SIAM J. NUMER. ANAL.
Vol. 57, No. 2, pp. 899–924

CONTINUOUS ANALOGUES OF KRYLOV SUBSPACE METHODS FOR DIFFERENTIAL OPERATORS[*]

MARC AURÈLE GILLES[†] AND ALEX TOWNSEND[‡]

Abstract. Analogues of the conjugate gradient method, minimum residual method, and generalized minimum residual method are derived for solving boundary value problems (BVPs) involving ordinary differential equations. Two challenges arise: imposing the boundary conditions on the solution while building up a Krylov subspace and guaranteeing convergence of the Krylov-based method on unbounded operators. Our approach employs projection operators to guarantee that the boundary conditions are satisfied, and we develop an operator preconditioner that ensures that an approximate solution is computed after a finite number of iterations. The developed Krylov methods are practical iterative BVP solvers that are particularly efficient when a fast operator-function product is available. An extension to partial differential operators is also presented.

Key words. Krylov methods, conjugate gradient, differential operators, spectral methods

AMS subject classifications. 65F10, 65N35, 47E05

DOI. 10.1137/18M1177810

1. Introduction.

Krylov subspace methods, such as the conjugate gradient (CG) method, the minimum residual method (MINRES), and the generalized minimum residual method (GMRES), are iterative algorithms that solve $Ax = b$ using matrix-vector products [40]. After k iterations, they typically compute an approximate solution to $Ax = b$ from the Krylov subspace $\mathcal{K}_k(A, b) = \text{Span}\{b, Ab, \ldots, A^{k-1}b\}$. They provide a toolkit for solving large sparse or structured linear systems, which are omnipresent in computational mathematics. Krylov subspace methods are particularly prevalent in the context of solving differential equations, where a differential equation $\mathcal{L}u = f$ associated with a set of boundary conditions is discretized into a typically large sparse linear system $Ax = b$, and a Krylov subspace method is used to solve the resulting linear system.

During a discussion at the Chebfun and Beyond conference in September 2012 attended by about 150 numerical analysts,[1] the question was raised: can we design operator Krylov methods to solve differential equations without the need for discretization? In other words, can we build a Krylov-like method for solving differential equations by building up a Krylov subspace through operator-function products? This is of particular interest for the spectral community as spectral discretizations are often dense, ill-conditioned (see Figure 1), and do not always inherit the structure of the continuous problem (e.g., spectral discretizations of self-adjoint operators and not necessarily symmetric).

For these reasons, Krylov methods are not ubiquitously employed in the spectral method community [9, 37], despite $n \times n$ Chebyshev-based spectral discretization matrices of (1.1) having a fast $\mathcal{O}(n \log n)$ matrix-vector product based on the FFT [22].

[*]Received by the editors March 28, 2018; accepted for publication (in revised form) February 19, 2019; published electronically April 30, 2019.
http://www.siam.org/journals/sinum/57-2/M117781.html

Funding: The work of the authors was supported by National Science Foundation grant 1645445.

[†]Center for Applied Mathematics, Cornell University, Ithaca, NY 14853 (mtg79@cornell.edu).

[‡]Department of Mathematics, Cornell University, Ithaca, NY 14853 (townsend@cornell.edu).

[1]The session was chaired by Nick Higham. The second author was a graduate student at the time and scribed the discussion, following Nick Trefethen's advice.

In this paper, we describe the first practical implementation of operator analogues of Krylov methods for solving two-point boundary value problems (BVPs) [8, Chap. 6]. Although the presented method does not directly extend to partial differential equations, we discuss possible ways forward in section 6. In order to simplify the exposition, we proceed with the assumption that $\mathcal{L}$ is a second-order operator, but we extend the methods to all even-ordered ODEs in section 5. Thus, we focus on a simple problem of the form:

$$\mathcal{L}u = f \quad \text{on } \Omega = (-1,1), \qquad u(\pm 1) = 0, \tag{1.1}$$

where $\mathcal{L}u = -(a(x)u'(x))' + b(x)u'(x) + c(x)u$, $\mathcal{L} : \mathcal{H}_0^1(\Omega) \cap \mathcal{H}^2(\Omega) \to L^2(\Omega)$, $a, b, c \in L^\infty(\Omega)$ and $f \in L^2(\Omega)$.

If there are no additional assumptions[2] on $\mathcal{L}$, then we propose an analogue of GMRES to solve (1.1) (see subsection 4.1). If $b(x) = 0$, then $\mathcal{L}$ is self-adjoint, which is analogous to a symmetric matrix, and we propose an analogue of MINRES (see subsection 4.2). When $a(x) > 0$, $b(x) = 0$, and $c(x) \geq 0$, $\mathcal{L}$ is self-adjoint with real positive eigenvalues [8, sect. 6.5, Thm. 1], which is analogous to a symmetric positive definite matrix, and we propose an analogue of the CG method (see section 2).

To see the difficulties in developing a Krylov-based method for differential operators, consider solving $-u''(x) = 1 - x^2$ on $\Omega = (-1,1)$ with $u(\pm 1) = 0$. The exact solution is $u(x) = (x^4 - 6x^2 + 5)/12$. A naive generalization of the Krylov subspace is $\mathcal{K}_k(\mathcal{L}, f) = \text{Span}\{f, \mathcal{L}f, \mathcal{L}(\mathcal{L}f), \ldots, \mathcal{L}^{k-1}f\}$ with $f = 1 - x^2$. Since $\mathcal{L}u = -u''$, this leads to $\mathcal{K}_k(\mathcal{L}, f) = \text{Span}\{1 - x^2, 2\}$ for $k \geq 2$. This example illustrates that such an approach is flawed, as $\mathcal{K}_k(\mathcal{L}, f)$ does not contain a good approximation to the exact solution. Moreover, the boundary conditions are not imposed because $\mathcal{K}_k(\mathcal{L}, f) \not\subset \mathcal{H}_0^1(\Omega)$ for $k \geq 2$. There are at least three major theoretical issues to overcome.

Problem 1.1. Since $f \notin \mathcal{H}_0^1(\Omega)$ and $\text{Range}(\mathcal{L}) \not\subset \mathcal{H}_0^1(\Omega)$, how does one construct a Krylov subspace that satisfies the boundary conditions? Our answer involves using orthogonal projection operators to ensure that each term in the Krylov subspace is in $\mathcal{H}_0^1(\Omega)$ (see subsection 2.1), and solving an ancillary problem (see subsection 2.5).

Problem 1.2. Since $\mathcal{L} : \mathcal{H}_0^1(\Omega) \cap \mathcal{H}^2(\Omega) \to L^2(\Omega)$, how does one repeatedly apply operator-function products that are necessary to build up a Krylov subspace? To achieve this, we use an orthogonal projection operator and a preconditioner that acts as a "smoother" (see subsection 2.2).

Problem 1.3. Since $\mathcal{L}$ is an unbounded operator, how does one construct a Krylov method that rapidly converges to the solution of (1.1)? Our answer is to use operator preconditioners that allow for our Krylov iterations to be terminated after a finite number of iterations with an approximate solution (see subsection 2.4).

The Krylov methods that we develop solve (1.1) by directly applying $\mathcal{L}$ to functions, and we prove that the iterates from our preconditioned CG method geometrically converge to the solution (see Corollary 2.8). Our operator Krylov methods are not equivalent to matrix Krylov methods applied to a standard discretization of (1.1), and offer several advantages: (1) operator preconditioners are motivated by the differential operator as opposed to the properties of a discretization scheme, (2) the resulting CG method can always be applied to (1.1) with $a(x) > 0$, $b(x) = 0$, and $c(x) \geq 0$ without the need for structure-preserving discretizations [34, Chap. 4],

[2]We note that the assumption already made on $\mathcal{L}$ is a slight restriction from the more typical $\mathcal{L} : \mathcal{H}_0^1(\Omega) \to \mathcal{H}^{-1}(\Omega)$ setup. However, this restriction is natural in the present context where we develop practical algorithms which apply $\mathcal{L}$ to functions by weak differentiation operations and function products instead of having to revert to a bilinear form interpretation of the function product (see section 3).

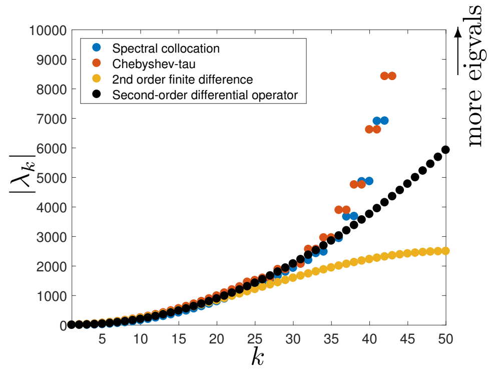

FIG. 1. *Spectra of* 50×50 *discretizations of* $\mathcal{L}u = -u''$ *with zero Dirichlet boundary conditions. Similar figures appear in* [41, Fig. 1]. *Spectral collocation (blue dots)* [37], *and Chebyshev tau (red dots)* [26] *discretizations typically have spectra that grow asymptotically faster than the spectra of the underlying differential operator (black dots), while the spectra of finite difference (yellow dots)* [18] *discretizations grow asymptotically slower. Most popular spectral discretizations are more ill-conditioned than expected, leading to poor convergence properties of Krylov subspace solvers. Our operator Krylov methods avoid discretizing BVPs and employs preconditioners that are motivated from the differential operator (see subsection* 2.2*).*

(3) the iterates converge to the desired solution of (1.1), as opposed to the solution of a discretization, and (4) the method is fully adaptive: it automatically chooses the complexity needed to represent each of the iterates.

