Adv Comput Math (2021) 47:42
https://doi.org/10.1007/s10444-021-09866-7

A fast sparse spectral method for nonlinear integro-differential Volterra equations with general kernels

Timon S. Gutleb[1,2]

Received: 28 January 2020 / Accepted: 8 April 2021 / Published online: 07 May 2021

Abstract
We present a sparse spectral method for nonlinear integro-differential Volterra equations based on the Volterra operator's banded sparsity structure when acting on specific Jacobi polynomial bases. The method is not restricted to convolution-type kernels of the form $K(x, y) = K(x - y)$ but instead works for general kernels at competitive speeds and with exponential convergence. We provide various numerical experiments based on an open-source implementation for problems with and without known analytic solutions and comparisons with other methods.

Keywords Volterra · Integral equations · Nonlinear · Integro-differential · General kernels · Spectral methods · Multivariate orthogonal polynomials

Mathematics Subject Classification (2010) 45D05 · 65N35 · 65R20

1 Introduction

The *Volterra integral operator* is defined by

$$(\mathcal{V}_K u)(x) := \int_0^x K(x, y)u(y)\mathrm{d}y. \tag{1}$$

Integral equations involving Volterra operators occur in diverse applications in various disciplines of engineering [55, 59] and finance and economics [3, 37], as well

Communicated by: Tobin Driscoll

✉ Timon S. Gutleb
t.gutleb18@imperial.ac.uk

[1] Department of Mathematics, Imperial College London, London, UK

[2] Fakultät für Mathematik, University of Vienna, Vienna, Austria

as the natural sciences [9, 20, 28, 30, 31, 35, 44, 58]. Beyond *linear* Volterra integral equations of first and second kinds, which respectively take the forms

$$\mathcal{V}_K u = g \qquad \text{or} \qquad (\lambda I + \mathcal{V}_K)u = g,$$

there exist a vast range of possible generalizations, the most important of which are linear Volterra integro-differential equations (VIDEs), nonlinear Volterra integral equations, and in particular nonlinear VIDEs, the last of which takes the form:

$$\sum_{k=0}^{m} \lambda_k \frac{\mathrm{d}^k}{\mathrm{d}x^k} u(x) = g + \mathcal{V}_K f(u), \tag{2}$$

where $m \in \mathbb{N}$ and $\forall i : \lambda_i \in \mathbb{R}$. Linear Volterra equations thus correspond to the special case $f(u) = u$. For certain very well-behaved cases, one can use variational iteration, Adomian decomposition, or Laplace transform methods [56] to try to obtain analytic solutions for such equations but in general one must use numerical methods to find approximate solutions. We assume in this paper that the equations we intend to solve with our proposed numerical scheme are solvable with unique solutions and omit discussion of ill-posed problems. A number of results on criteria for the existence of solutions are known; see, e.g., [22, 34, 60] and the references therein. If the variable limit of integration is replaced with a constant value, the integral equations are instead known as Fredholm integral equations [25]. Fredholm integrals and integral equations tend to have substantially different properties than their Volterra counterparts, e.g., when it comes to well-posedness [57], and as a result are generally better treated with specialized approaches.

Interest in efficient algorithms with good convergence properties for Volterra integral equations is high, resulting in a variety of competitive methods for linear, nonlinear, and integro-differential Volterra equations. For decades [11], researchers have been proposing various forms of increasingly refined collocation and iteration methods to approach nonlinear VIDEs [1, 2, 13, 15, 47, 53]. More recently, they were also joined by a number of wavelet-based methods [8, 29, 32, 45, 46, 61]. Numerical solvers based on homotopy perturbation methods have also been proposed [21]. A highly efficient spectral solver for the case of convolution kernels $K(x, y) = K(x - y)$ for linear VIDEs using low-rank approximations was recently described by Hale [26]. Gutleb and Olver described a general kernel sparse spectral method for linear Volterra integral equations in [24], which forms the background of the present paper.

In this paper, we present a spectral method which works for linear, nonlinear, integro-differential, and many other types of Volterra equations of first, second, and third kinds while utilizing polynomial spaces in which the Volterra operators have a sparse banded structure. As such, this paper is a direct generalization of results in [24], which derived said banded structure of the Volterra operator in Jacobi polynomial bases and on the basis of [42] developed an efficient Clenshaw algorithm-based approach to numerically generate the operator. This method thus combines the unmatched exponential convergence of spectral methods with highly efficient sparse

linear algebra, a very promising combination which has been successful in recent years [27, 40, 41, 52, 54]. In addition to efficiency via bandedness and exponential convergence rate, the proposed method is furthermore not limited to convolution kernel cases, i.e., kernels of form $K(x, y) = K(x - y)$, a common restriction in competitively fast and accurate approaches [26]. Due to the extensive literature and code libraries available on numerical solutions for linear and nonlinear VIDEs, an exhaustive comparison with other methods is far beyond the scope of a single paper. We will however present comparisons with recent collocation methods, as these are the most competitive and ubiquitous solvers available.

The structure of this paper is as follows: In Sections 1.1 and 1.2, we introduce the necessary framework of uni- and multivariate orthogonal polynomial approximation of functions. Section 1.3 briefly recounts relevant results from [24], detailing the banded structure of the Volterra operator on appropriately chosen triangle domain Jacobi polynomial bases. Section 2 details the extension of the linear Volterra integral equation method to a general linear integro-differential case by augmenting the system with appropriate evaluation operators. Section 3 details the extension to nonlinear Volterra integral equations using an iterative approach. Section 4 combines these two approaches, finally extending the method to general kernel nonlinear VIDEs. Section 5 showcases various numerical experiments based on open-source code [23] for linear VIDEs as well as nonlinear VIEs and VIDEs to verify and test applicability, convergence rate, and competitiveness including comparisons to collocation methods in Chebfun. We close with notes on convergence for the methods proposed in this paper in Section 6 and discuss applicability and potential further research directions in the conclusion.

1.1 Function approximation with univariate orthogonal polynomials

In what follows, we introduce the relevant elements of function approximation in univariate orthogonal polynomial bases, primarily focusing on the Jacobi polynomials, which are required to understand the proposed spectral method for nonlinear VIDEs. The reasons for highlighting the specific chosen properties above others will become clear when we discuss the methods in Sections 2 and 3. For a more general and complete introduction into the theory of function approximation with univariate orthogonal polynomials, see, e.g., [6, 19].

The Jacobi polynomials $P_n^{(\alpha,\beta)}(t)$ are a univariate complete set of polynomials orthogonal with respect to the weight function $(1-t)^\alpha(1+t)^\beta$ on their natural domain $t \in [-1, 1]$, meaning they satisfy

$$\int_{-1}^{1} (1-x)^\alpha (1+t)^\beta P_n^{(\alpha,\beta)}(t) P_m^{(\alpha,\beta)}(t)\, dt = \frac{2^{\alpha+\beta+1}}{2n+\alpha+\beta+1} \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{n!\Gamma(n+\alpha+\beta+1)} \delta_{nm},$$

where $\alpha, \beta \geq -1$. As such, the Legendre polynomials correspond to the special case $\alpha = \beta = 0$ and the ultraspherical or Gegenbauer polynomials correspond to the special case in which $\alpha = \beta$. Spectral methods can make use of the fact that complete

sets of orthogonal polynomials can be used to approximate any sufficiently smooth function $f(t)$ defined on a real interval (a, b) via the expansion

$$f(t) = \sum_{n=0}^{\infty} p_n(x) f_n = \mathbf{p}(t)^{\mathsf{T}} \mathbf{f},$$

where f_n are the unique constant coefficients of $f(t)$ in the given complete polynomial basis $\mathbf{p}(t)$ which is orthogonal on the domain (a, b). These coefficients f_n may be computed efficiently for various sets of orthogonal polynomials using methods and C libraries by Slevinsky [48–50], while evaluation of polynomials is efficiently performed using Clenshaw's algorithm; see, e.g., [19]. While the interval $[-1, 1]$ is the natural choice for the Jacobi polynomials, one can easily shift them to any other real interval required by an application. The method in this paper exclusively makes use of the Jacobi polynomials shifted to the unit interval [0, 1], so we introduce the following shorthand notation:

$$\tilde{\mathbf{P}}(x) = \mathbf{P}(2x - 1), \quad x \in [0, 1].$$

Once expanded in the above way, performing addition and subtraction of functions has an obvious element-wise implementation. Additionally, being orthogonal polynomials, the Jacobi polynomials satisfy a three-term recurrence relationship

$$t P_n^{(\alpha,\beta)}(t) = c_{n-1} P_{n-1}^{(\alpha,\beta)}(t) + a_n P_n^{(\alpha,\beta)}(t) + b_n P_{n+1}^{(\alpha,\beta)}(t), \quad n \geq 1,$$

which allows for efficient computation of function multiplication in this framework via a tridiagonal so-called Jacobi operator:

$$tf(t) = \mathbf{P}(t)^{\mathsf{T}} \mathrm{J}^{\mathsf{T}} \mathbf{f},$$
$$\mathrm{J} = \begin{pmatrix} a_0 & b_0 & & \\ c_0 & a_1 & b_1 & \\ & c_1 & a_2 & \ddots \\ & & \ddots & \ddots \end{pmatrix}.$$

The exact elements of the Jacobi operator depend on the Jacobi parameters (α, β) of the chosen basis; see, e.g., [38, 18.9(i)] for explicit values of a_i, b_i, and c_i. As the sparsity of the operators in this paper relies heavily on correctly moving between bases with different Jacobi parameters (α, β), we will index coefficient vectors with the Jacobi parameters of the basis of expansion, i.e., by writing $\mathbf{f}_{(\alpha,\beta)}$, where it might otherwise be ambiguous. The above properties allow for the development of software packages capable of performing arithmetic on functions using highly efficient sparse linear algebra, where functions are replaced by coefficient vectors. One such package is ApproxFun.jl [39] writen in the Julia programming language [7], which is used as the background environment of the implementations presented in this paper. Other available software packages include among others the Dedalus project [14], Chebfun [5, 16, 43].

