binary representation of a and b in $\mathcal{O}(1)$ complexity. This allows to determine $L_{\text{coarse}} = \arg\max_k\{B(m;k) \neq B(m-1;k)\}$ in $\mathcal{O}(1)$ complexity in each step and is required for achieving a theoretical runtime of $\mathcal{O}(N)$. In actual implementations, one may just set $L_{\text{coarse}} = L(m)$, without any notable difference in computation times. Further note that only one value $u^{(n;\ell)}$ and two values of $g^{(n;\ell)}$ are required for each level ℓ. Moreover, at most two values of f^n are required at any time step. The required buffers are denoted by $\mathtt{u}^{(\ell)}$, $\mathtt{f}^{(i)}$, and $\mathtt{g}^{(i;\ell)}$, $i = 1, 2$, $\ell = 1, 2, 3, \ldots$. The complexity of the overall algorithm is analyzed in detail in the next section.

Remark 6 Readers familiar with $\mathcal{H}^2$-matrices or the fast multipole method may notice that our algorithm is similar to the corresponding matrix-vector multiplications but with rearranged computations. In each time step, the algorithm checks for changes in the matrix partitioning structure compared to the previous time step. Then, starting from the coarsest level, an upward pass of one level is executed for all coarsened intervals of the farfield by applying the *transfer* or *multipole-to-multipole matrices*. Entities from the coarsened intervals are overwritten. Thereafter, the farfield interactions are computed by applying the *kernel* or *multipole-to-local matrices* corresponding to the changed intervals and directly including the contributions of the next coarser level with a downward pass by appling *transfer* or *local-to-local matrices*. Finally, the nearfield contributions are applied.

Algorithm 2 A fast and oblivious evolutionary algorithm.

```
1:  for $m = 1, \dots, N$ do
2:      $L_{\text{coarse}} = 1 + \lfloor \log_2(\mathtt{bitxor}(m, m-1)) \rfloor$
3:      for $\ell = L_{\text{coarse}}, \dots, 1$ do
4:          if $B(m;\ell) \neq B(m-1;\ell)$ then
5:              $\mathtt{g}^{(2;\ell)} = \mathtt{g}^{(1;\ell)}$
6:              if $\ell > 1$ then
7:                  $\mathtt{g}^{(1;\ell)} = A^{(1)}\mathtt{g}^{(1;\ell-1)} + A^{(2)}\mathtt{g}^{(2;\ell-1)}$
8:              else
9:                  $\mathtt{g}^{(1;\ell)} = Q\mathtt{f}^{(2)}$
10:             end if
11:             Set $(\mathtt{K}_n)_{i,j} = k_{i,j}^{(C(n;\ell),P(m,n;\ell);\ell)}$ for $n \in \{1, B(m;\ell)\}$
12:             $\mathtt{u}^{\ell} = \mathtt{K}_1\mathtt{g}^{(1;\ell)}$
13:             if $B(m;\ell) = 2$ then
14:                 $\mathtt{u}^{\ell} = \mathtt{u}^{\ell} + \mathtt{K}_2\mathtt{g}^{(2;\ell)}$
15:             end if
16:             $\mathtt{u}^{\ell} = \mathtt{u}^{\ell} + \left(A^{(B(m;\ell))}\right)^{\top} \mathtt{u}^{(\ell+1)}$
17:         end if
18:     end for
19:     $\mathtt{f}^{(2)} = \mathtt{f}^{(1)}$
20:     $\mathtt{f}_j^{(1)} = f(t_j^m),\ j = 0, \dots, p$
21:     $z^m = K^{m,m-1}\mathtt{f}^{(2)} + K^{m,m}\mathtt{f}^{(1)}$
22:     $y^m = P\mathtt{u}^{(1)} + z^m$
23: end for
```

3.4 Complexity estimates

In the following, we consider Algorithm 2 for the evaluation of (18) with approximate kernel $k_h \in \mathcal{P}_{q,q}(\mathcal{A}\mathcal{T}_h)$ and data $f_h \in \mathcal{P}_p(\mathcal{T}_h)$, and with $N = 2^L$ denoting the number of time intervals in $\mathcal{T}_h$. The assertions of the following two lemmas are well-known, see, e.g., [5], but their reasoning is simple and illustrative such that we repeat it for the convenience of the reader.