Several attempts to develop operator Krylov methods for differential equations have been proposed that we believe date back to 1967 [4], where it was shown that an operator CG method can be reduced to a sequence of Poisson problems with Dirichlet boundary conditions. In 2009, a promising differential GMRES method for computing oscillatory integrals [24] was developed in the context of spectral methods, but it has remained unclear how to successfully incorporate boundary conditions. A theoretical foundation for a CG method on ordinary and partial differential operators [21] was introduced in 2015. The authors use a Riesz map $\tau : \mathcal{H}^{-1}(\Omega) \to \mathcal{H}_0^1(\Omega)$ to precondition a differential operator [21, Chap. 4] and successfully construct a Krylov subspace of the form $\mathrm{Span}\{\tau f, \tau\mathcal{L}\tau f, (\tau\mathcal{L})^2\tau f, \ldots\}$. This work is an insightful theoretical framework and our paper expands on their contribution in order to develop a collection of practical Krylov methods for solving (1.1).

Though we do not discretize the differential operator itself, for our operator Krylov methods to be of practical interest, one must employ an approximation space for the solution and right-hand side (see section 3). Unlike most BVP solvers, the approximation space can be all of $\mathcal{H}_0^1(\Omega)$ or an infinite dimensional dense subspace of $\mathcal{H}_0^1(\Omega)$. This allows one to implement highly adaptive Krylov subspace methods that automatically resolve the solution to machine precision (see section 3).

Intuitively, our main idea is to modify the operator-function products with $\mathcal{L}$ while preserving the weak form of (1.1). That is, we respect the bilinear form [8, p. 316] associated with (1.1), i.e.,

(1.2)
$$\mathcal{B}[\phi,\psi] = \int_{-1}^{1} a(x)\phi'(x)\psi'(x) + b(x)\phi'(x)\psi(x) + c(x)\phi(x)\psi(x)dx, \qquad \phi,\psi \in \mathcal{H}_0^1(\Omega),$$

as well as the weak form of the solution as $\mathcal{B}[u,\psi] = \langle f,\psi\rangle$ for all $\psi \in \mathcal{H}_0^1(\Omega)$. Here, and throughout the paper, we use $\langle\cdot,\cdot\rangle$ to denote the standard L^2 inner-product and $\|\psi\|^2 = \langle\psi,\psi\rangle$.

The paper is structured as follows. In section 2 we derive an unpreconditioned and preconditioned CG method for solving (1.1) when $\mathcal{L}$ is a self-adjoint second-order differential operator with $a(x) > 0$, $b(x) = 0$, and $c(x) \geq 0$. In section 3 we use our CG theory to develop practical iterative BVP solvers for (1.1). In section 4, we extend our CG method to operator analogues of MINRES and GMRES. In section 5 we show how our ideas can be applied to higher-order BVPs, and in section 6 we tentatively consider PDEs.

2. The CG method for differential operators.

The CG method for matrices is an iterative algorithm for solving $Ax = b$, where A is a symmetric positive definite matrix [15]. It constructs iterates $x_0 = 0, x_1, x_2, \ldots,$ such that x_k is the best approximate from $\mathcal{K}_k(A, b)$ as measured by the energy norm. That is,

$$x_k = \operatorname*{arg\,min}_{y\in\mathcal{K}_k(A,b)} \|x - y\|_A, \qquad \mathcal{K}_k(A, b) = \text{Span}\left\{b, Ab, \ldots, A^{k-1}b\right\},$$

where $\|y\|_A^2 = y^T Ay$ and $x = A^{-1}b$ is the exact solution. The fact that $\|\cdot\|_A$ defines a norm is central to the development and analysis of the CG method for matrices [19, sect. 5.6].

Just like symmetric positive definite matrices, self-adjoint differential operators with $a(x) > 0$ and $c(x) \geq 0$ have real positive eigenvalues and an orthogonal basis of eigenfunctions [8, sect. 6.5, Thm. 1]. The analogue of the energy norm in this setting is $\|\phi\|_{\mathcal{L}}^2 = \mathcal{B}[\phi, \phi]$ for $\phi \in \mathcal{H}_0^1(\Omega)$, where $\mathcal{B}$ is the bilinear form associated to $\mathcal{L}$ in (1.2). The fact that $\|\cdot\|_{\mathcal{L}}$ defines a norm is equally important for the development and analysis of a CG method for (1.1).

If $p_0, p_1, \ldots,$ form a complete basis for $\mathcal{H}_0^1(\Omega)$ so that $\mathcal{B}[p_i, p_j] = 0$ for $i \neq j \geq 0$, then since $f \in L^2(\Omega)$, we may formally write the solution to (1.1) as

$$u = \sum_{j=0}^{\infty} \frac{\langle f, p_j\rangle}{\mathcal{B}[p_j, p_j]} p_j.$$

Our CG method carefully constructs functions $p_0, p_1, \ldots,$ sequentially, such that $\mathcal{B}[p_i, p_j] = 0$ for $i \neq j$, in the hope that we may not need all of them to obtain a good approximation to u.

2.1. The unpreconditioned CG method with a restricted right-hand side.

In order to tackle the first major issue highlighted in the introduction (see Problem 1.1), we compose $\mathcal{L}$ with a projection operator[3] to ensure that any solution from the constructed Krylov subspace satisfies the zero Dirichlet conditions of (1.1).

Let $\mathcal{V}_0$ be an approximation space for the solution of (1.1). We wish to construct a projection onto $\mathcal{V}_0$ and apply it after each operator-function product so that the constructed Krylov subspace is a subspace of $\mathcal{V}_0$. We temporarily make the following assumptions.

Assumption 2.1. $\mathcal{V}_0$ is a closed (potentially infinite dimensional) subspace of the solution space $\mathcal{H}_0^1(\Omega) \cap \mathcal{H}^2(\Omega)$.

Assumption 2.2. $f \in \mathcal{V}_0$.

In subsection 2.2, we introduce a preconditioner that acts as a "smoother" to eliminate the need for Assumption 2.1 and allows us to set $\mathcal{V}_0 = \mathcal{H}_0^1(\Omega)$. We avoid Assumption 2.2 by solving an ancillary problem (see subsection 2.5).

[3]The idea of composing a matrix with a projection operator to generate a Krylov subspace is also used for solving saddle-point problems [12].

Proceeding under Assumptions 2.1 and 2.2, we define an orthogonal projection operator onto $\mathcal{V}_0$ (because $\mathcal{V}_0$ is a closed subspace of $L^2(\Omega)$) as

$$\Pi_{\mathcal{V}_0}\phi = \operatorname*{arg\,min}_{p\in\mathcal{V}_0} \|\phi - p\|, \qquad \Pi_{\mathcal{V}_0} : L^2(\Omega) \to \mathcal{V}_0.$$

We work with the modified operator $\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0} : L^2(\Omega) \to \mathcal{V}_0$, where $\Pi^*_{\mathcal{V}_0} : L^2(\Omega) \to \mathcal{V}_0$ is the adjoint of $\Pi_{\mathcal{V}_0}$ over the L^2 inner-product. Since $\Pi_{\mathcal{V}_0}$ is an orthogonal projection, it is self-adjoint, i.e., $\Pi^*_{\mathcal{V}_0} = \Pi_{\mathcal{V}_0}$ [29, Chap. 5]. This is important as it implies that the range of $\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0}$ is $\mathcal{V}_0$, and that the operator $\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0}$ is self-adjoint. Consequently, it is reasonable to imagine applying a CG method with $\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0}$.

The operator $\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0} : L^2(\Omega) \to \mathcal{V}_0$ is well defined since $\mathcal{V}_0 \subset \mathcal{H}^1_0(\Omega) \cap \mathcal{H}^2(\Omega)$, and we are interested in Krylov subspaces of the form

$$\mathcal{K}_k(\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0}, f) = \operatorname{Span}\left\{f, \Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0} f, \ldots, (\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0})^{k-1} f\right\}, \qquad k \geq 1. \tag{2.1}$$

Since $f \in \mathcal{V}_0$, we know that $\mathcal{K}_k(\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0}, f) \subseteq \mathcal{V}_0$ so that the boundary conditions are successfully incorporated into the Krylov subspace. Therefore, any iterative method that constructs iterates from the Krylov subspace in (2.1) automatically imposes zero Dirichlet boundary conditions.

An unpreconditioned CG method can now be derived that generates iterates $u_0 = 0, u_1, u_2, \ldots,$ such that

$$u_k = \operatorname*{arg\,min}_{v\in\mathcal{K}_k(\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0}, f)} \|u - v\|_{\mathcal{L}},$$

where u is the exact solution to (1.1).[4] The derivation of this method follows almost immediately from the CG method for matrices [39, Alg. 38.1], where in the derivation terms of the form x^TAy are replaced by $\mathcal{B}[\phi, \psi]$, x^Ty by $\langle\phi, \psi\rangle$, and Ax by $\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0}\phi$. The resulting unpreconditioned CG method for (1.1) is given in Algorithm 2.2. We also give the matrix CG method in Algorithm 2.1 for comparison, and we emphasize that the two algorithms are essentially the same except the operations Algorithm 2.1 are with vectors and matrices while the operations in Algorithm 2.2 are with functions and operators.