Beyond arithmetic with functions, there are a number of other useful operators in the sparse spectral method toolbox. We may, for example, freely shift from a

basis with Jacobi parameters (α, β) to one with higher parameters $(\alpha + n, \beta + m)$ via

$$\tilde{\mathbf{P}}^{(\alpha+n,\beta+m)}(x)^{\mathsf{T}} \mathrm{S}_{(\alpha,\beta)}^{(\alpha+n,\beta+m)} \mathbf{f}_{(\alpha,\beta)} = \tilde{\mathbf{P}}^{(\alpha+n,\beta+m)}(x)^{\mathsf{T}} \mathbf{f}_{(\alpha+n,\beta+m)},$$

where $n, m \in \mathbb{N}$ using a sequence of upper bidiagonal raising operators [38, 18.9.5]

$$\mathrm{S}_{(\alpha,\beta)}^{(\alpha+n,\beta)} = \mathrm{S}_{(\alpha+n-1,\beta)}^{(\alpha+n,\beta)} \cdots \mathrm{S}_{(\alpha+1,\beta)}^{(\alpha+2,\beta)} \mathrm{S}_{(\alpha,\beta)}^{(\alpha+1,\beta)}$$

and an analogous sequence of operators for the second Jacobi parameter. Lowering operators are also available, although this is in general only possible in a sparse (lower bidiagonal) way with added weights [38, 18.9.6]:

$$\begin{aligned} x f(x) &= \tilde{\mathbf{P}}^{(\alpha-1,\beta)}(x)^{\mathsf{T}} \mathrm{L}_{(\alpha,\beta)}^{(\alpha-1,\beta)} \mathbf{f}_{(\alpha,\beta)}, \\ (1-x) f(x) &= \tilde{\mathbf{P}}^{(\alpha,\beta-1)}(x)^{\mathsf{T}} \mathrm{L}_{(\alpha,\beta)}^{(\alpha,\beta-1)} \mathbf{f}_{(\alpha,\beta)}. \end{aligned}$$

We may also mirror functions on their domain in a given Jacobi polynomial basis by using the very useful symmetry property [38, Table 18.6.1]:

$$P_n^{(\alpha,\beta)}(-x) = (-1)^n P_n^{(\beta,\alpha)}(x). \tag{3}$$

In particular, on $[0, 1]$, we can define a diagonal reflection operator via

$$f(1-x) = \sum_n (-1)^n f_{(\alpha,\beta),n} \tilde{P}_n^{(\beta,\alpha)} = \tilde{\mathbf{P}}^{(\alpha,\beta)}(x)^{\mathsf{T}} \mathrm{R} \mathbf{f}_{(\alpha,\beta)}.$$

Differentiation is also a sparse (diagonal) operation if we simultaneously increment the Jacobi parameters [38, 18.9.15], i.e.:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x} f(x) &= \sum_n f_{(\alpha,\beta),n} \frac{\mathrm{d}}{\mathrm{d}x} P_n^{(\alpha,\beta)}(x) && (4) \\ &= \sum_n f_{(\alpha,\beta),n} \frac{1}{2} (n + \alpha + \beta + 1) P_{n-1}^{(\alpha+1,\beta+1)}(x) && (5) \\ &= \tilde{\mathbf{P}}^{(\alpha+1,\beta+1)}(x)^{\mathsf{T}} \mathcal{D}^{(\alpha,\beta)} \mathbf{f}_{(\alpha,\beta)}. && (6) \end{aligned}$$

Importantly, this means that repeated sparse differentiation is not equivalent to a repeat application of the same operator $\mathcal{D}^{(\alpha,\beta)}$. As the derivative operator shifts coefficient vectors to a higher parameter basis, the second derivative operator is actually a combination of two distinct derivative operators acting on different bases and so on for higher derivatives. We thus denote the n-th derivative operator acting on a coefficient vector in $\tilde{\mathbf{P}}^{(\alpha,\beta)}$ basis by $\mathcal{D}_n^{(\alpha,\beta)}$, where

$$\begin{aligned} \frac{\mathrm{d}^n}{\mathrm{d}x^n} f(x) &= \tilde{\mathbf{P}}^{(\alpha+n,\beta+n)}(x)^{\mathsf{T}} \mathcal{D}^{(\alpha+n-1,\beta+n-1)} \cdots \mathcal{D}^{(\alpha,\beta)} \mathbf{f}_{(\alpha,\beta)} \\ &= \tilde{\mathbf{P}}^{(\alpha+n,\beta+n)}(x)^{\mathsf{T}} \mathcal{D}_n^{(\alpha,\beta)} \mathbf{f}_{(\alpha,\beta)}, \end{aligned}$$

instead of the potentially misleading notation $\mathcal{D}^n$ which may evoke false intuitions of commutativity of the operators. The last component of theory we need for our univariate function approximation purposes are endpoint evaluation operators which will be used to enforce boundary conditions in integro-differential equations. From the viewpoint described above, functions are coefficient vectors and multiplications,

derivatives, and basis changes are operators on coefficient vectors (matrices in finite-dimensional approximation space). Functionals, e.g., evaluation operators $\mathcal{E}$ at an endpoint, must act on coefficient vectors to return a scalar value and are thus represented by row vectors. In particular, for the Jacobi polynomials, we can make use of the known property [38, Table 18.6.1]:

$$f(1) = \mathbf{P}^{(\alpha,\beta)}(x)^{\mathsf{T}} \mathcal{E}_1 \mathbf{f}_{(\alpha,\beta)} = \sum_n f_{(\alpha,\beta),n} P_n^{(\alpha,\beta)}(1) = \sum_n f_{(\alpha,\beta),n} \frac{(\alpha+1)_n}{n!},$$

where $(\cdot)_n$ denotes the Pochhammer symbol or rising factorial [38, 5.2(iii)]. Via the symmetry property in Eq. 3, we obtain a similar evaluation operator for the other endpoint of our chosen interval domain.

1.2 Function approximation with multivariate orth. polynomials

This section introduces required elements of function approximation in multivariate orthogonal polynomial bases, focusing on the Jacobi polynomials on the triangle domain

$$T^2 = \{(x, y) : 0 \le x \le 1, 0 \le y \le 1 - x\},$$

which was discussed in detail in [42]. Multivariate polynomials are not yet widely used in numerical methods despite their vast potential. For a more general and complete introduction to theoretical aspects of multivariate orthogonal polynomials, see [17].

Function approximation with multivariate orthogonal polynomials works in direct analogy to the univariate case. For a given multivariate function $f(x, y)$ defined on some suitable 2-dimensional domain and given a set of complete multivariate orthogonal polynomials on said domain, we may expand the function via

$$f(x, y) = \sum_{n=0}^{\infty} \sum_{k=0}^{n} f_{nk} p_{nk}(x, y) = \mathbf{p}(x, y)^{\mathsf{T}} \mathbf{f}.$$

A generalization to n-dimensional cases is straightforward, cf. [17]. On the triangle T^2, a sensible choice of polynomial basis is found in the triangle Jacobi polynomials, also known as Proriol polynomials, which are defined via reference to the univariate Jacobi polynomials [17, Proposition 2.4.1]:

$$P_{k,n}^{(\alpha,\beta,\gamma)}(x, y) = (1 - x)^k \tilde{P}_{n-k}^{(2k+\beta+\gamma+1,\alpha)}(x) \tilde{P}_k^{(\gamma,\beta)}\left(\frac{y}{1 - x}\right).$$

Expansion coefficients for functions on the triangle Jacobi polynomial basis, such as required for the kernel $K(x, y)$ discussed in the next section, may be computed efficiently using C libraries by Slevinsky [48–50]. As in the univariate case, we can define a variety of operators acting on coefficient space such as multiplication operators based on Jacobi operators, derivative operators, and basis change operators [42]. The novelty is that for 2-dimensional spaces such as the triangle we need to distinguish between the x and y variables and thus have two different Jacobi operators: J_x for the x variable and J_y for the y variable, which are now block tri-diagonal operators instead of being tridiagonal; see [42] for details.

1.3 Banded sparsity of the linear Volterra operator in Jacobi bases

It was shown in [24] that the Volterra integral operator is sparse with banded structure on appropriate Jacobi polynomial spaces. Based on this, a sparse spectral method with exponential convergence for linear Volterra equations with general kernels was motivated and analyzed. The results are based on interpreting the Volterra operator as acting on multivariate Jacobi bases on a triangle domain. The idea behind the linear method follows the schemes in Algorithms 1 and 2. In this section, we briefly review these methods to the degree necessary to follow the integro-differential and nonlinear extension in this paper. For the full discussion of the linear case, we refer to [24].

The move to the triangle domain may initially be motivated by noting that the Volterra integral operator $\int_0^{l(x)} K(x, y)u(y)\mathrm{d}y$ acting on u may be considered for other upper bounds $l(x)$ than x, in particular $l(x) = 1 - x$. The Proriol polynomials with parameters $(0, 0, 0)$, being orthogonal on the triangle domain, behave well with respect to this integration:

$$\begin{aligned}
\int_0^{1-x} f(x, y)\mathrm{d}y &= \int_0^{1-x} \sum_{n=0}^{\infty}\sum_{k=0}^{n} p_{n,k}(x, y)\, f_{n,k}\mathrm{d}y \\
&= \sum_{n=0}^{\infty}\sum_{k=0}^{n} f_{n,k}(1-x)^k \tilde{P}_{n-k}^{(2k+1,0)}(x) \int_0^{1-x} \tilde{P}_k^{(0,0)}\left(\frac{y}{1-x}\right)\mathrm{d}y \\
&= \sum_{n=0}^{\infty}\sum_{k=0}^{n} f_{n,k}(1-x)^{k+1} \tilde{P}_{n-k}^{(2k+1,0)}(x) \int_0^{1} \tilde{P}_k^{(0,0)}(s)\,\mathrm{d}s \\
&= \sum_{n=0}^{\infty} f_{n,0}(1-x)\tilde{P}_n^{(1,0)}(x)
\end{aligned}$$