Lemma 2 *Algorithm 2 can be executed in $\mathcal{O}\left(N\left(p^2 + q^2\right)\right)$ operations.*

Proof The algorithm rearranges the operations of a standard $\mathcal{H}^2$-matrix-vector multiplication without adding any significant operations. We therefore simply estimate the complexity of the corresponding $\mathcal{H}^2$-matrix-vector multiplication. Let us first remark that the computation of z^m in line 21 requires $\mathcal{O}(p^2)$ operations in each time step. Second, on a given level ℓ, we have to perform $\mathcal{O}(2^\ell)$ applications of $A^{(1)}$ and $A^{(2)}$ in total for obtaining the $g^{(n;\ell)}$ from the ones on level $\ell - 1$, see line 7. Similarly, $\mathcal{O}(2^\ell)$ applications of $A^{(B(m;\ell))}$ in line 16 are in total required on level ℓ for the computation of the $u^{(n;\ell)}$ and $\mathcal{O}(2^\ell)$ multiplications by $k^{(k,n;\ell)}$ need to be performed in lines 12 and 14. Finally, $\mathcal{O}(N)$ values of $g^n = Qf^n$ and $P\mathrm{u}^{(1)}$ need to be computed in line 9 and line 22. Summing up yields

$$\mathcal{O}\left(Np^2\right) + 3\mathcal{O}\left(q^2\right)\sum_{\ell=1}^{L}\mathcal{O}\left(2^{L-\ell}\right) + 2\mathcal{O}(Npq) = \mathcal{O}\left(Np^2\right) + \mathcal{O}\left(2^L q^2\right) + \mathcal{O}(Npq),$$

and since $N = 2^L$ Young's inequality yields the assertion. □

Lemma 3 *The $\mathcal{H}^2$-matrix representation $\mathtt{K}$ of the adaptive hierarchic approximation $k_h \in \mathcal{P}_{q,q}(\mathcal{AT}_h)$ can be stored in $\mathcal{O}\left(N(p^2+q^2)\right)$ memory. If the kernel is of convolution type* (4)*, then the memory cost reduces to $\mathcal{O}\left(p^2 + \log_2(N)q^2)\right)$.*

Proof The proof for the adaptive approximation is similar to the previous lemma, with the p^2-related term arising from the nearfield and the q^2-related term from the farfield. For a kernel of convolution type, the hierarchical approximation provides a block Toeplitz structure, such that we only have to store $\mathcal{O}(1)$ coefficient matrices per level for the farfield and $\mathcal{O}(1)$ coefficient matrices for the nearfield. □

Let us finally also remark on the additional memory required during execution.

Lemma 4 *The active memory required for storing the data history required for Algorithm 2 is bounded by $\mathcal{O}(q\log_2 N + p)$.*

Proof We require $O(1)$ vectors of length p for the nearfield and at most two vectors $g^{(n;\ell)}$ of length q on $L = \log_2(N)$ levels for the farfield contributions. □

3.5 Summary

In this section, we discussed the adaptive hierarchical data-sparse approximation for the dense system matrix $\mathtt{K}$ in (3) stemming from a uniform polynomial-based discretization of the Volterra integral operators (2). This approximation amounts to an $\mathcal{H}^2$-matrix compression of the system matrix, leading to $\mathcal{O}(N)$ storage complexity for general and $\mathcal{O}(\log(N))$ storage complexity for convolution kernels. Using a multilevel basis representation on the hierarchy of the time intervals, we formulated a fast and oblivious evolutionary algorithm for the numerical evaluation of Volterra integrals (2). The overall complexity for computing the matrix-vector product in (3) is $\mathcal{O}(N)$ and only $\mathcal{O}(\log(N))$ memory is required to store the compressed history of the data. The algorithm is executed in an oblivious and evolutionary manner and can therefore be generalized immediately to integro-differential equations of the form (1). Moreover, knowledge of the number of time steps N is not required prior to execution.

4 Approximation of convolution operators

While our algorithm for the fast evaluation of Volterra integral operators is based on a time domain approximation, in many interesting applications, see [29] for examples and references, the kernel function in (5) is of convolution type

$$k(t,s) = k(t-s) \tag{31}$$

and only accessible indirectly via its Laplace transform, i.e., the transfer function

$$\hat{k}(s) := (\mathcal{L}k)(s) := \int_0^\infty e^{-st} k(t)\,\mathrm{d}t, \qquad s \in \mathbb{C}.$$

Let us note that, at least formally, the evaluation of the kernel function in the time domain can be achieved by the inverse Laplace transform

$$k(t) = (\mathcal{L}^{-1}\hat{k})(t) = \frac{1}{2\pi i}\int_\Gamma e^{t\lambda}\hat{k}(\lambda)\,\mathrm{d}\lambda, \qquad t > 0, \tag{32}$$

where Γ is an appropriate contour connecting $-i\infty$ with $i\infty$; see [3] for details. To ensure the existence of the inverse Laplace transform, we require that

$$\hat{k}(\lambda) \text{ is analytic in a sector } |\arg(\lambda - c)| < \varphi,\ \frac{\pi}{2} < \varphi < \pi, \tag{33}$$

$$\text{and } |\hat{k}(\lambda)| \le M|\lambda|^{-\mu} \text{ for some fixed } M, \mu > 0, \tag{34}$$

and tacitly assume that the contour Γ lies within the domain of analyticity of the function $\hat{k}$. In this section, we show that with some minor modifications, Algorithm 2 and our analysis are applicable also in this setting and we compare our method with the *fast and oblivious convolution quadrature* of [31, 35].