Algorithm 2.1 The CG method for solving $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix and $b \in \mathbb{R}^{n\times 1}$.

```
1: Set $x_0 = 0$, $r_0 = b$, and $p_0 = b$
2: for $k = 0, 1, \ldots$ (until converged) do
3:     $\alpha_k = r_k^T r_k / (p_k^T A p_k)$
4:     $x_{k+1} = x_k + \alpha_k p_k$
5:     $r_{k+1} = r_k - \alpha_k A p_k$
6:     $\beta_k = r_{k+1}^T r_{k+1} / r_k^T r_k$
7:     $p_{k+1} = r_{k+1} + \beta_k p_k$
8: end for
```

Algorithm 2.2 The CG method for (1.1), where $\mathcal{L}$ is self-adjoint with $a(x) > 0$ and $c(x) \geq 0$, and $f \in \mathcal{V}_0$.

```
1: Set $u_0 = 0$, $r_0 = f$, and $p_0 = f$
2: for $k = 0, 1, \ldots$ (until converged) do
3:     $\alpha_k = \langle r_k, r_k\rangle / \mathcal{B}[p_k, p_k]$
4:     $u_{k+1} = u_k + \alpha_k p_k$
5:     $r_{k+1} = r_k - \alpha_k \Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0} p_k$
6:     $\beta_k = \langle r_{k+1}, r_{k+1}\rangle / \langle r_k, r_k\rangle$
7:     $p_{k+1} = r_{k+1} + \beta_k p_k$
8: end for
```

[4]This follows from the fact that the discretization error is $\mathcal{B}$-orthogonal to the algebraic error in a Galerkin method [19, Thm. 2.5.2].

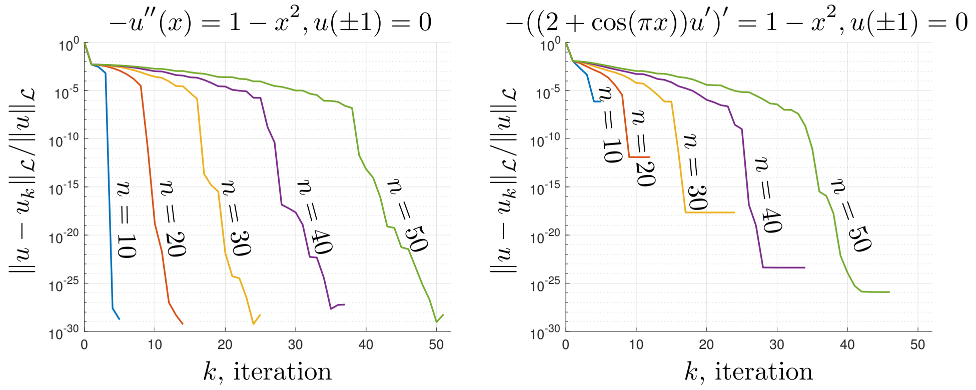

FIG. 2. *Convergence of the unpreconditioned CG method when* $\mathcal{V}_0 = \{v \in \mathcal{P}_n : v(\pm 1) = 0\}$ *and* $10 \leq n \leq 50$*, where* $\mathcal{P}_n$ *is the space of polynomials of degree* $\leq n$*. Left: The CG error when solving* $-u'' = 1 - x^2$ *on* $(-1, 1)$ *and* $u(\pm 1) = 0$*. Right: The CG error when solving* $-((2 + \cos(\pi x))u')' = 1 - x^2$ *on* $(-1, 1)$ *and* $u(\pm 1) = 0$*. The unpreconditioned CG method here is rarely useful because differential operators are unbounded and the number of required CG iterations is generically* $\dim(\mathcal{V}_0)$*. To overcome this, we develop operator preconditioners (see subsection* 2.2*).*

For Algorithm 2.2 to be well-defined we must check that: (1) $r_0, r_1, \ldots,$ are in $L^2(\Omega)$ so that $\langle r_k, r_k\rangle$ is valid, (2) $p_0, p_1, \ldots,$ are in $L^2(\Omega)$ so that $\Pi^*_{\mathcal{V}_0}\mathcal{L}\Pi_{\mathcal{V}_0}p_k$ is well defined, and (3) $p_0, p_1, \ldots,$ are in $\mathcal{H}^1_0(\Omega)$ so that $\mathcal{B}[p_k, p_k]$ is valid. All these statements hold when $f \in \mathcal{V}_0 \subset \mathcal{H}^1_0(\Omega) \cap \mathcal{H}^2(\Omega)$ and can be verified by mathematical induction.

The CG method in Algorithm 2.2 is theoretically justified for any $\mathcal{V}_0$ that is a closed subspace of $\mathcal{H}^1_0(\Omega) \cap \mathcal{H}^2(\Omega)$. In particular, this includes the space $\mathcal{V}_0 = \{v \in \mathcal{P}_n : v(\pm 1) = 0\}$ for some integer n, where $\mathcal{P}_n$ is the space of polynomials of degree $\leq n$. Furthermore, if the basis for $\mathcal{P}_n$ is selected to be the Legendre polynomials, then the CG method in Algorithm 2.2 is closely related to applying the CG method to a Legendre–Galerkin discretization of (1.1) [34, sect. 4.1]. The advantage of Algorithm 2.2 is that it provides important insights into how to derive a preconditioned CG method (see subsection 2.3).

The convergence of the unpreconditioned CG method in Algorithm 2.2 is generically poor (see Figure 2). The unboundedness of the differential operator means that $k = \dim(\mathcal{V}_0)$ iterations are typically necessary (see Figure 2) and, in our setting, $\mathcal{V}_0$ could potentially be an infinite dimensional subspace.

2.2. Operator preconditioning.

Improving the convergence of Algorithm 2.2 requires the development of preconditioners. The preconditioned CG method for solving $Ax = b$ is equivalent to applying the CG method to $R^TARy = R^Tb$, where $x = Ry$ and R is a square matrix [23, sect. 8.1]. Motivated by this, we consider solving

$$\mathcal{R}^*\mathcal{L}\mathcal{R}v = \mathcal{R}^*f \quad \text{on } \Omega = (-1, 1), \qquad (\mathcal{R}v)(\pm 1) = 0, \tag{2.2}$$

where $\mathcal{R} : L^2(\Omega) \to L^2(\Omega)$ is a linear operator and $\mathcal{R}^*$ is the adjoint of $\mathcal{R}$, i.e., $\langle \mathcal{R}^*\phi, \psi\rangle = \langle \phi, \mathcal{R}\psi\rangle$ for all $\phi, \psi \in L^2(\Omega)$. We call $\mathcal{R}$ an *operator preconditioner.*[5]

[5]In the Petrov–Galerkin literature, the concept of "operator preconditioning" is similar and refers to a recipe for constructing preconditioners so that they are robust with respect to the choice of trial and test basis [16]. A related concept is "equivalent preconditioners," where one constructs a preconditioner by simplifying the given differential operator [2].

We make the following requirements on the operator preconditioner $\mathcal{R} : L^2(\Omega) \to L^2(\Omega)$, which appear to be necessary in our framework to overcome the remaining two major issues highlighted in the introduction (see Problems 1.2 and 1.3).

Bounded. The preconditioner $\mathcal{R} : L^2(\Omega) \to L^2(\Omega)$ is a bounded linear operator. That is, $\|\mathcal{R}\|_{\text{op}} = \sup_{\phi \in L^2(\Omega), \|\phi\|=1} \|\mathcal{R}\phi\| < \infty$.

Smoother. The preconditioner and its adjoint over $L^2(\Omega)$ are smoothers, i.e., $\mathcal{R} : L^2(\Omega) \to \mathcal{H}^1(\Omega)$, $\mathcal{R} : \mathcal{H}^1(\Omega) \to \mathcal{H}^2(\Omega)$, $\mathcal{R}^* : L^2(\Omega) \to \mathcal{H}^1(\Omega)$, and $\mathcal{R}^* : \mathcal{H}^1(\Omega) \to \mathcal{H}^2(\Omega)$.[6]

Preconditioner for the Laplacian. There are constants $0 < \gamma_0 \leq \gamma_1 < \infty$ such that $\gamma_0\|\phi\|^2 \leq \|(\mathcal{R}\phi)'\|^2 \leq \gamma_1\|\phi\|^2$ for all $\phi \in L^2(\Omega)$.

A natural operator preconditioner for (1.1), and our canonical choice, is the indefinite integration operator $\mathcal{R} : L^2(\Omega) \to L^2(\Omega)$, defined as

$$(\mathcal{R}\phi)(x) = \int_{-1}^{x} \phi(s)ds, \qquad (\mathcal{R}^*\phi)(x) = \int_{x}^{1} \phi(s)ds, \qquad \phi \in L^2(\Omega). \tag{2.3}$$

The preconditioner and its adjoint act as "smoothers" and $\|\mathcal{R}\|_{\text{op}} = 4/\pi < \infty$ [14, Prob. 188]. If $\mathcal{L}u = -u''$, then the associated bilinear form of the operator $\mathcal{R}^*\mathcal{L}\mathcal{R}$ is

$$\mathcal{B}[\mathcal{R}\phi, \mathcal{R}\psi] = \int_{-1}^{1}\left(\int_{-1}^{x}\phi(s)ds\right)'\left(\int_{-1}^{x}\psi(s)ds\right)'dx = \int_{-1}^{1}\phi(x)\psi(x)dx = \langle \phi, \psi \rangle ,$$

where $\phi, \psi \in L^2(\Omega)$. Therefore, $\mathcal{R}$ is a preconditioner for the Laplacian with $\gamma_0 = \gamma_1 = 1$ so that the $\mathcal{R}$ in (2.3) satisfies all of our requirements.