Labeling the $(1-x)$ as a weight term and referring to what remains as operator Q_y, in reference of it being an integration with respect to y, aligns our notation with that in [24]. Using reflection operators, this can be adapted for the more standard $l(x) = x$ case (see [24]), which means that considering the kernel $K(x, y)$ means looking at $K(1 - x, y)$ on this domain. This operator Q_y acts on a function expanded in the Proriol polynomials with parameters $(0, 0, 0)$ on T^2 and as seen above has the form

$$\mathrm{Q}_y = \begin{pmatrix} \overline{|1|} & & & \\ & \overline{|1\ 0|} & & \\ & & \overline{|1\ 0\ 0|} & \\ & & & \overline{|\ddots\ \ddots\ \ddots\ \ddots|} \end{pmatrix}.$$

We may account for the as of now omitted weight term $(1 - x)$ by using a direct multiplication with Jacobi operators but for reasons of efficiency and due to the need to reflect when $l(x) = x$ is better performed using a bidiagonal lowering operator

followed by a diagonal reflection and finally a bidiagonal raising operator. The discussion so far explains the form of the operator for linear Volterra integral equations of second kind in Algorithm 2 being $\left(\mathbb{1} - \mathrm{S}_{(0,0)}^{(1,0)}\mathrm{RL}_{(1,0)}^{(0,0)}\mathrm{V}_K\right)$. Equations of first kind are somewhat more subtle and we thus omit discussion of further details, referring instead to the original linear method derivations and proofs in [24]. We have assumed above that the function may be expanded in the Proriol polynomials but we can make use of additional sparsity structures when instead thinking of $f_{n,k}$ as the extension of univariate function coefficients f_n extended to the triangle domain via the expansion operator

$$\mathbf{P}(x, y)^{\mathsf{T}}\mathbf{f}_{\triangle} = \mathbf{P}(x, y)^{\mathsf{T}}\mathrm{E}_y\mathbf{f}.$$

Choosing the respectively optimal bases $\tilde{\mathbf{P}}^{(1,0)}(x)$ and $\mathbf{P}^{(0,0,0)}(x, y)$ for this purpose results in the extension operator found in [24], which when multiplied with the integration from 0 to $1 - x$ operator above results in the following diagonal operator

$$(\mathrm{Q}_y\mathrm{E}_y)_{n,n} = (\mathrm{D}_y)_{n,n} = \frac{(-1)^{n+1}}{n}.$$

Via certain quasi-commutativity properties of the above-discussed operators and the Jacobi operators on the triangle domain and using diagonal reflection operators appropriately, one may iteratively build the full Volterra integral operator for a general kernel via the efficient operator-valued Clenshaw algorithm for general kernel linear Volterra integral equations introduced in [24]. We will refer to this Volterra operator *without* the weight $(1 - x)$ as V_K in the following sections and assume it is computed using the methods outlined here and detailed in [24]. The weight is accounted for by using appropriate basis shifts or multiplication as detailed in the algorithm steps.

Algorithm 1 Linear Volterra integral equation of first kind [24].

$$\int_0^x K(x, y)u(y)\mathrm{d}y = g(x).$$

1. Expand $q(x) = \frac{g(1-x)}{1-x}$ in $\tilde{\mathbf{P}}^{(1,0)}(x)$.
2. Generate V_K recursively via an operator-valued Clenshaw algorithm for the flipped kernel $K(1 - x, y)$.
3. Solve the linear system $\mathrm{V}_K\mathbf{u} = \mathbf{q}$ for $\mathbf{u}$.
4. The approximate solution is $\tilde{\mathbf{P}}^{(1,0)}(x)^{\mathsf{T}}\mathbf{u}$.

2 Extension of the linear case sparse spectral method to integro-differential equations

Volterra integro-differential equations (VIDEs) are named such because the unknown appears in the equation under the action of both a Volterra integral and a derivative

Algorithm 2 Linear Volterra integral equation of second kind [24].

$$u(x) = g(x) + \int_0^x K(x, y)u(y)\mathrm{d}y.$$

1. Expand $g(x)$ in $\tilde{\mathbf{P}}^{(1,0)}(x)$.
2. Generate V_K recursively via an operator-valued Clenshaw algorithm for the flipped kernel $K(1-x, y)$.
3. Solve the linear system $\left(\mathbb{1} - \mathrm{S}_{(0,0)}^{(1,0)}\mathrm{RL}_{(1,0)}^{(0,0)}\mathrm{V}_K\right)\mathbf{u} = \mathbf{g}$ for $\mathbf{u}$.
4. The approximate solution is $\tilde{\mathbf{P}}^{(1,0)}(x)^\mathsf{T}\mathbf{u}$.

operator. In this section, we will consider linear VIDEs of the following generic form:

$$\sum_{m=0}^{M} \lambda_m \frac{\mathrm{d}^m}{\mathrm{d}x^m} u(x) = g + \mathcal{V}_K u, \tag{7}$$

with constants λ_m and $M \in \mathbb{N}$. Within the context of the present spectral method, the integral operator is the Volterra operator from Section 1.3, which as we saw maps a coefficient vector of a function in the $\tilde{\mathbf{P}}^{(1,0)}(x)$ basis to the solution in the same basis. Consistent basis considerations, specifically the Jacobi parameters of our chosen basis, are crucial when developing a solution algorithm for integro-differential equations. As noted in Section 1.1, there is an abundance of useful structure in the Jacobi polynomials which among other things allows us to take derivatives by shifting the basis parameters as in Eqs. 4–6. Applying a derivative operator is the same as applying a parameter-scaled raising operator and thus incurs a basis change, which needs to be accounted for in the other operators. We choose the integro-differential equation of second order

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2} u(x) = g(x) + \int_0^x (x - y)u(y)\mathrm{d}y.$$

as an example to illustrate this. To consistently obtain a solution from an extension to the above linear method, it does not suffice to simply replace the second-order derivative operator with the appropriate Jacobi polynomial basis derivative operator. Instead, due to the incurred basis shift, an additional conversion or shift operator must be applied to the Volterra operator as well. Starting from the $\tilde{\mathbf{P}}^{(1,0)}(x)$ basis in which we obtain our solution $\mathbf{u}$, the second derivative operator carries us into the basis $\tilde{\mathbf{P}}^{(3,2)}(x)$, meaning that, taking note of the steps in Algorithm 2, the appropriate operator form of the above second-order example equation is

$$\tilde{\mathbf{P}}^{(3,2)\mathsf{T}}\left(\mathcal{D}_2^{(1,0)} - \mathrm{S}_{(1,0)}^{(3,2)}\mathrm{S}_{(0,0)}^{(1,0)}\mathrm{RL}_{(1,0)}^{(0,0)}\mathrm{V}_K\right)\mathbf{u}_{(1,0)} = \tilde{\mathbf{P}}^{(3,2)\mathsf{T}}\mathbf{g}_{(3,2)}.$$

We may collapse the compatible conversion operators down into a single one to obtain the slightly simpler

$$\tilde{\mathbf{P}}^{(3,2)\mathsf{T}}\left(\mathcal{D}_2^{(1,0)} - \mathrm{S}_{(0,0)}^{(3,2)}\mathrm{RL}_{(1,0)}^{(0,0)}\mathrm{V}_K\right)\mathbf{u}_{(1,0)} = \tilde{\mathbf{P}}^{(3,2)\mathsf{T}}\mathbf{g}_{(3,2)}. \tag{8}$$

Note that $g(x)$ must be expanded in the $\tilde{\mathbf{P}}^{(3,2)}(x)$ basis instead of the $\tilde{\mathbf{P}}^{(1,0)}(x)$ basis or converted into said basis using the above-defined basis shift operators. This is for consistency reasons as the operators acting on $\mathbf{u}_{(1,0)}$ shifting the basis from $\tilde{\mathbf{P}}^{(1,0)}(x)$ to $\tilde{\mathbf{P}}^{(3,2)}(x)$ means that the inverse of said operation must act on a function expanded in $\tilde{\mathbf{P}}^{(3,2)}(x)$. This is not an artifact of our choice of the $\tilde{\mathbf{P}}^{(1,0)}(x)$ basis for our solution: While that basis is particularly well-suited for Volterra integral equations as it results in a far more efficient kernel computation [24], the derivative operator in the VIDE will always shift the basis of our solution, so to optimize efficiency $g(x)$ is always initially expanded in the $\tilde{\mathbf{P}}^{(1+M,M)}(x)$ basis where M is the order of the highest appearing derivative operator. Even in the general case with multiple derivative operators of different orders, the basis for the solution always remains $\tilde{\mathbf{P}}^{(1,0)}(x)$ (for efficiency of kernel computations) while the highest order derivative operator determines the basis in which $g(x)$ must be expanded along with the shift operators which respectively act on all lower order operators as well as the Volterra integral operator.

Attempting to invert the operator on the left-hand side of Eq. 8 as-is will yield nonsensical results. This should be unsurprising, as the differential equation it corresponds with does not have a unique solution unless initial conditions are supplied as well. Given a Volterra integro-differential equation with highest appearing derivative operator of order $M \in \mathbb{N}$, we will in general require initial conditions for all lower order derivatives to be given, i.e.:

$$\frac{\mathrm{d}^m}{\mathrm{d}x^m} u(0) = c_m, \quad m = 0...M-1,$$

for given constants c_m. In the example case of Eq. 8, the values $u(0)$ and $u'(0)$ must be given. In spectral methods such as the one discussed in this paper, boundary or initial conditions are enforced by extending the to-be-inverted operator by appropriate evaluation operators. The relevant Jacobi basis evaluation operators, being functionals, are represented in the coefficient vector and operator language as row vectors, as discussed in Section 1.1. For the example in Eq. 8, we thus append the two initial condition evaluations at the top of the operator as follows obtaining the now solvable system:

$$\begin{pmatrix} \mathcal{E}_0 \\ \mathcal{E}_0 \mathcal{D}^{(1,0)} \\ \mathcal{D}_2^{(1,0)} - S_{(1,0)}^{(2,3)} V \end{pmatrix} \mathbf{u}_{(1,0)} = \begin{pmatrix} c_0 \\ c_1 \\ \mathbf{g}_{(3,2)} \end{pmatrix}, \tag{9}$$

with consistently modified right hand side. Similar procedures have previously been used to solve differential equations, cf. [27, 54]. The discussion in this section in combination with the linear Volterra integral method in Algorithms 1 and 2 thus provides a recipe for the solution of general linear Volterra integro-differential equations satisfying a sufficient set of initial conditions. We produce the general case method in Algorithm 3. The resulting operator on the left-hand side has filled-in top rows for each initial condition and thus is no longer fully banded but still retains very well-behaved sparsity structure (semi-banded) leading to fast solutions even for high orders of polynomial approximation.