4.1 Approximation and numerical realization

As a first step, we show that the convolution kernel k given implicitly by (32) indeed satisfies the assumption (27) on asymptotic smoothness. Thus, an accurate adaptive hierarchical approximation as discussed in Section 3 is feasible.

Lemma 5 *Assume that $\hat{k}$ satisfies (33) and (34). Then, k as defined in (32) is asymptotically smooth, i.e., it satisfies (27) with $c_2 = \sin(\varphi - \pi/2)$.*

Proof It is sufficient to consider the case $c = 0$ in (33) and $\mu = 1$ in (34). Otherwise, we simply transform $\hat{k}(\lambda + c) = \mathcal{L}(e^{-ct}k(t))(\lambda)$ and $k(t) = k_*^{(\mu-1)}(t)$ with $\hat{k}_*(\lambda) := \mathcal{L}(k_*)(\lambda) = |\lambda|^{\mu-1}\hat{k}(\lambda)$ for $\mu \ne 1$. From [3, Theorem 2.6.1], also see [36], we deduce that k has a holomorphic extension into the sector $|\arg(\lambda)| < \varphi - \pi/2$ with φ as in (33). Thus, the radius of convergence of the Taylor series of k around $t_0 \in (0,\infty)$ is given by $c_2 t_0$, with $c_2 = \sin(\varphi - \pi/2)$ independent of t_0. This implies

$$\left|\partial_t^\alpha k(t)\right| \le c_1 \frac{\alpha!}{c_2^\alpha t^\alpha}$$

for some constant $c_1 > 0$. Condition (27) then follows by the chain rule. □

For the construction of the adaptive approximation k_h, we can now proceed in complete analogy to (11), i.e., we split the convolution integral

$$y_h(t^n) = \int_0^{t^{n-2}} k_h(t^n, s) f_h(s)\,\mathrm{d}s + \int_{t^{n-2}}^{t^n} k_h(t^n, s) f_h(s)\,\mathrm{d}s \tag{35}$$

into a farfield and a nearfield contribution. The latter can be computed stably and efficiently with Lubich's convolution quadrature; see [27, 28] for details.

For the farfield contributions, we utilize the adaptive hierarchical approximation discussed in the previous section. If direct access to the kernel $k(t, s) = k(t - s)$ is available, Algorithm 2 can be applied directly. If, on the other hand, only the transfer function $\hat{k}(s)$ is accessible, the values of $k_h(t, s) = k(t - s)$ can be computed by fast numerical Laplace inversion; see [11, 26, 37]. Here we follow the approach of [26, 35] which is based on hyperbolic contours of the form

$$\gamma(\theta) = \mu(1 - \sin(\alpha + i\theta)) + \sigma, \qquad \theta \in \mathbb{R}, \tag{36}$$

with $0 < \mu$, $0 < \alpha < \pi/2 - \varphi$, and $\sigma \in \mathbb{R}$, such that the contour remains in the sector of analyticity (33) of $\hat{k}$. The discretization of the contour integral (32) by the trapezoidal rule with uniform step with τ yields

$$k(t) \approx \sum_{r=-R}^{R} \frac{i\tau}{2\pi} e^{\gamma(\theta_r)t} \gamma'(\theta_r)\hat{k}(\gamma(\theta_r)), \tag{37}$$

with $\theta_r = \tau r$. Given we are interested in $k(t)$ for $t \in [t_{\min}, t_{\max}]$ and have fixed values for α and σ, suitable parameters τ and μ are given by

$$\tau = a_\rho(\rho_{\text{opt}}), \quad \mu = \frac{2\pi\alpha R(1 - \rho_{\text{opt}})}{t_{\max} a_\rho(\rho_{\text{opt}})}, \quad \rho_{\text{opt}} = \underset{\rho\in(0,1)}{\arg\min} \left(\varepsilon\varepsilon_R(\rho)^{\rho-1} + \varepsilon_R(\rho)^\rho\right),$$

where ε is the machine precision and

$$a_\rho(\rho) = \operatorname{acosh}\left(\frac{t_{\max}/t_{\min}}{(1 - \rho)\sin(\alpha)}\right), \quad \varepsilon_R(\rho) = \exp\left(-\frac{2\pi\alpha R}{a_\rho(\rho)}\right),$$

see [26, 35]. In our examples in Section 5, we chose $\alpha = 3/16\pi$, $\sigma = 0$. For error bounds concerning this approach, we refer to [26, 35].

4.2 Comparison with fast and oblivious convolution quadrature

Similar to Algorithm 2, the *fast and oblivious convolution quadrature* (FOCQ) method of [31, 35] is also based on a splitting (35) of the convolution integral into nearfield and farfield contributions, and the former can again be computed stably and efficiently with Lubich's convolution quadrature [27, 28].