The integral preconditioner in (2.3) appears throughout the literature and is exploited to construct preconditioners for finite element discretizations [17] as well as for spectral Galerkin discretizations [3, Chap. 4].

2.3. The preconditioned CG method. With an operator preconditioner in hand, we are able to derive a satisfying operator CG method. In order for $\mathcal{H}_0^1(\Omega)$ to be the solution space for $u = \mathcal{R}v$ in (2.2), the space $\mathcal{W}_0 = \{\phi \in L^2(\Omega) : \mathcal{R}\phi \in \mathcal{H}_0^1(\Omega)\}$ must be the approximation space for v in (2.2). Moreover, instead of assuming that $f \in \mathcal{V}_0$, we must now work under the the following assumption.

Assumption 2.3. $\mathcal{R}^*f \in \mathcal{W}_0$.

We no longer need Assumption 2.1, and we remove Assumption 2.3 in subsection 2.5. Since we are using a preconditioner and the approximation space for the solution of (2.2) is $\mathcal{W}_0$, we first need to design an orthogonal projection operator $\Pi_{\mathcal{W}_0} : L^2(\Omega) \to \mathcal{W}_0$. This task appears challenging for general preconditioners $\mathcal{R}$. However, when $\mathcal{R}$ is the indefinite integral preconditioner in (2.3), we note that $\mathcal{W}_0$ is the space of $L^2(\Omega)$ functions with zero mean. Moreover, $(\mathcal{R}\phi)(-1) = 0$ for all $\phi \in L^2(\Omega)$, and hence we find that the orthogonal projection $\Pi_{\mathcal{W}_0} : L^2(\Omega) \to \mathcal{W}_0$ is given by

$$\Pi_{\mathcal{W}_0}\phi = \phi - \frac{1}{2}\int_{-1}^{1}\phi(s)ds.$$

It is easy to verify that this projection operator is self-adjoint, and thus orthogonal:

$$\langle \Pi_{\mathcal{W}_0}\phi, \psi \rangle = \langle \phi, \psi \rangle - \frac{1}{2}\int_{-1}^{1}\phi(s)ds\int_{-1}^{1}\psi(s)ds = \langle \phi, \Pi_{\mathcal{W}_0}\psi \rangle , \qquad \phi, \psi \in L^2(\Omega).$$

[6]Note that if $f \in L^2(\Omega)$ this implies that $\mathcal{R}^*f \in \mathcal{H}^1(\Omega)$ and $\mathcal{R}^*\mathcal{L}\mathcal{R} : \mathcal{H}^1(\Omega) \to \mathcal{H}^1(\Omega)$.

Given an orthogonal projection operator $\Pi_{\mathcal{W}_0} : L^2(\Omega) \to \mathcal{W}_0$, we can derive a preconditioned CG method that constructs iterates $v_0 = 0, v_1, v_2, \ldots,$ so that $u_k = \mathcal{R}v_k$ approximates the solution to (1.1). The Krylov subspace of interest is now

(2.4)
$$\mathcal{K}_k(\mathcal{T}, \mathcal{R}^* f) = \text{Span}\left\{\mathcal{R}^* f, \mathcal{T}(\mathcal{R}^* f), \ldots, \mathcal{T}^{k-1}\mathcal{R}^* f\right\}, \qquad \mathcal{T} = \Pi^*_{\mathcal{W}_0}\mathcal{R}^*\mathcal{L}\mathcal{R}\Pi_{\mathcal{W}_0},$$

where $\mathcal{K}_k(\mathcal{T}, \mathcal{R}^* f) \subset \mathcal{W}_0$ because Assumption 2.3 ensures that $\mathcal{R}^* f \in \mathcal{W}_0$. The associated preconditioned CG method is given in Algorithm 2.3.

Algorithm 2.3 The preconditioned CG method for (1.1), where $\mathcal{L}$ is self-adjoint with $a(x) > 0$ and $c(x) \geq 0$, and $\mathcal{R}^* f \in \mathcal{W}_0$.

```
1: Set $v_0 = 0$, $r_0 = \mathcal{R}^* f$, and $p_0 = \mathcal{R}^* f$
2: for $k = 0, 1, \ldots,$ (until converged) do
3:     $\alpha_k = \langle r_k, r_k\rangle / \mathcal{B}[\mathcal{R}p_k, \mathcal{R}p_k]$
4:     $v_{k+1} = v_k + \alpha_k p_k$
5:     $r_{k+1} = r_k - \alpha_k \mathcal{T} p_k$
6:     $\beta_k = \langle r_{k+1}, r_{k+1}\rangle / \langle r_k, r_k\rangle$
7:     $p_{k+1} = r_{k+1} + \beta_k p_k$
8:     $u_{k+1} = \mathcal{R}v_{k+1}$
9: end for
```

To verify that Algorithm 2.3 is well defined we check that: (1) $r_0, r_1, \ldots,$ are in $L^2(\Omega)$ so that $\langle r_k, r_k\rangle$ is valid, (2) $p_0, p_1, \ldots,$ are in $L^2(\Omega)$ so that $\mathcal{T}p_k$ and $\mathcal{B}[\mathcal{R}p_k, \mathcal{R}p_k]$ are valid operations. All these statements hold when $\mathcal{R}^* f \in \mathcal{W}_0$ and can be proved by mathematical induction.

The preconditioned CG method in Algorithm 2.3 immediately inherits many of the theoretical properties from the CG method for matrices [23]. Here are two immediate facts that are analogous to familiar results for the matrix CG method.

LEMMA 2.4. *The functions $r_0, r_1, \ldots,$ in Algorithm 2.3 satisfy $\langle r_i, r_j\rangle = 0$ for $i \neq j$. Moreover, the functions $p_0, p_1, \ldots,$ satisfy $\mathcal{B}[\mathcal{R}p_i, \mathcal{R}p_j] = 0$ for $i \neq j$.*

Proof. The constant α_k is selected so that $\langle r_{k+1}, r_k\rangle = 0$ for $k \geq 0$. This gives the formula $\alpha_k = \langle r_k, r_k\rangle / \mathcal{B}[\mathcal{R}r_k, \mathcal{R}p_k]$, which can be simplified to the formula in Algorithm 2.3 since $r_{k+1} = p_{k+1} - \beta_k p_k$. The constant β_k is selected so that $\mathcal{B}[\mathcal{R}p_{k+1}, \mathcal{R}p_k] = 0$ for $k \geq 0$. This gives the formula $\beta_k = -\mathcal{B}[\mathcal{R}r_{k+1}, \mathcal{R}p_k]/\mathcal{B}[\mathcal{R}p_k, \mathcal{R}p_k]$, which can be simplified to the formula in Algorithm 2.3 since $r_{k+1} = r_k - \alpha_k \mathcal{T}p_k$. The result immediately follows. ◻

Lemma 2.4 also shows that Algorithm 2.3 is solving a best approximation problem.

THEOREM 2.5. *Let $u_0 = 0, u_1, \ldots,$ be the CG iterates from Algorithm 2.3 and u the solution to (1.1). Then,*

$$u_k = \underset{p \in \mathcal{X}_k}{\arg\min} \|u - p\|_{\mathcal{L}}, \qquad k \geq 1,$$

where $\mathcal{X}_k = \{p \in \mathcal{H}^1_0(\Omega) : p = \mathcal{R}q, q \in \mathcal{K}_k(\mathcal{T}, \mathcal{R}^* f)\}$.

Proof. From Lemma 2.4, we find that $\mathcal{B}[\mathcal{R}(v - v_k), \mathcal{R}p_j] = 0$ for $j \geq k + 1$, where $u = \mathcal{R}v$. In other words, we have

$$v_k = \underset{q \in \mathcal{K}_k(\mathcal{T}, \mathcal{R}^* f)}{\arg\min} \|v - q\|_{\mathcal{T}}.$$

Since $\|v-q\|_{\mathcal{T}}^2 = \mathcal{B}[\mathcal{R}(v-q),\mathcal{R}(v-q)] = \mathcal{B}[u-\mathcal{R}q, u-\mathcal{R}q] = \|u-\mathcal{R}q\|_{\mathcal{L}}^2$, this is equivalent to $v_k = \arg\min_{q\in\mathcal{K}_k(\mathcal{T},\mathcal{R}^*f)} \|u-\mathcal{R}q\|_{\mathcal{L}}$. Finally, we note that $u_k = \mathcal{R}v_k$, and therefore, $u_k = \arg\min_{p\in\mathcal{X}_k} \|u-p\|_{\mathcal{L}}$, where $\mathcal{X}_k = \{p \in \mathcal{H}_0^1(\Omega) : p = \mathcal{R}q, q \in \mathcal{K}_k(\mathcal{T},\mathcal{R}^*f)\}$. ∎

Theorem 2.5 shows that Algorithm 2.3 is calculating the best approximation from $\mathcal{X}_k$ to u in the $\|\cdot\|_{\mathcal{L}}$ norm and also guarantees that the error $e_k = \|u-u_k\|_{\mathcal{L}}$ is monotonically nonincreasing, i.e.,

$$\|u-u_{k+1}\|_{\mathcal{L}} \leq \|u-u_k\|_{\mathcal{L}}, \qquad k\geq 0.$$

In practice, designing good preconditioners is paramount for an efficient BVP solver. One could imagine being confronted with the same dilemma as preconditioning the CG method for matrices. On the one hand, we want to select $\mathcal{R}$ so that $\mathcal{R}\phi$ can be computed efficiently for any $\phi\in\mathcal{W}_0$. On the other hand, we want $\mathcal{T}$ to be a well-conditioned operator over $\mathcal{W}_0$ (see (2.5)). Here, we have an additional desire that is not present for matrices: we would like an efficient algorithm to compute $\Pi_{\mathcal{W}_0}\psi$ for any $\psi\in L^2(\Omega)$, where $\Pi_{\mathcal{W}_0} : L^2(\Omega)\to\mathcal{W}_0$ is an orthogonal projection operator (see section 3). In this paper, we always select $\mathcal{R}$ to be the indefinite integral operator in (2.3).