Algorithm 3 Linear integro-differential Volterra equation of second kind.

$$\sum_{m=0}^{M} \lambda_m \frac{\mathrm{d}^m}{\mathrm{d}x^m} u(x) = g(x) + \int_0^x K(x, y)u(y)\mathrm{d}y \quad , \quad \lambda_m \in \mathbb{R}; m, M \in \mathbb{N}$$
$$\frac{\mathrm{d}^m}{\mathrm{d}x^m} u(0) = c_m \ , \quad m = 0...M-1, c_m \in \mathbb{R}.$$

1. Expand $g(x)$ in $\tilde{\mathbf{P}}^{(1+M,M)}(x)$.
2. Generate V_K recursively via an operator-valued Clenshaw algorithm for the flipped kernel $K(1-x, y)$.
3. Generate the operator $\left(\sum_{m=0}^{M} \lambda_m \mathrm{S}^{(1+M,M)}_{(1+m,m)} \mathcal{D}^{(1,0)}_m - \mathrm{S}^{(1+M,M)}_{(0,0)} \mathrm{RL}^{(0,0)}_{(1,0)} \mathrm{V}_K\right)$.
4. Append evaluation operators $(\mathcal{E}_0, \mathcal{E}_0\mathcal{D}^{(1,0)}, ...)$ to the top row of the operator and corresponding initial conditions $(c_0, c_1, ...)$ to the top of $\mathbf{g}_{(1+M,M)}$.
5. Solve the semi-banded linear system for $\mathbf{u}_{(1,0)}$:
$$\begin{pmatrix} \mathcal{E}_0 \\ \mathcal{E}_0\mathcal{D}^{(1,0)} \\ \vdots \\ \mathcal{E}_0\mathcal{D}^{(M,M-1)} \\ \sum_{m=0}^{M} \lambda_m \mathrm{S}^{(1+M,M)}_{(1+m,m)} \mathcal{D}^{(1,0)}_m - \mathrm{S}^{(1+M,M)}_{(0,0)} \mathrm{RL}^{(0,0)}_{(1,0)} \mathrm{V}_K \end{pmatrix} \mathbf{u}_{(1,0)} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{M-1} \\ \mathbf{g}_{(1+M,M)} \end{pmatrix}$$
6. The approximate solution is $\tilde{\mathbf{P}}^{(1,0)}(x)^{\mathsf{T}} \mathbf{u}_{(1,0)}$.

3 Nonlinear Volterra equations via iterative methods

In this section, we develop an iterative approach for solving nonlinear Volterra integral equations based on the linear case sparse spectral method. Computing solutions to nonlinear Volterra and Fredholm integral equations with iterative methods is not a novel idea in itself (see, e.g., [15]), but typically comes with significant drawbacks, cf. remarks in [4, 18]. The core problem with iterative methods is how rapidly their computational cost scales with the expense of evaluation in each iteration. The slower the rate of convergence and the more expensive the individual evaluation in each step, the less feasible iterative methods become. Conversely, the presented sparse spectral method is very well suited to be used in conjunction with iterative methods as it not only converges exponentially but also keeps evaluation cost comparatively low by making use of operator bandedness in the chosen bases.

We will primarily use a simple Newton iteration algorithm without linesearch on the basis of implementations in NLsolve.jl [36] for the numerical experiments but in principle many other iterative approaches may be used, resulting in further speed-ups in some cases.

The main idea of the extension to the nonlinear case is to notice that given functions K, g, and f the general nonlinear, second kind Volterra equation

$$u = g + \mathcal{V}_K f(u),$$

may be cast into the form of a root-finding problem in function space for the objective function $F(u)$ defined by

$$F(u) := u - \mathcal{V}_K f(u) - g = 0.$$

The initial guess required for iterative approaches is thus made at the level of coefficient vectors, meaning that a guessed column vector representing the solution in the $\tilde{\mathbf{P}}^{(1,0)}(x)$ basis is supplied to the iterative solver. When no convergence automation is used, the supplied length of the guess as well as $\mathbf{g}$ determines the maximum polynomial degree and thus the approximation error. The step-by-step method is stated in Algorithm 4.

Algorithm 4 Nonlinear Volterra integral equation of second kind.

$$u(x) = g(x) + \int_0^x K(x, y) f(y, u(y)) \mathrm{d}y.$$

1. Expand $g(x)$ in $\tilde{\mathbf{P}}^{(1,0)}(x)$.
2. Generate V_K recursively via an operator-valued Clenshaw algorithm for the flipped kernel $K(1 - x, y)$.
3. Generate the operator $\left(\mathbb{1} - \mathrm{S}^{(1,0)}_{(0,0)} \mathrm{RL}^{(0,0)}_{(1,0)} \mathrm{V}_K\right)$.
4. Apply a simultaneous root-search (e.g., Newton method) to components of objective function $F(\mathbf{u}) = \left(\mathbb{1} - \mathrm{S}^{(1,0)}_{(0,0)} \mathrm{RL}^{(0,0)}_{(1,0)} \mathrm{V}_K\right) \mathbf{f}(y, \mathbf{u}) - \mathbf{g}$.
5. The approximate solution is the obtained root $\tilde{\mathbf{P}}^{(1,0)}(x)^{\mathsf{T}}\mathbf{u}$.

4 Nonlinear integro-differential Volterra equations

We can straightforwardly combine considerations in Sections 2 and 3 to obtain a sparse spectral method suitable for solving Volterra equations featuring both derivative operators and Volterra integral operators with nonlinearities. A very (but not exhaustively) general case of such an equation of second kind is

$$\sum_{k=0}^{m} \lambda_k \frac{\mathrm{d}^k}{\mathrm{d}x^k} u(x) = g + \mathcal{V}_K f(u). \tag{10}$$

For brevity, we only address equations of the form in Eq. 10 but the methodology outlined in this paper is applicable for a much broader class of problems. The full step-by-step method is stated in Algorithm 5.

Algorithm 5 Nonlinear integro-differential Volterra equation of second kind.

$$\sum_{m=0}^{M} \lambda_m \frac{\mathrm{d}^m}{\mathrm{d}x^m} u(x) = g(x) + \int_0^x K(x,y) f(y,u(y)) \mathrm{d}y \quad , \quad \lambda_m \in \mathbb{R}; m, M \in \mathbb{N}$$

$$\frac{\mathrm{d}^m}{\mathrm{d}x^m} u(0) = c_m \quad , \quad m = 0 \ldots M-1.$$

1. Expand $g(x)$ in $\tilde{\mathbf{P}}^{(1+M,M)}(x)$.
2. Generate V_K recursively via an operator-valued Clenshaw algorithm for the flipped kernel $K(1-x, y)$.
3. Generate the operator $\left(\sum_{m=0}^{M} \lambda_m \mathrm{S}_{(1+m,m)}^{(1+M,M)} \mathcal{D}_m^{(1,0)} - \mathrm{S}_{(0,0)}^{(1+M,M)} \mathrm{RL}_{(1,0)}^{(0,0)} \mathrm{V}_K\right)$.
4. Append evaluation operators ($\mathcal{E}_0$, $\mathcal{E}_0\mathcal{D}^{(1,0)}$, ...) to the top row of the operator and corresponding initial conditions (c_0, c_1, ...) to the top of $\mathbf{g}_{(1+M,M)}$.
5. Apply a simultaneous root-search (e.g., Newton method) to components of objective function $F(\mathbf{u})$ defined by
$$\begin{pmatrix} \mathcal{E}_0\mathbf{u} - c_0 \\ \mathcal{E}_0\mathcal{D}^{(1,0)}\mathbf{u} - c_1 \\ \vdots \\ \mathcal{E}_0\mathcal{D}^{(M,M-1)}\mathbf{u} - c_{M-1} \\ \left(\sum_{m=0}^{M} \lambda_m \mathrm{S}_{(1+m,m)}^{(1+M,M)} \mathcal{D}_m^{(1,0)} - \mathrm{S}_{(0,0)}^{(1+M,M)} \mathrm{RL}_{(1,0)}^{(0,0)} \mathrm{V}_K\right) \mathbf{f}(y,\mathbf{u}) - \mathbf{g}_{(1+M,M)} \end{pmatrix}.$$
6. The approximate solution is the obtained root $\tilde{\mathbf{P}}^{(1,0)}(x)^{\mathsf{T}}\mathbf{u}$.

5 Numerical experiments

The Julia code implemented for the numerical experiments in this section is available at [23] and includes example files named after the appropriate subsections to help reproduce our figures.

Throughout this section, we measure errors between analytic solutions $u(x)$ and computed approximate solutions $\tilde{\mathbf{P}}^{(1,0)}(x)^{\mathsf{T}}\mathbf{u}_{(1,0)}$ in each point of the domain via the infinity norm of the absolute error

$$\|u(x) - \tilde{\mathbf{P}}^{(1,0)}(x)^{\mathsf{T}}\mathbf{u}_{(1,0)}\|_\infty = \sup_{x\in[0,1]} |u(x) - \tilde{\mathbf{P}}^{(1,0)}(x)^{\mathsf{T}}\mathbf{u}_{(1,0)}|.$$

5.1 Numerical experiments with linear VIDEs

5.1.1 Set 1: Second kind, convolution kernels, one derivative operator

As a proof-of-concept, we first test the above method on three simple convolution kernel cases with analytically known results:

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}u_1(x) = 1 + \int_0^x (x-y)u_1(y)\mathrm{d}y, \tag{11}$$

$$\frac{\mathrm{d}^4}{\mathrm{d}x^4}u_2(x) = -1 + x + \int_0^x (y-x)u_2(y)\mathrm{d}y, \tag{12}$$

$$\frac{\mathrm{d}^3}{\mathrm{d}x^3}u_3(x) = 1 + x + \frac{x^2}{2} - \frac{x^4}{4!} + \int_0^x \frac{(x-y)^2}{2}u_3(y)\mathrm{d}y, \tag{13}$$

with initial conditions given by

$$u_1(0) = 1, \quad u_1'(0) = 0, \tag{14}$$

$$u_2(0) = -1, \quad u_2'(0) = 1 \quad u_2''(0) = 1 \quad u_2'''(0) = -1, \tag{15}$$

$$u_3(0) = 1, \quad u_3'(0) = 2, \quad u_3''(0) = 1. \tag{16}$$

The following analytic solutions derived respectively via variational iteration, Adomian decomposition, and Laplace transform methods are found in [56]:

$$\begin{aligned} u_1(x) &= \cosh(x), \\ u_2(x) &= \sin(x) - \cos(x), \\ u_3(x) &= x + e^x. \end{aligned}$$

We plot the absolute error of the computed solution compared to the analytic solution in semi-logarithmic scale in Fig. 1, showing exponential convergence to the exact solutions. As these are simple problems which may be solved with low-order approximations (and thus small matrix dimensions) even with dense methods, the speed advantage gained from bandedness for such problems is naturally small. We state timings for these problems in Table 1.