A different adaptive hierarchic approximation based on L-shaped cells is now used for the approximation in the farfield; see Fig. 5. The farfield part of the integration domain for computing the entry $y_h(t^m)$ is then partitioned into

$$\left[0, t^{m-2}\right] = \bigcup_{n=1}^{m-2} I^n = \bigcup_{\ell=1}^{L(m)} \bigcup_{n=1}^{B(m;\ell)} I^{(P(m,n;\ell);\ell)} = \bigcup_{\ell=1}^{L(m)} I^{\ell}_{\text{FOCQ},m}.$$

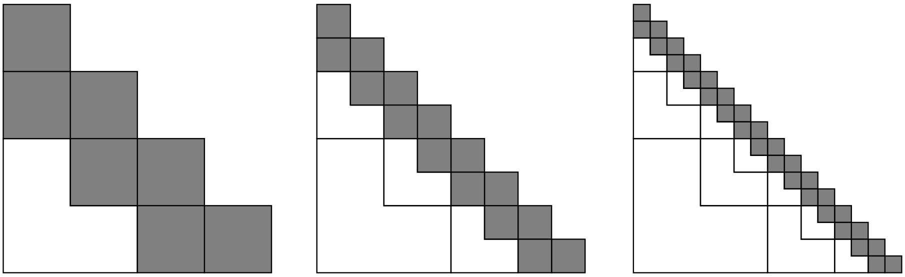

Fig. 5 Hierarchical partitions of fast and oblivious convolution quadrature [35]

Choosing an appropriate contour Γ_ℓ, see (36), and corresponding quadrature points $\theta_r^{(\ell)}$ for each farfield cell and using (37) yields an approximation

$$\begin{aligned}&\int_{I^\ell_{\mathrm{FOCQ},m}} k\left(t^m, s\right) f(s)\,\mathrm{d}s \qquad (38)\\ &\approx \frac{i\tau}{2\pi}\sum_{r=-R}^{R} \hat{k}\left(\gamma\left(\theta_r^{(\ell)}\right)\right)\gamma'\left(\theta_r^{(\ell)}\right) e^{\gamma\left(\theta_r^{(\ell)}\right)\left(t^m-b^{(\ell)}\right)} \underbrace{\int_{I^\ell_{\mathrm{FOCQ},n}} e^{\gamma\left(\theta_r^{(\ell)}\right)\left(b^{(\ell)}-s\right)} f(s)\,\mathrm{d}s}_{=z\left(c^{(\ell)};b^{(\ell)},\gamma\left(\theta_r^{(\ell)}\right)\right)},\end{aligned}$$

with $b^{(\ell)} = \min I^\ell_{\mathrm{FOCQ},m}$ and $c^{(\ell)} = \max I^\ell_{\mathrm{FOCQ},m}$. The values $z\left(c^{(\ell)}; b^{(\ell)}, \gamma\left(\theta_r^{(\ell)}\right)\right)$ can be computed by numerically solving the ordinary differential equation

$$\frac{d}{dt}z(t;s,\gamma) = \gamma z(t;s,\gamma) + f(t), \qquad z(s;s,\gamma) = 0 \qquad (39)$$

with appropriate values $s = b^{(\ell)}$ and $\gamma = \gamma\left(\theta_r^{(\ell)}\right)$. Thus, the fast and oblivious convolution quadrature provides an approximation of the convolution matrix by solving an auxiliary set of $(2R+1)L$ differential equations. In order to obtain an oblivious algorithm, it is crucial that the solution of each differential equation is updated in each time step, i.e., the compressed convolution matrix must be evaluated from *left to right*; see [31, 35] for details.

The connection to our approach is revealed upon noticing that the compression approach underlying the fast and oblivious convolution quadrature actually implements a low-rank approximation in each of the farfields L-shaped blocks, i.e.,

$$\begin{aligned}k(t,s) &\approx \sum_{r=-R}^{R}\left(\frac{i\tau}{2\pi} e^{\gamma\left(\theta_r^{(\ell)}\right)\left(t-b^{(\ell)}\right)}\hat{k}\left(\gamma\left(\theta_r^{(\ell)}\right)\right)\gamma'\left(\theta_r^{(\ell)}\right)\right) e^{\gamma\left(\theta_r^{(\ell)}\right)\left(b^{(\ell)}-s\right)}\\ &= \sum_{r=-R}^{R} U\left(t,\theta_r^{(\ell)}\right) V\left(s,\theta_r^{(\ell)}\right).\end{aligned}$$