2.4. Convergence theory for the preconditioned CG method. In this section, we show that the preconditioned CG method converges at a geometric rate when the operator preconditioner is bounded, is a smoother, and is a preconditioner for the Laplacian (see subsection 2.2). The standard bound on the convergence of the CG method for $Ax=b$ involves the condition number of A [23, Chap. 2]. Though this bound is not always descriptive, it is explicit and is the first canonical convergence result. Similarly, the convergence of our operator CG method can be bounded using the condition number of the operator $\mathcal{R}^*\mathcal{L}\mathcal{R}$ from a restricted subspace of $L^2(\Omega)$.

The condition number of $\mathcal{R}^*\mathcal{L}\mathcal{R} : \mathcal{W}_0 \to L^2(\Omega)$ is given by [17]:

$$\kappa_{\mathcal{W}_0}(\mathcal{R}^*\mathcal{L}\mathcal{R}) = \frac{\sup_{\phi\in\mathcal{W}_0,\|\phi\|=1}\mathcal{B}[\mathcal{R}\phi,\mathcal{R}\phi]}{\inf_{\phi\in\mathcal{W}_0,\|\phi\|=1}\mathcal{B}[\mathcal{R}\phi,\mathcal{R}\phi]}, \tag{2.5}$$

where $\mathcal{W}_0 = \{\phi\in L^2(\Omega) : \mathcal{R}\phi \in \mathcal{H}_0^1(\Omega)\}$. The following theorem bounds $\kappa_{\mathcal{W}_0}(\mathcal{R}^*\mathcal{L}\mathcal{R})$ and is used in Corollary 2.8 to derive a CG convergence bound.

THEOREM 2.6. *Let $\Omega = (-1,1)$, $a, c \in L^\infty(\Omega)$, $a(x) > 0$ for $x\in\Omega$, $c(x)\geq 0$ for $x\in\Omega$, and $\mathcal{L}u = -(a(x)u'(x))' + c(x)u$ with bilinear form $\mathcal{B} : \mathcal{H}_0^1(\Omega)\times\mathcal{H}_0^1(\Omega)\to\mathbb{R}$. Given an operator preconditioner $\mathcal{R}$ that is bounded, is a smoother, and is a preconditioner for the Laplacian (see subsection 2.2), the (restricted) condition number of $\mathcal{R}^*\mathcal{L}\mathcal{R}$ is bounded. Furthermore,*

$$\kappa_{\mathcal{W}_0}(\mathcal{R}^*\mathcal{L}\mathcal{R}) \leq \frac{\gamma_1\|a\|_\infty + \|c\|_\infty\|\mathcal{R}\|_{op}^2}{\gamma_0\inf_{x\in\Omega}|a(x)|},$$

where $\mathcal{W}_0 = \{\phi\in L^2(\Omega) : \mathcal{R}\phi\in\mathcal{H}_0^1(\Omega)\}$ and $\|\mathcal{R}\|_{op} = \sup_{\phi\in\mathcal{W}_0,\|\phi\|=1}\|\mathcal{R}\phi\|$.

Proof. If $\phi\in\mathcal{W}_0$, then $\mathcal{R}\phi\in\mathcal{H}_0^1(\Omega)$, and we have

$$\mathcal{B}[\mathcal{R}\phi,\mathcal{R}\phi] = \int_{-1}^1 a(x)(\mathcal{R}\phi)'(x)(\mathcal{R}\phi)'(x)dx + \int_{-1}^1 c(x)(\mathcal{R}\phi(x))^2 dx. \tag{2.6}$$

The first term in (2.6) can be bounded as follows:

$$\int_{-1}^{1} a(x)(\mathcal{R}\phi)'(x)(\mathcal{R}\phi)'(x)dx \leq \|a\|_\infty \|(\mathcal{R}\phi)'\|^2 \leq \gamma_1 \|a\|_\infty \|\phi\|^2,$$

where the last inequality uses the fact that $\mathcal{R}$ is a preconditioner for the Laplacian (see subsection 2.2). We also find that $\int_{-1}^{1} a(x)(\mathcal{R}\phi)'(x)(\mathcal{R}\phi)'(x)dx \geq \gamma_0 \inf_{x\in\Omega} |a(x)| \|\phi\|^2$. For the second term in (2.6), we simply have

$$0 \leq \int_{-1}^{1} c(x)(\mathcal{R}\phi(x))^2 dx \leq \|c\|_\infty \|\mathcal{R}\phi\|^2 \leq \|c\|_\infty \|\mathcal{R}\|_{\text{op}}^2 \|\phi\|^2.$$

The bound on $\kappa_{\mathcal{W}_0}(\mathcal{R}^*\mathcal{L}\mathcal{R})$ immediately follows. ∎

Similar statements to Theorem 2.6 appear in the literature on operator preconditioners for Galerkin discretizations [16, 17]. Theorem 2.6 has a slightly different flavor because $\mathcal{R}$ and $\mathcal{L}$ are operators.

In (2.4), the Krylov space is based on the operator $\mathcal{T} = \Pi_{\mathcal{W}_0}^* \mathcal{R}^*\mathcal{L}\mathcal{R}\Pi_{\mathcal{W}_0}$ and the (restricted) condition number of $\mathcal{T}$ immediately follows from Theorem 2.6.

Corollary 2.7. *With the same assumptions as Theorem* 2.6*, we have*

$$\kappa_{\mathcal{W}_0}(\mathcal{T}) = \kappa_{\mathcal{W}_0}(\mathcal{R}^*\mathcal{L}\mathcal{R}), \qquad \mathcal{T} = \Pi_{\mathcal{W}_0}^* \mathcal{R}^*\mathcal{L}\mathcal{R}\Pi_{\mathcal{W}_0},$$

where $\Pi_{\mathcal{W}_0} : L^2(\Omega) \to \mathcal{W}_0$ *is the orthogonal projection operator onto* $\mathcal{W}_0$.

The bound on $\kappa_{\mathcal{W}_0}(\mathcal{T})$ allows us to prove that $\|u - u_k\|_{\mathcal{L}}$ geometrically decays to zero as $k \to \infty$.

Corollary 2.8. *With the same assumptions as Theorem* 2.6*, let* $u_0 = 0, u_1, \ldots,$ *be the CG iterates from Algorithm* 2.3*. Then,*

$$\|u - u_k\|_{\mathcal{L}} \leq 2\left(\frac{\sqrt{\kappa_{\mathcal{W}_0}(\mathcal{T})} - 1}{\sqrt{\kappa_{\mathcal{W}_0}(\mathcal{T})} + 1}\right)^k \|u\|_{\mathcal{L}},, \qquad k \geq 0, \tag{2.7}$$

where $\mathcal{T} = \Pi_{\mathcal{W}_0}^* \mathcal{R}^*\mathcal{L}\mathcal{R}\Pi_{\mathcal{W}_0}$ *and* u *is the exact solution to* (1.1).

Proof. Corollary 2.7 shows that $\kappa_{\mathcal{W}_0}(\mathcal{T})$ is bounded. By copying the proof of the convergence bound for the CG method for matrices [23], we find that the iterates $v_0 = 0, v_1, v_2, \ldots,$ satisfy

$$\|v - v_k\|_{\mathcal{T}} \leq 2\left(\frac{\sqrt{\kappa_{\mathcal{W}_0}(\mathcal{T})} - 1}{\sqrt{\kappa_{\mathcal{W}_0}(\mathcal{T})} + 1}\right)^k \|v\|_{\mathcal{T}}, \qquad k \geq 0,$$

where $u = \mathcal{R}v$. The result follows since $\|v\|_{\mathcal{T}}^2 = \mathcal{B}[\mathcal{R}v, \mathcal{R}v] = \|\mathcal{R}v\|_{\mathcal{L}}^2$, and $u_k = \mathcal{R}v_k$. ∎

Corollary 2.8 implies that the preconditioned CG method in Algorithm 2.3 constructs iterates $u_0 = 0, u_1, u_2, \ldots,$ that converge geometrically to u in the $\|\cdot\|_{\mathcal{L}}$ norm. In other words, for an accuracy goal of $0 < \epsilon < 1$ we require

$$k \geq \left\lceil \frac{\log(2/\epsilon)}{\log\left(\sqrt{\kappa_{\mathcal{W}_0}(\mathcal{T})} + 1\right) - \log\left(\sqrt{\kappa_{\mathcal{W}_0}(\mathcal{T})} - 1\right)} \right\rceil$$

iterations to guarantee that $\|u-u_k\|_{\mathcal{L}} \leq \epsilon\|u\|_{\mathcal{L}}$. Here, $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. Since $\kappa_{\mathcal{W}_0}(\mathcal{T})$ is bounded, the preconditioned CG method can be terminated after a finite number of iterations.