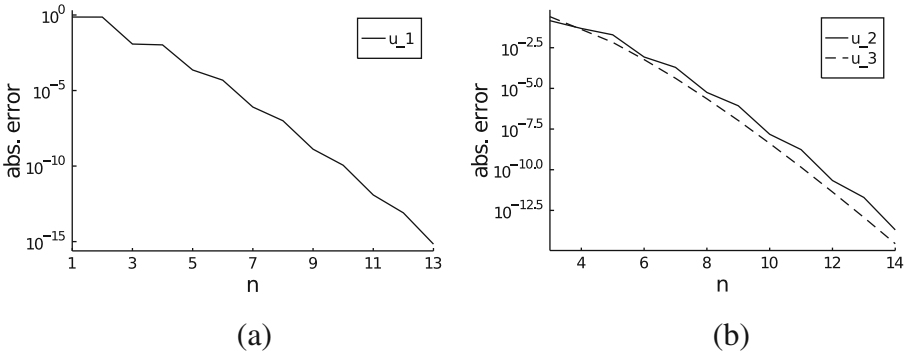

Fig. 1 Absolute error between analytic and computed solutions for $u_1(x)$, $u_2(x)$, and $u_3(x)$ in Eqs. 11–13 for polynomial approximation of order n with initial conditions in Eqs. 14–16

Table 1 Performance of sparse method for Eqs. 11–13 with approximation order 15

Problem	CPU time	approx. order	abs. error
Equation 11	0.04 s	15	7.4e−15
Equation 12	0.07 s	15	6.5e−16
Equation 13	0.06 s	15	2.3e−15

CPU time measured on Intel Core i7-8550U CPU @ 1.80GHz

5.1.2 Set 2: Collocation method for third kind integro-differential equations

A collocation method is used in [47] to solve certain types of third kind integro-differential Volterra equations of form

$$x^{\beta}\frac{\mathrm{d}}{\mathrm{d}x}u(x) = x^{\beta}a(x)u(x) + x^{\beta}g(x) + \int_0^x K(x,y)u(y)\mathrm{d}y.$$

While we do not explicitly treat third kind equations in this paper, the discussion of first and second kind integro-differential equations in Section 2 suggests a natural extension to these cases. Shayanfard, Dastjerdi, and Ghaini [47] discuss two numerical examples and provide a table of error values for differently chosen collocation points. The two numerical experiments with non-convolution kernels are

$$x^{\frac{2}{3}}u_1'(x) = x^{\frac{2}{3}}\left(\frac{10}{3}x^{\frac{7}{3}} - \frac{3}{16}x^{\frac{14}{3}}\right) + \int_0^x yu_1(y)\mathrm{d}y, \tag{17}$$

$$x^{\frac{1}{2}}u_2'(x) = \frac{1}{20}xu_2(x) + \frac{9}{2}x^4 - \frac{1}{20}x^{\frac{11}{2}} - \frac{1}{6}x^6 + \int_0^x y^{\frac{1}{2}}u_2(y)dy, \tag{18}$$

with initial conditions respectively given by

$$u_1(0) = 0, \quad u_2(0) = 0, \tag{19}$$

and known analytic solutions

$$u_1(x) = x^{\frac{10}{3}},$$
$$u_2(x) = x^{\frac{9}{2}}.$$

In Fig. 2, we compare absolute errors of the results obtained with our approach to the errors obtained with their collocation method (as given in Tables 1 and 2 in [47]).

The bandwidth of the operators for such third kind VIDEs is large or even dense as they feature additional multiplications with Jacobi operators with poorly approximated rational powers in them. While our proposed method still performs better in terms of accuracy and convergence as seen in Fig. 2, it is not immediately obvious that a speed advantage can be gained over standard collocation methods. In third kind equations which require a high order of approximation, however, we can nevertheless use an approximately sparse approach as the resulting third kind Volterra operators are banded-dominant and decay exponentially off the main band; see Fig. 3. By generating these operators in a banded form with a set bandwidth, we can obtain much more efficient solutions to third kind integro-differential problems while still

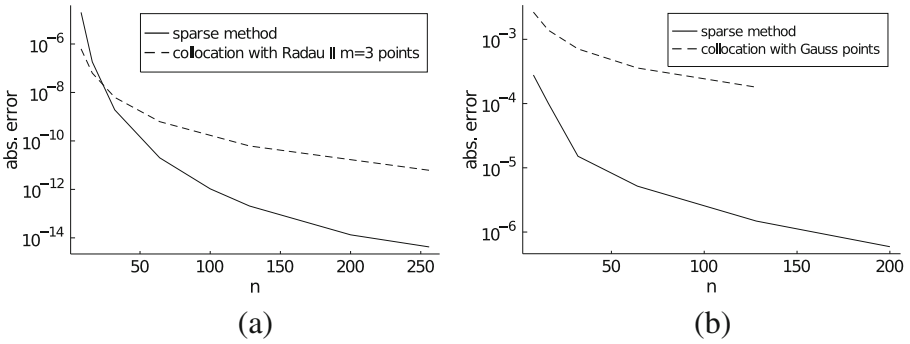

Fig. 2 **a** shows absolute error between analytic and computed solutions for Eq. 17 and **b** for Eq. 18 for approximations of order n. The errors for the collocation method are taken directly from Tables 1 and 2 in [47]. For the errors with our method, the bandwidth was set to increase to convergence; see Table 2 for a bandwidth-based view of errors

retaining good accuracy; see the timing comparisons in Table 2 for the example of $n = 100$.

5.2 Integro-differential equations in Chebfun

The Chebfun package allows state-of-the-art computations using polynomial approximations and collocation methods in MATLAB [5, 16, 43]. An implementation of an automatic collocation method for integral and integro-differential Volterra and Fredholm equations in Chebfun was presented in [15]. In this section, we aim to compare performance of the sparse method compared to the dense collocation method used in Chebfun for problems requiring low and high polynomial orders.

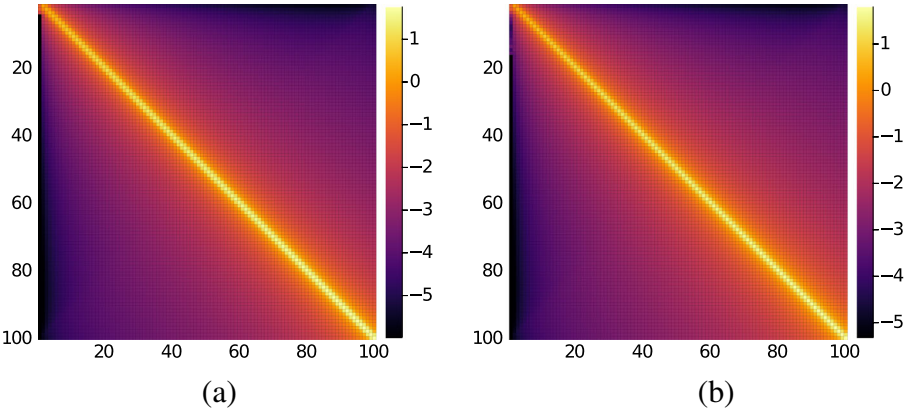

Fig. 3 Dense view of approximately banded form of third kind Volterra integro-diff. operators in Eqs. 17–18, with logarithmic legend indicating the order of magnitude of the elements. Generating these operators with fixed bandwidth yields results in good accuracy and significant speed improvements; see Table 2

Table 2 Performance comparison of our method for Eqs. 17–18

Equation 17, off-diagonal bands	CPU time	approx. order	abs. error
autom. (max)	3.9 s	100	1.1e−12
50	0.17 s	100	1.6e−8
30	0.12 s	100	1.4e−7
10	0.11 s	100	1.5e−5
Equation 18, off-diagonal bands	CPU time	approx. order	abs. error
autom. (max)	4.0 s	100	2.5e−6
30	0.19 s	100	2.6e−6
10	0.16 s	100	3.9e−5

Approximating the operator as banded with stated off-diagonal bands yields significant speed improvements while retaining good accuracy. CPU time measured on Intel Core i7-8550U CPU @ 1.80GHz

5.2.1 Low-order solutions

The example in this section is given in [15] and is a non-convolution kernel linear VIDE which previously appeared in a discussion of higher order collocation methods for VIDEs by Brunner [12]. We seek a solution to

$$u_1'(x) + u_1(x) = 1 + 2x + \int_0^x x(1+2x)e^{y(x-y)}u_1(y)\mathrm{d}y, \tag{20}$$

with initial condition

$$u_1(0) = 1, \tag{21}$$

and known analytic solution

$$u_1(x) = e^{x^2}.$$

In Fig. 4a, we plot the absolute error of the solution obtained via the sparse spectral method with maximal polynomial approximation order n. We present a spy plot of the quasi-banded integro-differential operator generated by our sparse method in Fig. 4b. We find that for orders around $n = 20$, where machine precision accuracy is within reach, the operator for this problem is still dense and thus the proposed method should realistically only match Chebfun's performance. That a speed-up is nevertheless observed (see Table 3) may be explained by language-specific differences between MATLAB and Julia, the automatic convergence search which Chebfun performs but was not used for the sparse method or a combination of such factors. Sparsity becomes an important factor for efficiency when treating equations where more complicated solutions are to be expected which require polynomial approximations in the order of hundreds or thousands of coefficients.