The corresponding farfield approximation (38) thus effectively reads

$$\begin{aligned}\int_{I^{\ell}_{\mathrm{FOCQ},m}} k\left(t^{n}, s\right) f(s)\,\mathrm{d}s &\approx \sum_{r=-R}^{R} U\left(t, \theta_r^{(\ell)}\right) \int_{I^{\ell}_{\mathrm{FOCQ},m}} V\left(s, \theta_r^{(\ell)}\right) f(s)\,\mathrm{d}s \\ &= \sum_{r=-R}^{R} U\left(t, \theta_r^{(\ell)}\right) z\left(c^{(\ell)}, b^{(\ell)}, \theta_r^{(\ell)}\right),\end{aligned}$$

which can be understood as a low-rank matrix-vector product realized implicitly by the numerical solution of a differential equation. Since the partitioning depicted in Fig. 5 can easily be refined to an adaptive partitioning as in Fig. 3, the fast and oblivious convolution quadrature can be interpreted as a particular case of an $\mathcal{H}$-matrix approximation with a particular realization of the $\mathcal{H}$-matrix-vector product. The $\mathcal{O}(\log(N))$ memory cost and $\mathcal{O}(N\log(N))$ complexity of the algorithm can then be immediately deduced from $\mathcal{H}$-matrix literature [5, 18].

As mentioned in the introduction, the algorithm proposed in Section 3.2, with the modifications discussed above, is based on an $\mathcal{H}^2$-matrix approximation and leads to a better complexity of $\mathcal{O}(N)$. It is also clear that the number of quadrature points for the numerical Laplace inversion determines the ranks of the farfield approximations for the $\mathcal{H}$-matrix approximations, which allows for an improved understanding of the approximation error in terms of the approximation order.

5 Numerical examples

In the following, we present a series of numerical examples to illustrate and verify the capabilities of our novel algorithms. The experiments are performed in MATLAB, in which we implemented our new algorithm as well as a version of the fast-and-oblivious convolution quadrature algorithm of [26, 31] for reference. Although our new algorithm performed considerably faster in all of the following examples, we do not present a detailed comparison.

In accordance with the $\mathcal{H}$- and $\mathcal{H}^2$-literature, we require the farfield cells in our implementation to be at least $n_{\min} \times n_{\min} = 16 \times 16$ times larger than the nearfield cells. This simply means that the cells in Fig. 3 represent 16×16 blocks of fine grid cells. Following [7, Chapter 2], we choose Radau-IIA collocation methods of stage $p = 1, 2, 3$, for the discretization of the Volterra integral operators, which is exactly the scheme used for the approximation as outlined in Section 2.2 and the beginning of Section 2.3. This fixes the approximation spaces $\mathcal{P}_p(\mathcal{T}_h)$ for the solution y_h and the data f_h. The error of this discretization scheme is given by

$$e_h =: \max_{t_i \in \mathcal{T}_h} |y(t_i) - y_h(t_i)| \leq Ch^{2p-1} \tag{40}$$

for smooth data $f \in C^{2p-1}([0, T])$ and kernel $k \in C^{2p-1}(\{(t, s)\colon 0 \leq t \leq s \leq T\})$; see [7, Chapter 2] for details. For convolution kernels $k(t-s)$, we utilize Lubich's convolution quadrature [27, 28] in the nearfield and the same strategy as above in the farfield. Again, the Radau-IIA method is used as the underlying integration scheme.

This allows to estimate the approximation error by

$$e_h =: \max_{t_i \in \mathcal{T}_h} |y(t_i) - y_h(t_i)| \leq C \left(h^{2p-1} + h^{p+1+\mu} \right) \tag{41}$$

for transfer functions satisfying (33)–(34) and data $f \in C^{2p-1}([0, T])$; see [30] for details. The parameter μ is related to the singularity of the kernel $k(t-s)$ at $t = s$.

5.1 Variation of constants formula

The first example is dedicated to the solution of the differential equation

$$y'(t) = -2ty(t) + 5\cos(5t), \qquad t \in (0, 10], \tag{42}$$

$$y(0) = 2. \tag{43}$$

By the variation of constants formula, the solution can be expressed as

$$y(t) = 2e^{-t^2} + 5\int_0^t e^{s^2-t^2} \cos(5s)\, \mathrm{d}s. \tag{44}$$

Let us note that the integral kernel $k(t, s) = e^{s^2-t^2}$ satisfies the asymptotic smoothness assumption (27), but is not of convolution type $k(t-s)$. To obtain a reference solution, we solve (42)–(43) numerically with a 3-stage Radau-IIA method and with $N_\infty = 2^{19}$ time steps. For the numerical solution of (44), we then employ Algorithm 2 with polynomial degree $q = 16$ for the kernel approximation and various degrees p for the approximation of the data f and the solution y. The left plot of Fig. 6 illustrates that we indeed reach the theoretical convergence rates predicted by (40) up to a tolerance of around 10^{-10} at which numerical noise begins to dominate. From the right plot, one can immediately deduce the linear complexity of the algorithm.