Figure 3 shows the convergence of the preconditioned CG method compared to the error bound in (2.7) when solving three BVPs using the indefinite integration preconditioner $\mathcal{R}v = \int_{-1}^{x} v(s)ds$. The convergence behavior of the preconditioned CG method comes with theoretical guarantees and is a vast improvement over the convergence of the unpreconditioned CG method (see Figure 2 (right)).

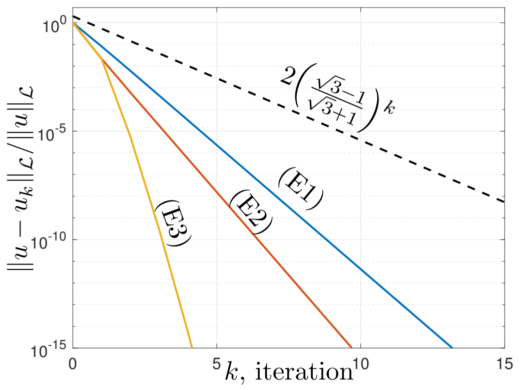

FIG. 3. *Convergence of the preconditioned CG method for three BVPs with zero Dirichlet boundary conditions.* (E1): $-((2+\cos(\pi x))u')' = f$ *(blue line),* (E2): $-((1+x^2)u')' + (\frac{\pi}{4}\cos(\pi x))^2 u = f$ *(red line), and* (E3): $-u'' + 2(\frac{\pi}{4})^2 u = f$ *(yellow line) with* $f = (1+x^2)^{-1}$. *Corollary* 2.8 *gives the same bound for these three examples (black dashed line). Note that* $\mathcal{R}^* f \notin \mathcal{W}_0$ *so an ancillary problem is solved before applying the CG method for these three BVPs (see subsection* 2.5*).*

2.5. General right-hand sides. Here, we remove the assumption that $\mathcal{R}^* f \in \mathcal{W}_0$ by solving an ancillary problem that converts f into a right-hand side that is amenable to our operator CG method.[7]

Write the solution to (2.2) as $v = v_1 + v_2$, where v_2 is any solution from $\mathcal{W}_0$ that solves the following ancillary problem:

$$\big[\mathcal{R}\mathcal{R}^*\mathcal{L}\mathcal{R}v_2\big](\pm 1) = \big[\mathcal{R}\mathcal{R}^* f\big](\pm 1). \tag{2.8}$$

The remaining part of the solution, i.e., v_1, then satisfies

$$\mathcal{R}^*\mathcal{L}\mathcal{R}v_1 = \mathcal{R}^* g, \qquad g = (f - \mathcal{R}^*\mathcal{L}\mathcal{R}v_2).$$

By construction, $\big[\mathcal{R}\mathcal{R}^* g\big](\pm 1) = 0$ and since $g \in L^2(\Omega)$, we find that $\mathcal{R}^* g \in \mathcal{W}_0$. Therefore, we can solve

$$\mathcal{R}^*\mathcal{L}\mathcal{R}v_1 = \mathcal{R}^* g$$

via the preconditioned CG method in Algorithm 2.3.

When $\mathcal{R}$ is the indefinite integral preconditioner in (2.3), the condition at -1 in (2.8) is trivially satisfied and the ancillary problem reduces to solving

(2.9)
$$\int_{-1}^{1} a(s)v_2(s)ds + \int_{-1}^{1}\left(\int_{s}^{1} c(t)(t+1)dt\right) v_2(s)ds = \int_{-1}^{1}(s+1)f(s)ds, \quad v_2 \in \mathcal{W}_0.$$

[7]In the standard Galerkin framework, one seeks to find a solution to (1.1) via the weak formulation $\mathcal{B}[u,\psi] = \langle f, \psi\rangle$ for all $\psi \in \mathcal{H}_0^1(\Omega)$, even when $f \notin \mathcal{H}_0^1(\Omega)$. This is theoretically justified because $\mathcal{H}_0^1(\Omega)$ is a dense subspace of $L^2(\Omega)$. In our setup, the test space $\mathcal{W}_0$ is not a dense subset of $L^2(\Omega)$ so solving an ancillary problem is necessary.

This problem can be solved efficiently by picking any $w \in \mathcal{W}_0$ such that the lefthand side of (2.9), with v_2 replaced by w, is a scalar $\eta \neq 0$ and setting $v_2 = (\frac{1}{\eta}\int_{-1}^{1}(s+1)f(s)ds)w$. Usually, $w(s) = s$ is an adequate choice.

3. Practical realizations of the operator CG method.

We now describe two realizations of the theoretical framework in section 2 for solving (1.1). While the theory in section 2 works for the solution space $\mathcal{H}_0^1(\Omega)$, in practice, we usually first define a dense subspace $\mathcal{V}$ of $\mathcal{H}^1(\Omega)$ and associated subspace $\mathcal{W} = \{w \in L^2(\Omega) : \mathcal{R}w \in \mathcal{V}\}$ on which the operations performed by the CG method can be efficiently computed. Provided that the operations performed by the CG method map functions from $\mathcal{W}$ to $\mathcal{W}$ and the right-hand side of (1.1) and its variable coefficients are in $\mathcal{W}$, the preconditioned CG method in section 2 is unaware of the subspace $\mathcal{V}$. In this section, we consider (1) $\mathcal{V}$ being the space of analytic functions and (2) $\mathcal{V}$ being the space of continuous piecewise analytic functions (with a finite number of fixed breakpoints). In these two cases the approximation space for the solution to (1.1) is $\mathcal{V}_0 = \{\phi \in \mathcal{V} : \phi(\pm 1) = 0\} \subset \mathcal{H}_0^1(\Omega)$.

We have implemented (1) and (2) in Chebfun [7] in the `pcg` command, which follows the syntax of the standard MATLAB `pcg` command for matrices. Fortunately, object-oriented programming allows us to only have one implementation of the operator CG method for (1) and (2) as Chebfun automatically calls the appropriate underlying algorithms to compute inner-products, integrals, and derivatives via operator overloading. This is one of the advantages of developing a Krylov-based solver that works independently from the underlying discretization of the solution and right-hand side. Unlike most BVP solvers, our Krylov-based solvers have no fixed discretization. Instead, we let Chebfun automatically resolve the functions that appear during the operator CG method to machine precision [1]. A summary of the main operations that the preconditioned CG method requires is given in Table 1, along with the corresponding Chebfun commands.

3.1. Analytic functions.

Let $\mathcal{V}$ be the space of functions that are analytic in an open neighborhood of $[-1, 1]$ and consider the preconditioner $\mathcal{R}\phi = \int_{-1}^{x}\phi(s)ds$. Note that the associated space $\mathcal{W}$ is closed under indefinite integration, differentiation, and function product, and that $\mathcal{R}$ is bounded, is a smoother, and preconditions the Laplacian. The choice of $\mathcal{V}$ and $\mathcal{R}$ completely determine a realization of the preconditioned CG method with the approximation space for the solution $\mathcal{V}_0 = \{\phi \in \mathcal{V} : \phi(\pm 1) = 0\}$. Here, we are implicitly assuming that the variable coefficients in (1.1) are analytic functions or have been approximated by analytic functions.

TABLE 1
Summary of the main operations that are required by the preconditioned CG method. The Chebfun commands that execute these mathematical operations are also given. Objected-oriented programming and operator overloading allows the same Chebfun command to employ different underlying algorithms depending on whether `p` *and* `q` *are analytic or piecewise analytic.*

Operation	Mathematical operation	Chebfun command
Preconditioner	$\int_{-1}^{x} p(s)ds$, $\int_{x}^{1} p(s)ds$	`cumsum(p)`, `sum(p)-cumsum(p)`
Differentiation	$p'(x)$	`diff(p)`
Product	$p(x)q(x)$	`p*q`
Inner-product	$\int_{-1}^{1} p(s)q(s)ds$	`p'*q`
Projector	$p - \frac{1}{2}\int_{-1}^{1} p(s)ds$	`p - mean(p)`

In order to implement an efficient practical algorithm, we approximate analytic functions to within machine precision ϵ_{mach} by Chebyshev expansions. That is, for some integer $n \geq 0$ that is adaptively determined [1], we approximate an analytic function $\phi \in \mathcal{V}$ by

$$\phi(x) \approx p(x) = \sum_{k=0}^{n} \alpha_k T_k(x), \qquad \|\phi - p\|_\infty < \epsilon_{\text{mach}} \|\phi\|_\infty, \tag{3.1}$$

where $T_k(x)$ is the degree k Chebyshev polynomial and $\|\cdot\|_\infty$ is the absolute maximum norm on $[-1, 1]$. If p is the Chebyshev interpolant of an analytic function ϕ, then the Chebyshev expansion coefficients in (3.1) converge geometrically to zero [38, Chap. 8]. Moreover, the expansion coefficients $\{\alpha_k\}$ in (3.1) can be computed in $\mathcal{O}(n \log n)$ via the discrete Chebyshev transform [11]. To automatically resolve a function $\phi \in \mathcal{V}$ to machine precision, we call the Chebfun command `p = chebfun(phi)`.

There are a number of operations that the CG method must perform on the adaptively determined Chebyshev expansions.

Applying the preconditioner and its adjoint. For $p(x) = \sum_{k=0}^{n} \alpha_k T_k(x)$, we need to compute $\mathcal{R}p = \int_{-1}^{x} p(s)ds$. The Chebyshev expansion coefficients for $\mathcal{R}p$ can be computed by using a simple recurrence relation [22, sect. 8.1], costing $\mathcal{O}(n)$ operations. This is implemented in the Chebfun command `cumsum(p)`. Similarly, $\mathcal{R}^*p$ can be computed with the Chebfun command `sum(p)-cumsum(p)` in $\mathcal{O}(n)$ operations.