5.2.2 High-order solutions

For an example which requires a higher n to solve with good accuracy, we consider

$$u_2'(x,k) = g_2(x,k) + \int_0^x ye^{x^2}u_2(y,k)\mathrm{d}y, \tag{22}$$

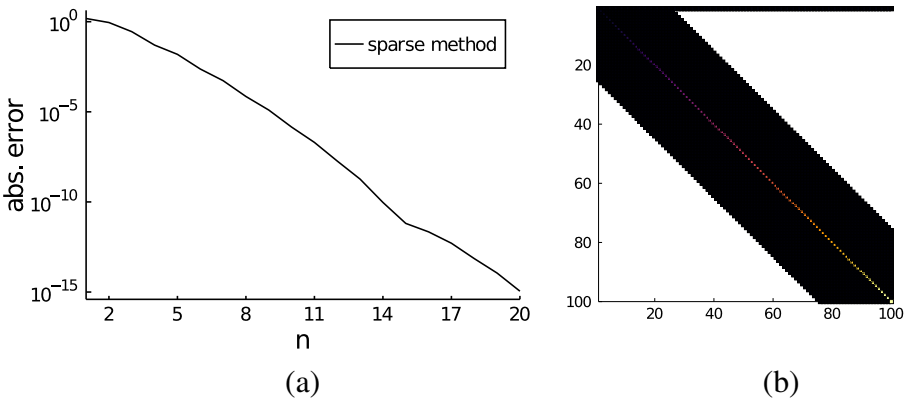

Fig. 4 **a** shows absolute error between analytic and computed solutions for u_1 in Eq. 20 for polynomial approximation of order n, **b** shows quasi-bandedness of full sparse method operator for $n = 100$

given initial condition

$$u_2(0, k) = 0, \tag{23}$$

and right-hand side function $g_2(x, k)$ defined by

$$g_2(x, k) = \frac{k}{k^2x^2 + 1} - \frac{e^{x^2}\arctan(kx)}{2k^2} + \frac{e^{x^2}x}{2k} - \frac{1}{2}e^{x^2}x^2\arctan(kx).$$

For all $k \in \mathbb{R}$ the analytic solution to this equation is given by

$$u_2(x, k) = \arctan(kx).$$

As this approximates a step-like function at $x = 0$ for increasing k (see Fig. 5a), it is easy to see why polynomial approximations quickly begin to require high orders. Figure 5b shows a spy plot of the quasi-banded integro-differential operator generated by our sparse method, while Fig. 6 shows the absolute error of some solutions obtained via the sparse spectral method. The solutions needs to be resolved in relatively high-order polynomial approximations, so the bandedness of the operator results in notable performance improvements compared to Chebfun's dense collocation method; see Table 4.

Table 3 Quantitative performance comparison of sparse method and Chebfun for Eq. 20

Method	CPU time	approx. order	abs. error
Sparse method	0.1 s	20	1.4e−15
Chebfun	0.2 s	18 (autom.)	2.7e−15

The automatically chosen convergence order was used for Chebfun's results. CPU time measured on Intel Core i7-6700T CPU @ 2.80GHz

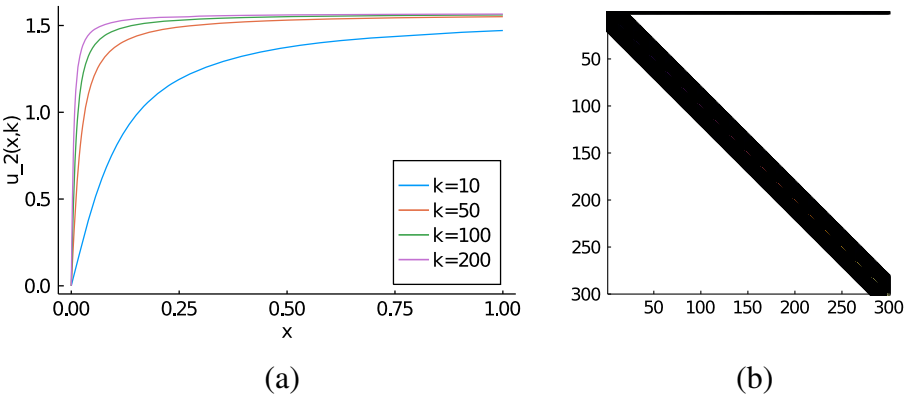

Fig. 5 **a** shows $u_2(x, k) = \arctan(kx)$ for some small values of k, **b** shows quasi-bandedness of full sparse method operator for Eq. 22 with $n = 300$. This banded structure makes computations for very high n not only possible but also fast

5.3 Bessel kernels with highly oscillatory solutions

Hale [26] discusses an integro-differential equation with Bessel function kernel, which appears in scattering applications and potential theory, on the basis of previous work on Bessel kernel Volterra equations by Xiang and Brunner [58]. We use this as the final numerical experiment in this section as it touches on the interesting case of highly oscillatory solutions without known analytic form. Specifically, we discuss the singularly perturbed version of the equation which appears in [26] and intentionally makes the problem significantly more oscillatory:

$$10^{-3}u''(x) + \omega^2 u(x) = g(x, \mu, \nu) - \omega \int_0^x J_\mu(\omega(x - y))u_2(y)\mathrm{d}y. \tag{24}$$

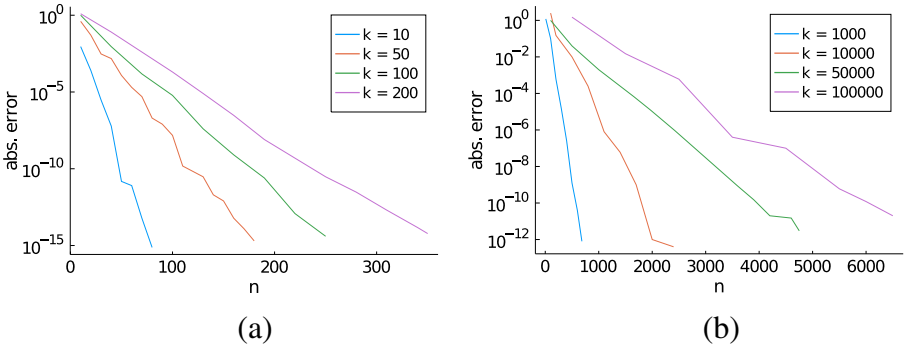

Fig. 6 **a** and **b** show absolute error between analytic and computed solutions for Eq. 22 for various values of k

Table 4 Quantitative performance comparison of sparse method and Chebfun for Eq. 22

k (sparse)	CPU time	approx. order	abs. error
100	0.3 s	300	6.0e−15
1000	0.4 s	800	2.9e−14
10,000	0.8 s	2000	9.3e−14
50,000	3.5 s	5000	2.8e−13
100,000	8.8 s	8000	8.1e−12
k (Chebfun)	CPU time	autom. order	abs. error
100	1.4 s	196	4.8e−14
1000	2.1 s	540	6.6e−12
10,000	10.7 s	1477	1.2e−09
50,000	10.4 s	2863	4.1e−08

The automatically chosen order was used for Chebfun's results, while the sparse method can generate higher accuracy results in less time. For $k = 100,000$ Chebfun issues an error after approximately 11 s that it may not have converged. CPU time measured on Intel Core i7-6700T CPU @ 2.80GHz

with $g(x, \mu, \nu)$ defined by

$$g(x, \mu, \nu) = J_{\mu+\nu}(\omega x) + \frac{1}{2x^2}((\nu-1)(\nu-2)J_{\nu-1}(\omega x) + (\nu+1)(\nu+2)J_{\nu+1}(\omega x)),$$

where J_μ are first kind Bessel functions, $\mu > 0$ and $\omega \in \mathbb{R}$. Equation 24 is further supplied with initial conditions

$$u(0) = u'(0) = 0.$$

To allow comparisons with [26], we will consider the example parameters

$$\nu = 3, \quad \mu = 2, \quad \omega = 20.$$

Analytic solutions to this equation are not known and convergence comparisons are thus made to high-order approximate solutions ($n = 2000$) instead. We plot the highly oscillatory solution to this in Fig. 7a and the convergence to the $n = 2000$ solution in Fig. 7b. Similarly to results in [26], we observe rapid exponential convergence once the polynomial order becomes sufficient to resolve the frequency of the oscillations. Better convergence up to machine precision is possible when using a more sophisticated balancing of approximation orders for the kernel, g and the solution, respectively, as opposed to linearly increasing the approximation order of each of them at the same time. This could also be done using an automated convergence algorithm if needed but this example is primarily presented to show the broad range of applicability even for oscillatory problems—as this particular Bessel kernel is ultimately a convolution kernel, methods which take the additional convolution kernel structure into account, e.g., Hale's method in [26], will generally outperform the general kernel method presented in this paper in accuracy or performance (in particular if operating in low polynomial approximation orders or if they themselves make use of sparsity structure).