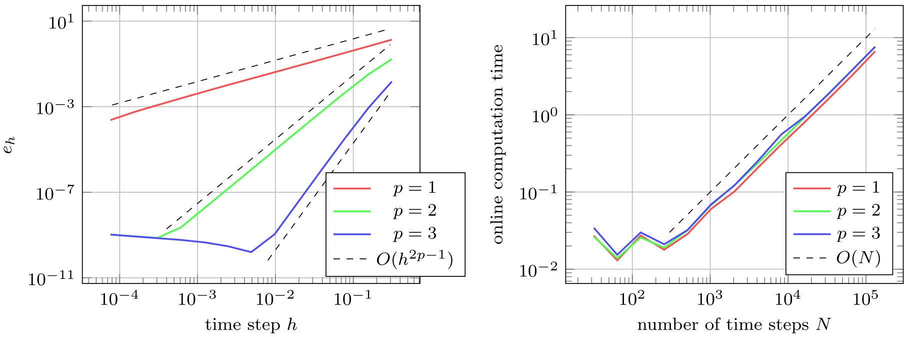

Fig. 6 Approximation errors (left) and computation times (right) for the variation of constants formula example of Section 5.1

5.2 Nonlinear Volterra integral equation

We continue with a second test example taken from [35], in which we consider the nonlinear Volterra integral equation

$$u(t) = -\int_0^t \frac{(u(\tau) - \sin(\tau))^3}{\sqrt{\pi(t-\tau)}}\,\mathrm{d}\tau, \qquad t \in [0, 60].$$

In this example, the convolution kernel $k(t-s) = 1/\sqrt{\pi(t-s)}$ is of convolution type, with Laplace transform given by $\hat{k}(\lambda) = 1/\sqrt{\lambda}$. The evaluation of the integral kernel $k(t-s)$ is realized via numerical inverse Laplace transforms with $R = 15$ quadrature points and the kernel is approximated by piecewise polynomials of degree $q = 8$ in the farfield; see Section 4. Since the data $f(t,u) = (u - \sin(t))^3$ here depend on u, a nonlinear equation must be solved for each time step, for which we employ a Newton method up to a tolerance of 10^{-12}. As a reference solution for computing the errors, we here simply take the numerical solution computed on a finer grid. The results of Fig. 7 again clearly show the predicted convergence rates up to the point where numerical noise begins to dominate, see (41) with $\mu = 1/2$, and the linear complexity of Algorithm 2.

5.3 Fractional diffusion with transparent boundary conditions

As a last example, which is taken from [9, 35], we consider the one-dimensional fractional diffusion equation

$$u(x,t) = u_0(x) + \int_0^t \frac{(t-\tau)^{\alpha-1}}{\Gamma(\alpha)} \Delta_x u(x,\tau)\,\mathrm{d}\tau + g(x,t), \qquad x \in \mathbb{R},\ t \in \mathbb{R}_{>0},$$

with $\alpha = 2/3$, $u(x,\cdot) \to 0$ for $|x| \to \infty$ and $g(x,0) = 0$. For the computations, we restrict the spatial domain to $x \in (-a,a)$, assume that u_0 and g have support in $[-a,a]$, and impose transparent boundary conditions on $x = \pm a$ which read

$$u(x,t) = -\int_0^t \frac{(t-\tau)^{\alpha/2-1}}{\Gamma(\alpha/2)} \partial_{\mathbf{n}} u(x,\tau)\,\mathrm{d}\tau, \qquad x = \pm a;$$

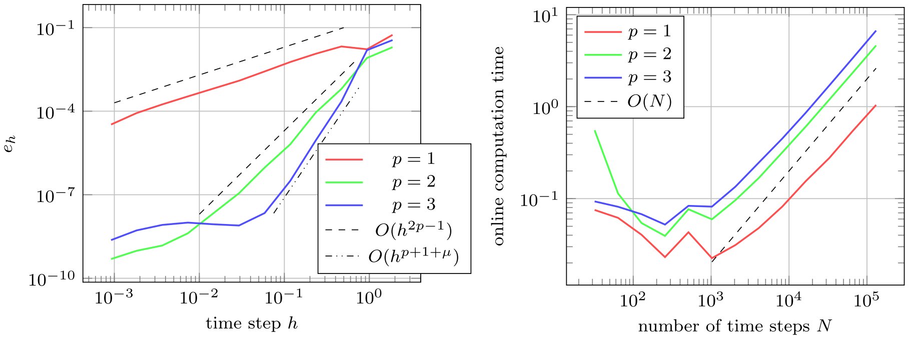

Fig. 7 Approximation errors (left) and computation times (right) for the nonlinear Volterra integral equation example of Section 5.2