Applying the differential operator. For $p(x) = \sum_{k=0}^{n} \alpha_k T_k(x)$, we need to compute $\mathcal{L}p$. If $\mathcal{L}p = -(a(x)p'(x))' + c(x)p(x)$ and $a(x)$ and $c(x)$ are analytic functions and represented by adaptively determined Chebyshev expansions, then we can compute $\mathcal{L}$ via the Chebun commands `Lp = -diff(a*diff(p))+c*p`. Computing the Chebyshev expansions of $p'(x)$ can be computed in $\mathcal{O}(n)$ operations via a recurrence relation [22, p. 34], and the coefficients for $a(x)p(x)$ can be computed in $\mathcal{O}(N \log N)$ operations with a discrete Chebyshev transform [11]. Here, N is the maximum polynomial degree required to resolve a and p.

Inner-products. Given $p(x) = \sum_{k=0}^{n} \alpha_k T_k(x)$ and $q(x) = \sum_{k=0}^{n} \beta_k T_k(x)$, we need to be able to compute

$$\langle p, q \rangle = \int_{-1}^{1} p(s)q(s)ds.$$

We compute this by Clenshaw–Curtis quadrature [38, Chap. 19], costing $\mathcal{O}(n \log n)$ operations. The integral is computed by the Chebfun command `p'*q`.

Applying the projection operator. For $p(x) = \sum_{k=0}^{n} \alpha_k T_k(x)$, we need to compute the projection

$$\Pi_{\mathcal{W}_0} p = p - \frac{1}{2} \int_{-1}^{1} p(s)ds.$$

This can be achieved in $\mathcal{O}(n \log n)$ operations by using Clenshaw–Curtis quadrature for definite integration [38, Chap. 19]. The projection operator is computed by Chebfun with the command `p-mean(p)`.

Since this realization of the preconditioned CG method employs adaptively selected polynomials to resolve the solution of (1.1), we compare our preconditioned CG method against adaptive implementations of the spectral collocation method[8]

[8]More precisely, we compare against rectangular spectral collocation [6], which performs a projection of the range of the matrices to automatically deal with boundary conditions of BVPs. Rectangular spectral collocation is employed by default in the `chebop` class of Chebfun [5].

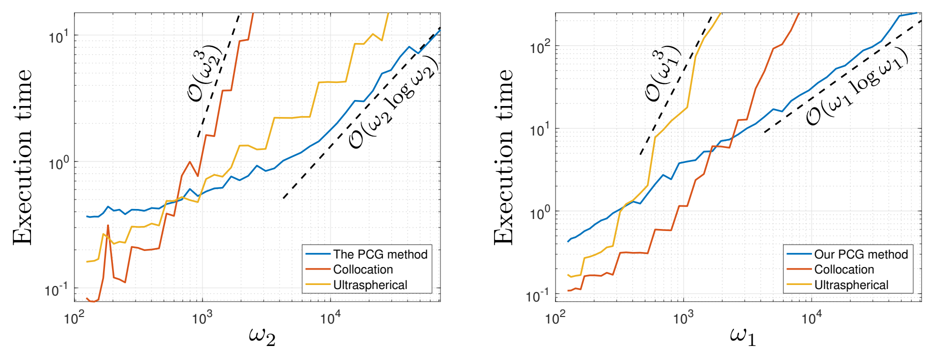

FIG. 4. *Comparison of execution timings of our preconditioned CG method (blue line), spectral collocation (red line), and the ultraspherical spectral method (yellow line) for the BVP* $-((2+\cos(\omega_1\pi x))u')' = f(x)$, $u(\pm 1) = 0$, *where* f *is chosen so that* $u(x) = \sin(\omega_2 \pi x)$ *is the solution. All spectral methods are implemented in an adaptive manner to automatically resolve the BVP solution to essentially machine precision. The spectral collocation method and the ultraspherical spectral method discretizes the BVP and then solves the resulting linear system. Left: The parameter* ω_2 *is increased while* $\omega_1 = 10$, *which defines a family of BVPs for which the solution requires a high polynomial degree to resolve to machine precision. Right: The parameter* ω_1 *is increased while* $\omega_2 = 10$, *which defines a family of BVPs for which the variable coefficients require a high polynomial degree to resolve to machine precision. The polynomial degree required to resolve* $\cos(\omega_1\pi x)$ *and* $\sin(\omega_2\pi x)$ *on* $[-1,1]$ *to machine precision is* $\mathcal{O}(\omega_1)$ *and* $\mathcal{O}(\omega_2)$, *respectively.*

and the ultraspherical discretization [25]. Both these adaptive spectral methods are implemented in Chebfun.

To do the comparison, we consider the family of BVPs parametrized by ω_1 and ω_2 such that

$$-((2+\cos(\omega_1\pi x))u'(x))' = f(x) \text{ on } \Omega = (-1,1), \qquad u(\pm 1) = 0,$$

where the right-hand side $f(x)$ is chosen so that $u(x) = \sin(\omega_2\pi x)$ is the exact solution. We investigate two regimes: (a) ω_1 fixed, $\omega_2 \to \infty$ and (b) ω_2 fixed, $\omega_1 \to \infty$. In the first regime, a high degree polynomial is required to resolve the solution to machine precision while the variable coefficients of the BVP can be resolved by a low degree polynomial. This is a setting in which the ultraspherical spectral method is competitive with the preconditioned CG method (see Figure 4 (left)). In the second regime, the variable coefficients of the BVP require high degree polynomials to resolve, leading to dense spectral discretization matrices for both spectral collocation and the ultraspherical spectral method. In this setting, we find that it is computationally beneficial to employ our preconditioned CG method.

From these experiments and others, we learn that the preconditioned CG method is computationally beneficial compared to standard spectral methods employing direct solvers when spectral methods generate linear systems that are large and dense. A similar comparison can be made between direct and iterative solvers for linear systems.

3.2. Continuous functions that are piecewise analytic. Let $\mathcal{V} \subset \mathcal{H}^1(\Omega)$ be the space of continuous functions that are piecewise analytic with a finite number of fixed breakpoints $-1 = x_0 < x_1 < \cdots < x_{M+1} = 1$. That is, the space of continuous functions ϕ such that $\phi|_{[x_i,x_{i+1}]}$ is analytic in a neighborhood of $[x_i, x_{i+1}]$ for $0 \leq i \leq M$. Again, we take the preconditioner to be $\mathcal{R}\phi = \int_{-1}^{x} \phi(s)ds$. The induced space $\mathcal{W} = \{v \in L^2(\Omega) : \mathcal{R}v \in \mathcal{V}\}$ does not have a continuity requirement. The approximation space $\mathcal{W}$ is closed under indefinite integration and multiplication and weak differentiation. This implies that all the functions that appear in the preconditioned CG method are in $\mathcal{W}$.

Given a function that is piecewise analytic, we represent it by subdividing the interval $[-1, 1]$ into $M + 1$ subintervals, i.e., $[-1, x_1] \cup [x_1, x_2] \cup \cdots \cup [x_M, 1]$, and representing the function by a Chebyshev expansion on each subinterval [27]. The Chebfun command that automatically determines the breakpoint locations and the polynomial degree to use on each subinterval is `p=chebfun(phi,'splitting','on')`. Any function that is computed during the CG method is automatically resolved in a piecewise fashion by Chebfun.

To solve for a piecewise smooth solution using spectral collocation or the ultraspherical spectral method, one has to construct a matrix that imposes the BVP operator on each subinterval along with continuity conditions at x_i for $1 \le i \le M$ [6]. In our preconditioned CG method the iterates v_k belong to $\mathcal{W}_0$, which is a space that contains functions that are not continuous. However, continuity on the approximate solutions $u_k = \mathcal{R} v_k$ is implicitly imposed because $\mathcal{R}$ acts as a smoother.

The algorithms to compute the tasks of applying the preconditioner, the differential operator, and the projection operator are almost immediate from the algorithms in subsection 3.1. For example, if $\phi \in \mathcal{V}$ and $x \in [x_m, x_{m+1}]$ for some $0 \le m \le M$, then

$$\mathcal{R}\phi = \int_{-1}^{x} \phi(s)ds = \sum_{i=0}^{m-1} \int_{x_i}^{x_{i+1}} \phi(s)ds + \int_{x_m}^{x} \phi(s)ds.$$

Therefore, to calculate the piecewise analytic function of $\mathcal{R}\phi$ on $[x_m, x_{m+1}]$ one performs indefinite integration on $[x_m, x_{m+1}]$ using a recurrence relation [22, sect. 8.1] and adds to that the constant $\int_{-1}^{x_m} v(s)ds$ computed by applying Clenshaw–Curtis quadrature to each subinterval [38, Chap. 19].

Figure 5 demonstrates the preconditioned CG method on three BVPs with piecewise smooth variable coefficients. The solutions of which have the same breakpoints as the variable coefficients. Since Chebfun automatically determines breakpoint locations for piecewise smooth functions [27], our BVP solver automatically inherits this adaptivity. For piecewise continuous solutions we execute the same `pcg` command as in subsection 3.1 without modification. As can be seen from the convergence theory in section 2 and Figure 5 (right), the convergence rate of the CG method is independent of the smoothness of the solution.