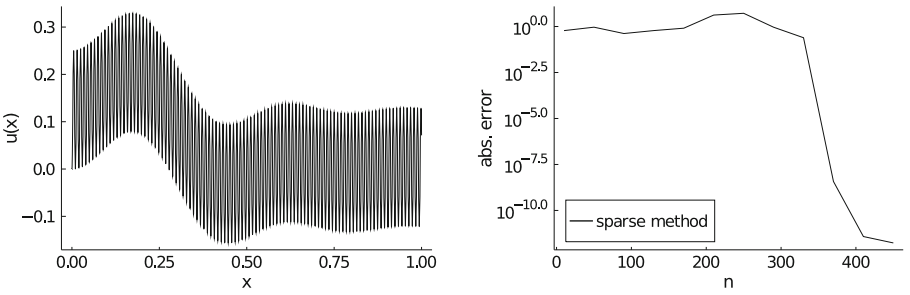

Fig. 7 **a** shows highly oscillatory solution for Eq. 24 for high approximation order $n = 2000$, **b** shows error of lower order approximations compared with the $n = 2000$ approximation as no analytic solutions are available

5.4 Numerical experiments with nonlinear equations

5.4.1 Set 1: Power nonlinearity Volterra integral equations

The simplest case of nonlinear Volterra integral equations, and thus also where most analytic solutions are available for direct comparison, is the case of power nonlinearities of the form $f(u) = u^m$ for some positive integer m. We thus consider the examples

$$u_1(x) = e^x + \frac{x(1 - e^{3x})}{3} + \int_0^x x u_1^3(y)\mathrm{d}y, \tag{25}$$

$$u_2(x) = \sin(x) + \frac{\sin^2(x)}{4} - \frac{x^2}{4} + \int_0^x (x - y)u_2^2(y)\mathrm{d}y, \tag{26}$$

whose analytic solutions are derived respectively via a Picard-type iteration and Adomian decomposition method in [56]:

$$u_1(x) = e^x,$$
$$u_2(x) = \sin(x).$$

As discussed above and as is true for any iterative method, there are now multiple parameters which may be fine-tuned to the problems at hand in order to achieve faster and more precise convergence. To that end, we may for example fine-tune the initial guess or the convergence cutoffs. As what can go wrong in a standard application case is of greater interest than what may happen in ideal circumstances, we omit such fine-tuning and instead simply supply a vector of all zeros of length n for Eq. 25 and a vector of all ones of length n for Eq. 26. We plot the maximal absolute errors between true and computed solutions in Fig. 8. We observe exponential convergence as n increases using simple Newton iteration without linesearch.

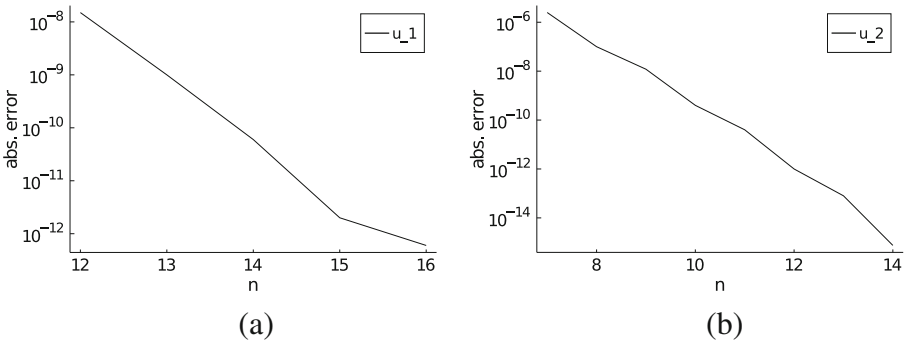

Fig. 8 Absolute error between analytic and computed solutions for $u_1(x)$ and $u_2(x)$ in Eqs. 25–26 for polynomial approximation of order n

5.4.2 Set 2: Numerical experiments with nonlinear VIDEs

For nonlinear VIDEs, we consider:

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}u_1(x) = -\frac{5}{3}\sin(x) + \frac{1}{3}\sin(2x) + \int_0^x \cos(x-y)u_1^2(y)\mathrm{d}y, \tag{27}$$

$$\frac{\mathrm{d}}{\mathrm{d}x}u_2(x) = x + \cos(x) - \tan(x) + \tan^2(x) + \int_0^x \left(\sin(x) + u_2^2(y)\right)\mathrm{d}y, \tag{28}$$

with initial conditions

$$u_1(0) = 0, \quad u_1'(0) = 1, \tag{29}$$

$$u_2(0) = 0. \tag{30}$$

Analytic solutions to these equations were derived in [56] using the variational iteration method:

$$u_1(x) = \sin(x),$$
$$u_2(x) = \tan(x).$$

As in Section 5.4, we avoid making educated guesses for the initial guess supplied to the algorithm and merely increase the maximal allowed length of the solution coefficient vector, i.e., the maximal polynomial degree of the computed approximation. The initial guess for Eq. 27 is a vector of all ones and the initial guess for Eq. 27 is a vector of all zeros of length n respectively. We plot the maximal absolute errors between analytic and computed solutions in Fig. 9. We again observe exponential convergence as n increases using Newton iteration without linesearch.

6 Notes on algorithm convergence

Convergence of the above-discussed method in the case of nonlinear equations arises as a function of the convergence properties of the root search algorithm that is utilized, combined with the proofs for the respective linear variants.

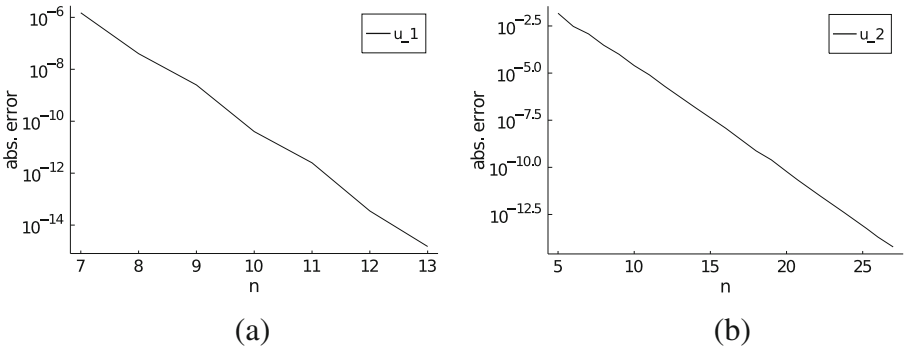

Fig. 9 Absolute error between analytic and computed solutions for $u_1(x)$ and $u_2(x)$ in Eqs. 27–28 with initial conditions in Eqs. 29–30 for polynomial approximation of order n

Proofs of convergence for second kind linear Volterra integro-differential equations may be given in a similar fashion to linear Volterra integral equations in [24] and differential equations in [27], the basic observation being that the full to-be-inverted operator is diagonally dominant for well-behaved functions and may be written as a compact perturbation of the identity, thus reducing the problem to standard finite section approximation convergence results, cf. [10, 33, 40, 51]. As seen in the linear case proof for first kind VIEs in [24], proofs for first kind equations would require a deeper functional analysis approach. The exponential nature of convergence for sufficiently smooth problems is inherited from the fact that the solution is approximated in terms of its coefficient vector in a basis of orthogonal polynomials.

7 Discussion

We have presented a competitively fast general kernel sparse spectral method for nonlinear Volterra integro-differential and integral equations which extends linear results in [24]. The method is notably not reliant on the structure of convolution kernels and applies for general kernels. Furthermore, as it does not rely on low rank approximations it is applicable in more general cases where these approximations fail. It thus combines very broad applicability with high performance and accuracy.

One noteworthy drawback of this method is that, although as discussed in the numerical experiments section in [24] the method may yield sensible results for some types of singular kernels, there are as of now no known guarantees for such cases. That said, the presented method was shown to be convergent and well-behaved with problems that may be well approximated in the specified polynomial bases, which allow for a very general range of kernels.

The numerical experiments in this paper serve an illustrative purpose—in a practical application setting one would choose more sophisticated and efficient root search algorithms than a simple Newton iteration without linesearch and make an educated initial guess for the root search based on background knowledge about the

structure of the problem instead of supplying simple zero or one filled coefficient vectors. These points were specifically ignored in this paper to illustrate that such more sophisticated methods are not required to achieve competitive performance and accuracy.

Acknowledgements The author would like to thank Sheehan Olver for reading a draft and providing helpful comments, and the anonymous reviewers for their useful comments and suggestions.

References

1. Agbolade, O.A., Anake, T.A.: Solutions of first-order Volterra type linear integrodifferential equations by collocation method. J. Appl. Math. **2017**, 1–5 (2017). https://doi.org/10.1155/2017/1510267
2. Allaei, S.S., Yang, Z.W., Brunner, H.: Collocation methods for third-kind VIEs. IMA J. Numer. Anal. **37**. https://doi.org/10.1093/imanum/drw033 (2017)
3. Apartsyn, A.S.: On some classes of linear Volterra integral equations abstract and applied analysis. https://doi.org/10.1155/2014/532409 (2014)
4. Atkinson, K.E.: A survey of numerical methods for solving nonlinear integral equations. Journal of Integral Equations and Applications **4**(1). https://doi.org/10.1216/jiea/1181075664 (1992)
5. Battles, Z., Trefethen, L.N.: An extension of MATLAB to continuous functions and operators. SIAM J. Sci. Comput. **25**(5). https://doi.org/10.1137/S1064827503430126 (2004)
6. Beals, R., Wong, R.: Special Functions and Orthogonal Polynomials. No. 153 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge (2016)
7. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671
8. Biazar, J., Ebrahimi, H.: Chebyshev wavelets approach for nonlinear systems of Volterra integral equations. Computers & Mathematics with Applications **63**(3), 608–616 (2012). https://doi.org/10.1016/j.camwa.2011.09.059
9. van den Bosch, F., Metz, J.A.J., Zadoks, J.C.: Pandemics of focal plant disease, a model. Phytopathology **89**(6), 495–505 (1999). https://doi.org/10.1094/PHYTO.1999.89.6.495
10. Böttcher, A., Silbermann, B., Karlovich, A. Analysis of Toeplitz Operators, 2nd edn. Springer Monographs in Mathematics. Springer, Berlin (2006). OCLC: 181538992
11. Brunner, H.: On the numerical solution of nonlinear Volterra integro-differential equations. BIT Numer. Math. **13**(4), 381–390 (1973). https://doi.org/10.1007/BF01933399
12. Brunner, H.: High-order Methods for the numerical solution of Volterra integro-differential equations. J. Comput. Appl. Math. **15**(3). https://doi.org/10.1016/0377-0427(86)90221-9 (1986)
13. Brunner, H.: Collocation Methods for Volterra Integral and Related Functional Differential Equations. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge (2004)
14. Burns, K.J., Vasil, G.M., Oishi, J.S., Lecoanet, D., Brown, B.P.: Dedalus: a flexible framework for numerical simulations with spectral methods. arXiv:1905.10388[astro-ph, physics:physics] (2019)
15. Driscoll, T.A.: Automatic spectral collocation for integral, integro-differential, and integrally reformulated differential equations. J. Comput. Phys. **229**(17). https://doi.org/10.1016/j.jcp.2010.04.029 (2010)