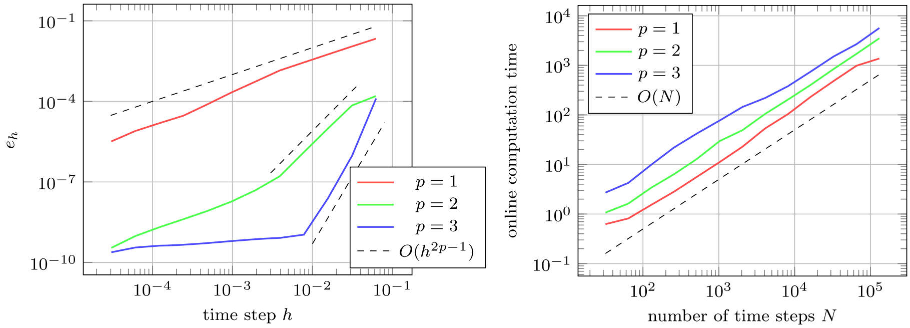

Fig. 8 Convergence plot (left) and computation times (right) for the fractional diffusion problem with transparent boundary conditions

we refer to [20, 35] for further details on the model. The Laplace transform of the convolution kernel $k(t-s) = (t-s)^{\alpha-1}/\Gamma(\alpha)$ is here given by $\hat{k}(\lambda) = 1/\lambda^{\alpha}$.

For the spatial discretization, we employ a finite difference scheme on an equidistant mesh $x_i = i\tau$, $\tau = a/M$, $i = -M, \ldots, M$ and use second-order finite differences within the domain and central differences to approximate the normal derivative on the boundary; see [9, 35]. For the time discretization, we employ the frequency domain version of our algorithm with $R = 30$ quadrature points for the inverse Laplace transform and polynomial degree $q = 16$ for the farfield interpolation. We note that two different convolution quadratures are required, one for the fractional derivative in $(-a, a)$ involving α and one for the fractional derivative of the boundary values, involving $\alpha/2$.

For the space discretization, we consider a fixed mesh with $M = 10^4$ which is fine enough to let the error of the time discretization dominate. As a reference solution, we take the method with order $p = 3$ on a finer mesh. Let us note that due to the lack of temporal smoothness of the solution at $t = 0$, we cannot expect the full order of convergence as predicted by (41); we refer to [9, Section 1] for further discussion. In Fig. 8, we however still observe a very good approximation in the pre-asymptotic phase and a substantial improvement in accuracy until the numerical noise level is reached when using higher approximation order p. As predicted by the theory, the computation times again increase linearly in the number of time steps. In the numerical tests, identical results were obtained for direct evaluation of $k(t, s)$ and evaluation of the kernel via inverse Laplace transforms, which indicates that the main approximation error comes from the adaptive hierarchical approximation.

Acknowledgements The authors would like to thank the first reviewer for an exceptionally careful reading and valuable comments and remarks.

Funding Open Access funding enabled and organized by Projekt DEAL. Support by the German Science Foundation (DFG) via TRR 146 (project C3), TRR 154 (project C04), and SPP 2256 (project Eg-331/2-1) is gratefully acknowledged.