16. Driscoll, T.A., Bornemann, F., Trefethen, L.N.: The chebop system for automatic solution of differential equations. BIT Numerical Mathematics **48**(4). https://doi.org/10.1007/s10543-008-0198-4 (2008)
17. Dunkl, C.F., Xu, Y.: Orthogonal Polynomials of Several Variables, Second Edn. No. 155 in Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge (2014)
18. Ezquerro, J.A., Hernández, M.A., Romero, N.: Solving nonlinear integral equations of Fredholm type with high order iterative methods. J. Comput. Appl. Math. **236**(6). https://doi.org/10.1016/j.cam.2011.09.009 (2011)
19. Gautschi, W.: Orthogonal Polynomials: Computation and Approximation. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford (2004)
20. Geiser, J.: An iterative splitting approach for linear integro-differential equations. Appl. Math. Lett. **26**, 1048–1052 (2013). https://doi.org/10.1016/j.aml.2013.05.012
21. Ghasemi, M., Kajani, M.T., Babolian, E.: Numerical solutions of the nonlinear Volterra–Fredholm integral equations by using homotopy perturbation method. Appl. Math. Comput. **188**(1), 446–449 (2007). https://doi.org/10.1016/j.amc.2006.10.015
22. Gordji, M.E., Baghani, H., Baghani, O.: On existence and uniqueness of solutions of a nonlinear integral equation. https://doi.org/10.1155/2011/743923 (2011)
23. Gutleb, T.S.: TSGut/SparseVolterraExamples.jl: v0.1.1. https://doi.org/10.5281/zenodo.4382253 (2020)
24. Gutleb, T.S., Olver, S.: A sparse spectral method for Volterra integral equations using orthogonal polynomials on the triangle. SIAM J. Numer. Anal. **58**(3), 1993–2018 (2020). https://doi.org/10.1137/19M1267441
25. Hackbusch, W.: Integral Equations: Theory and Numerical Treatment. No. 120 in International Series of Numerical Mathematics. Basel, Birkhäuser (1995)
26. Hale, N.: An ultraspherical spectral method for linear Fredholm and Volterra integro-differential equations of convolution type. IMA J. Numer. Anal. **39**(4), 1727–1746 (2019). https://doi.org/10.1093/imanum/dry042
27. Hale, N., Olver, S.: A fast and spectrally convergent algorithm for Rational-Order fractional integral and differential equations. SIAM J. Sci. Comput. **40**. https://doi.org/10.1137/16M1104901 (2018)
28. Hethcote, H.W., Tudor, D.W.: Integral equation models for endemic infectious diseases. J. Math. Biol. **9**(1), 37–47 (1980). https://doi.org/10.1007/BF00276034
29. Heydari, M.H., Hooshmandasl, M.R., Mohammadi, F., Cattani, C.: Wavelets Method for solving systems of nonlinear singular fractional Volterra integro-differential equations. Commun. Nonlinear Sci. Numer. Simul. https://doi.org/10.1016/j.cnsns.2013.04.026 (2014)
30. Krimer, D.O., Putz, S., Majer, J., Rotter, S.: Non-markovian dynamics of a single-mode cavity strongly coupled to an inhomogeneously broadened spin ensemble. Phys. Rev. A **90**(4). https://doi.org/10.1103/PhysRevA.90.043852 (2014)
31. Krimer, D.O., Zens, M., Putz, S., Rotter, S.: Sustained photon pulse revivals from inhomogeneously broadened spin ensembles. Laser Photonics Rev. (6): 1023–1030. https://doi.org/10.1002/lpor.201600189 (2016)
32. Lepik, U.: Haar wavelet method for nonlinear integro-differential equations. Appl. Math. Comput. **176**. https://doi.org/10.1016/j.amc.2005.09.021 (2006)
33. Lintner, S.K., Bruno, O.P.: A generalized calderón formula for open-arc diffraction problems: theoretical considerations. P. Roy. Soc. Edinb. A **145**(2). https://doi.org/10.1017/S0308210512000807 (2015)
34. Meehan, M., O'Regan, D.: Existence theory for nonlinear Volterra integrodifferential and integral equations. Nonlinear analysis: theory. Methods & Applications **31**. https://doi.org/10.1016/S0362-546X(96)00313-6 (1998)
35. Micke, A., Bülow, M.: Application of Volterra integral equations to the modelling of the sorption kinetics of multi-component mixtures in porous media. Gas Separation & Purification **4**(3), 158–164 (1990). https://doi.org/10.1016/0950-4214(90)80018-G
36. Mogensen, P.K., Carlsson, K., Villemot, S., Lyon, S., Gomez, M., Rackauckas, C., Holy, T., Widmann, D., Kelman, T., Macedo, M.R.G., Benneti, Bojesen, T.A., Arakaki, T., Christ, S., Byrne, S., Lubin, M., Barton, D., Kwon, C., Lucibello, C., Riseth, A.N., Levitt, A.: JuliaNLSolvers/NLsolve.jl: v4.2.0. https://doi.org/10.5281/zenodo.3527404 (2019)
37. Nedaiasl, K., Bastani, A.F., Rafiee, A.: A product integration method for the approximation of the early exercise boundary in the american option pricing problem. Mathematical Methods in the Applied Sciences **42**(8), 2825–2841 (2019). https://doi.org/10.1002/mma.5553

38. Olver, F., Daalhuis, A., Lozier, D., Schneider, B., Boisvert, R., Clark, C., Miller, B., Saunders (eds.), B.V.: NIST Digital Library of Mathematical Functions (2018). https://dlmf.nist.gov/
39. Olver, S.: JuliaApproximation/ApproxFun.jl (2019). https://github.com/JuliaApproximation/ApproxFun.jl
40. Olver, S., Townsend, A.: A fast and well-conditioned spectral method. SIAM Rev. **55**(3). https://doi.org/10.1137/120865458 (2013)
41. Olver, S., Townsend, A.: A practical framework for infinite-dimensional linear algebra. In: 2014 First Workshop for High Performance Technical Computing in Dynamic Languages. IEEE, LA, USA (2014). https://doi.org/10.1109/HPTCDL.2014.10
42. Olver, S., Townsend, A., Vasil, G.: A sparse spectral method on triangles. arXiv:1902.04863 (2019)
43. Pachon, R., Platte, R.B., Trefethen, L.N.: Piecewise-smooth chebfuns. IMA Journal of Numerical Analysis **30**(4). https://doi.org/10.1093/imanum/drp008 (2010)
44. Prüss, J.: Evolutionary Integral Equations and Applications. Modern Birkhäuser Classics. Springer, Basel, New York (2012). OCLC: ocn796763028
45. Saeedi, H., Mohseni Moghadam, M.: Numerical solution of nonlinear Volterra integro-differential equations of arbitrary order by CAS wavelets. Commun. Nonlinear Sci. Numer. Simul. **16**(3), 1216–1226 (2011). https://doi.org/10.1016/j.cnsns.2010.07.017
46. Sahu, P.K., Ray, S.S.: Legendre wavelets operational method for the numerical solutions of nonlinear Volterra integro-differential equations system. Appl. Math. Comput. https://doi.org/10.1016/j.amc.2015.01.063 (2015)
47. Shayanfard, F., Laeli Dastjerdi, H., Maalek Ghaini, F.: collocation method for approximate solution of Volterra integro-differential equations of the third-kind. Appl. Numer. Math. https://doi.org/10.1016/j.apnum.2019.09.020 (2019)
48. Slevinsky, R.M.: Conquering the pre-computation in two-dimensional harmonic polynomial transforms. arXiv:1711.07866 (2017)
49. Slevinsky, R.M.: Fast and backward stable transforms between spherical harmonic expansions and bivariate fourier series. Appl. Comput. Harmon Anal. https://doi.org/10.1016/j.acha.2017.11.001 (2017)
50. Slevinsky, R.M.: FastTransforms v0.1.1. https://github.com/MikaelSlevinsky/FastTransforms. Original-date: 2018-03-15T23:11:52Z (2019)
51. Slevinsky, R.M., Olver, S.: A fast and well-conditioned spectral method for singular integral equations. J. Comput. Phys. **332**, 290–315 (2017). https://doi.org/10.1016/j.jcp.2016.12.009
52. Snowball, B., Olver, S.: Sparse spectral and-finite element methods for partial differential equations on disk slices and trapeziums. Stud. Appl. Math. **145**(1), 3–35 (2020)
53. Song, H., Yang, Z., Brunner, H.: Analysis of collocation methods for nonlinear Volterra integral equations of the third kind calcolo. https://doi.org/10.1007/s10092-019-0304-9 (2019)
54. Townsend, A., Olver, S.: The automatic solution of partial differential equations using a global spectral method. J. Comput. Phys. **299**. 10.1016/j.jcp.2015.06.031 (2015)
55. Unterreiter, A.: Volterra integral equation models for semiconductor devices. Mathematical Methods in the Applied Sciences **19**(6), 425–450 (1996). https://doi.org/10.1002/(SICI)1099-1476(199604)19:6⟨425::AID-MMA744⟩3.0.CO;2-M
56. Wazwaz, A.M.: Linear an Nonlinear Integral Equations: Methods and Applications. Higher Education Press, Beijing (2011)
57. Wazwaz, A.M.: The regularization method for Fredholm integral equations of the first kind. Computers & Mathematics with Applications **61**(10), 2981–2986 (2011)
58. Xiang, S., Brunner, H.: Efficient methods for Volterra integral equations with highly oscillatory Bessel kernels. BIT Numer. Math. **53**(1), 241–263 (2013). 10.1007/s10543-012-0399-8
59. Zakes, F., Sniady, P.: Application of Volterra Integral Equations in Dynamics of Multispan Uniform Continuous Beams Subjected to a Moving Load (2016). https://doi.org/10.1155/2016/4070627
60. Zhang, P., Hao, X.: Existence and uniqueness of solutions for a class of nonlinear integro-differential equations on unbounded domains in Banach spaces. Advances in Difference Equations **2018**. https://doi.org/10.1186/s13662-018-1681-0 (2018)
61. Zhu, L., Fan, Q.: Solving fractional nonlinear Fredholm integro-differential equations by the second kind Chebyshev wavelet. Commun. Nonlinear Sci. Numer. Simul. **17**. https://doi.org/10.1016/j.cnsns.2011.10.014 (2012)