References

1. Alpert, B., Greengard, L., Hagstrom, T.: Rapid evaluation of nonreflecting boundary kernels for time-domain wave propagation. SIAM J. Numer. Anal. **37**(4), 1138–1164 (2000)
2. Amari, S.: Dynamics of pattern formation in lateral-inhibition type neural fields. Biol. Cybern. **27**(2), 77–87 (1977)
3. Arendt, W., Batty, C.J.K., Hieber, M., Neubrander, F.: Vector-Valued Laplace Transforms and Cauchy Problems. Springer Basel, Basel (2011)
4. Baffet, D., Hesthaven, J.S.: A kernel compression scheme for fractional differential equations. SIAM J. Numer. Anal. **55**(2), 496–520 (2017)
5. Börm, S.: Efficient Numerical Methods for Non-Local Operators, volume 14 of EMS Tracts in Mathematics, European Mathematical Society (EMS) Zürich (2010)
6. Brandt, A., Lubrecht, A.A.: Multilevel matrix multiplication and fast solution of integral equations. J. Comput. Phys. **90**(2), 348–370 (1990)
7. Brunner, H.: Collocation Methods for Volterra Integral and Related Functional Differential Equations. Number 15 in Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge (2004)
8. Brunner, H.: Volterra Integral Equations: An Introduction to Theory and Applications. Cambridge University Press, Cambridge (2017)
9. Cuesta, E., Lubich, C., Palencia, C.: Convolution quadrature time discretization of fractional diffusion-wave equations. Math. Comput. **75**(254), 673–697 (2006)
10. Dahmen, W., Prössdorf, S., Schneider, R.: Wavelet approximation methods for pseudodifferential equations II: matrix compression and fast solution. Adv. Comput. Math. **1**(3), 259–335 (1993)
11. Dingfelder, B., Weideman, J.A.C.: An improved Talbot method for numerical Laplace transform inversion. Numer. Algorithm. **68**(1), 167–183 (2015)
12. Dölz, J., Egger, H., Shashkov, V.: A convolution quadrature method for Maxwell's equations in dispersive media (2020)
13. Egger, H., Schmidt, K., Shashkov, V.: Multistep and Runge-Kutta convolution quadrature methods for coupled dynamical systems. J. Comput. Appl. Math. **387**, 112618 (2020)
14. Fong, W., Darve, E.: The black-box fast multipole method. J. Comput. Phys. **228**(23), 8712–8725 (2009)
15. Giebermann, K.: Multilevel approximation of boundary integral operators. Computing **67**(3), 183–207 (2001)
16. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comput. Phys. **73**(2), 325–348 (1987)
17. Hackbusch, W.: A sparse matrix arithmetic based on $\mathcal{H}$-matrices part I: introduction to $\mathcal{H}$-matrices. Computing **62**(2), 89–108 (1999)
18. Hackbusch, W.: Hierarchical Matrices: Algorithms and Analysis. Springer, Heidelberg (2015)
19. Hackbusch, W., Nowak, Z.P.: On the fast matrix multiplication in the boundary element method by panel clustering. Numer. Math. **54**(4), 463–491 (1989)
20. Hagstrom, T.: Radiation boundary conditions for the numerical simulation of waves. Acta Numerica **8**, 47–106 (1999)
21. Hairer, E., Lubich, C.h., Schlichte, M.: Fast numerical solution of nonlinear Volterra convolution equations. SIAM J. Sci. Stat. Comput. **6**(3), 532–541 (1985)
22. Jiang, S., Greengard, L.: Fast evaluation of nonreflecting boundary conditions for the Schrödinger equation in one dimension. Comput. Math. Appl. **47**(6-7), 955–966 (2004)
23. Jiang, S., Greengard, L.: Efficient representation of nonreflecting boundary conditions for the time-dependent Schrödinger equation in two dimensions. Commun. Pure Appl. Math. **61**(2), 261–288 (2008)

24. Kapur, S., Long, D.E., Roychowdhury, J.: Efficient time-domain simulation of frequency-dependent elements. In: Proceedings of International Conference on Computer Aided Design, pp. 569–573. IEEE Comput. Soc. Press, San Jose, CA, USA (1996)
25. Kaye, J., Golež, D.: Low rank compression in the numerical solution of the nonequilibrium Dyson equation. SciPost Phys. **10**(4), 091 (2021)
26. López-Fernández, M., Palencia, C., Schädle, A.: A spectral order method for inverting sectorial laplace transforms. SIAM J. Numer. Anal. **44**(3), 1332–1350 (2006)
27. Lubich, C.H.: Convolution quadrature and discretized operational calculus. I. Numer. Math. **52**(2), 129–145 (1988)
28. Lubich, C.H.: Convolution quadrature discretized operational calculus. II. Numer. Math. **52**(4), 413–425 (1988)
29. Lubich, C.H.: Convolution quadrature revisited. BIT Numer. Math. **44**(3), 503–514 (2004)
30. Lubich, C.H., Ostermann, A.: Runge-Kutta methods for parabolic equations and convolution quadrature. Math. Comput. **60**(201), 105–105 (1993)
31. Lubich, C.H., Schädle, A.: Fast convolution for nonreflecting boundary conditions. SIAM J. Sci. Comput. **24**(1), 161–182 (2002)
32. Metzler, R., Klafter, J.: The random walk's guide to anomalous diffusion: a fractional dynamics approach. Phys. Rep. **339**(1), 1–77 (2000)
33. Rokhlin, V.: Rapid solution of integral equations of classical potential theory. J. Comput. Phys. **60**(2), 187–207 (1985)
34. Sayas, F.-J.: Retarded Potentials and Time Domain Boundary Integral Equations, volume 50 of Springer Series in Computational Mathematics. Springer International Publishing, Cham (2016)
35. Schädle, A., López-Fernández, M., Lubich, C.H.: Fast and oblivious convolution quadrature. SIAM J. Sci. Comput. **28**(2), 421–438 (2006)
36. Sova, M.: The Laplace transform of analytic vector-valued functions (complex conditions). Čas. pro pěstování matematiky **104**(3), 267–280 (1979)
37. Talbot, A.: The accurate numerical inversion of laplace transforms. IMA J. Appl. Math. **23**(1), 97–120 (1979)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

J. Dölz[1] · H. Egger[2] · V. Shashkov[2]

H. Egger
egger@mathematik.tu-darmstadt.de

V. Shashkov
shashkov@mathematik.tu-darmstadt.de

[1] Institute for Numerical Simulation, University of Bonn, Friedrich-Hirzebruch-Allee 7,
53115, Bonn, Germany

[2] Department of Mathematics, Numerical Analysis and Scientific Computing,
TU Darmstadt, Dolivostr. 15, 64293, Darmstadt, Germany