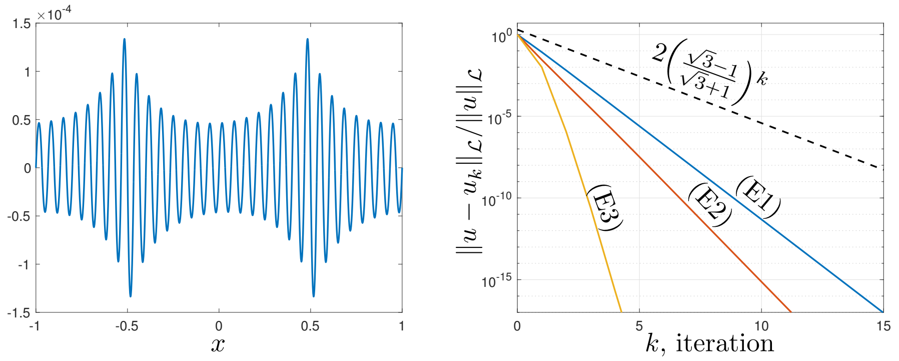

FIG. 5. *The preconditioned CG method for continuous functions that are piecewise analytic. Left: Solution to* $-((1 + 2|\cos(\pi x)|)u')' = \operatorname{sign}(\cos(30\pi x))$ *with* $u(\pm 1) = 0$. *Right: The convergence of the preconditioned CG method for three BVPS with zero Dirichlet boundary conditions.* (E1): $-((1 + 2|\cos(\pi x)|)u')' = f$ *(blue line),* (E2): $-((1 + |\sin(\pi x^2)|)u')' + (\frac{\pi}{4})^2|\cos(2\pi x)|u = f$ *(red line), and* (E3): $-u'' + 2(\frac{\pi}{4})^2|\cos(20\pi x)|u = f$ *(yellow line), where* $f = (1 + x^2)^{-1}$. *Corollary* 2.8 *gives the same convergence bound for these three BVPs (black dashed line).*

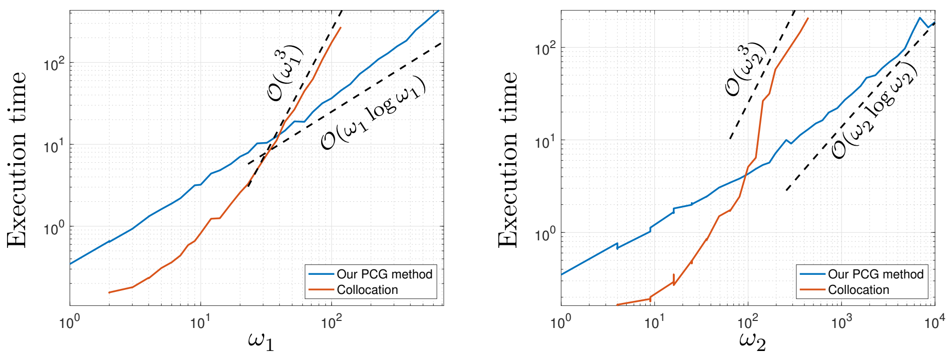

FIG. 6. *Comparison of execution timings of our preconditioned PCG method (blue line) and spectral collocation (red line) for the BVP* $-((2+\cos(\omega_2\pi x))u'(x))'+(2+|\cos(\omega_1\pi x^2)|)u(x) = f(x)$, $u(\pm1) = 0$, *where* f *is chosen so that* $u(x) = \sin(10\pi x)$ *is the solution. All spectral methods are implemented in an adaptive manner to automatically resolve the BVP solution to essentially machine precision. Spectral collocation discretizes the BVP and then solves the resulting linear system. Left: The parameter* ω_1 *is increased and* $\omega_2 = 10\omega_1$, *which defines a family of BVPs for which the variable coefficients requires a high number of subintervals but a low polynomial degree on each subinterval to resolve to machine accuracy. Right: The parameter* ω_1 *is increased while* $\omega_2 = \omega_1^2$, *which defines a family of BVPs for which the variable coefficients require a high number of subintervals and a high polynomial degree on each subinterval to resolve the variable coefficients to machine accuracy. The number of subintervals required to resolve* $|\cos(\omega_1\pi x^2)|$ *to machine precision is* $\mathcal{O}(\omega_1)$.

In Figure 6 we demonstrate the scaling of the PCG method on the family of BVPs with nonsmooth coefficients:

$$-((2+\cos(\omega_2\pi x))u'(x))'+(2+|\cos(\omega_1\pi x^2)|)u(x) = f(x) \text{ on } \Omega = (-1,1), \quad u(\pm1) = 0,$$

where $f(x)$ is chosen so that $u(x) = \sin(10\pi x)$ is the exact solution. We investigate two regimes: (1) $\omega_1 = \omega_2$, and (2) $\omega_2 = \omega_1^2$. In the first regime, the number of intervals increases but the degree of the polynomial on each subinterval stays roughly constant. In the second regime, both the number of intervals and the polynomial degree required to resolve the coefficients on each subinterval increases.

4. Other Krylov-based methods.

The preconditioned CG method in section 2 has provided us with an operator analogue of a Krylov subspace for solving (1.1) (see (2.4)). Two additional Krylov subspace methods for solving $Ax = b$ are MINRES (for symmetric linear systems) [28] and GMRES (for general linear systems) [31]. In the matrix setting, MINRES and GMRES generate iterates by computing the best solution to $Ax = b$ from a Krylov subspace, as measured by the Euclidean norm of the residual, i.e.,

$$x_k = \operatorname*{arg\,min}_{y\in\mathcal{K}_k(A,b)} \|b - Ay\|_2, \qquad \mathcal{K}_k(A,b) = \operatorname{Span}\left\{b, Ab, \ldots, A^{k-1}b\right\}, \tag{4.1}$$

where $\|\cdot\|_2$ denotes the Euclidean norm of a vector.

Motivated by (4.1), we set out to derive a MINRES and GMRES method for solving (1.1) that constructs iterates so that

$$v_k = \operatorname*{arg\,min}_{p\in\mathcal{K}_k(\mathcal{T},\mathcal{R}^*f)} \|\mathcal{R}^*f - \mathcal{T}p\|, \tag{4.2}$$

where $\mathcal{R}$ is given in (2.3), and $\mathcal{T}$ and $\mathcal{K}_k(\mathcal{T},\mathcal{R}^*f)$ are given in (2.4). The hope is that the iterates $u_k = \mathcal{R}v_k$ converge to the solution u of (1.1). In (4.2), we assume that $\mathcal{R}^*f \in \mathcal{W}_0 = \{v \in L^2(\Omega) : \mathcal{R}v \in \mathcal{H}_0^1(\Omega)\}$; otherwise, the ancillary problem in subsection 2.5 is used to modify the right-hand side. In the case where $\mathcal{L}$ is self-adjoint with positive eigenvalues, a convergence bound analogous to (2.7) holds for GMRES and MINRES. However, even though the preconditioned operator $\mathcal{R}^*\mathcal{L}\mathcal{R}$ is always bounded for the choice of the integration preconditioner, little can be said about the convergence of the methods in a general case.

4.1. The GMRES method for differential operators. The kth step of the GMRES method for solving $Ax = b$ computes an orthogonal basis for $\mathcal{K}_k(A, b)$ and then solves the least squares problem in (4.1) for x_k. Analogously, our operator GMRES method computes an orthogonal basis for the Krylov subspace $\mathcal{K}_k(\mathcal{T}, \mathcal{R}^*f)$. The orthogonal basis is computed via the decomposition

$$\mathcal{T}\mathcal{Q}_k = \mathcal{Q}_{k+1}\tilde{H}_k, \tag{4.3}$$

where $\tilde{H}_k$ is a $(k+1) \times k$ upper Hessenberg matrix and $\mathcal{Q}_k$ is a quasimatrix with k orthonormal columns.[9] The decomposition is computed by an Arnoldi iteration on functions in $L^2(\Omega)$ using modified Gram–Schmidt (see Algorithm 4.1).

Algorithm 4.1 Arnoldi iteration. Here, $\mathcal{T}$ is the operator in (2.4) and $\mathcal{R}^*f \in \mathcal{W}_0$.

```
1: $q_1 = \mathcal{R}^*f/\|\mathcal{R}^*f\|$
2: for $k = 2, \ldots, m$ do
3:     $q_k = \mathcal{T}q_{k-1}$
4:     for $j = 1, \ldots, k-1$ do
5:         $h_{j,k-1} = \langle q_j, q_k\rangle$, $q_k = q_k - h_{j,k-1}q_j$
6:     end for
7:     $h_{k,k-1} = \|q_k\|$, $q_k = q_k/h_{k,k-1}$
8: end for
```

Once an orthogonal basis for $\mathcal{K}_k(\mathcal{T}, \mathcal{R}^*f)$ is computed by Algorithm 4.1, the iterates from (4.2) can be computed as follows:

$$\operatorname*{arg\,min}_{p\in\mathcal{K}_k(\mathcal{T},\mathcal{R}^*f)} \|\mathcal{R}^*f - \mathcal{T}p\| = \operatorname*{arg\,min}_{y\in\mathbb{R}^k} \|\mathcal{R}^*f - \mathcal{T}\mathcal{Q}_k y\| = \operatorname*{arg\,min}_{y\in\mathbb{R}^k} \left\| \|\mathcal{R}^*f\| e_1 - \tilde{H}_k y \right\|_2 ,$$

which is a standard least squares problem that is typically solved by updating a QR factorization of $\tilde{H}_k$ at each iteration using Givens rotations [40]. We derive the following operator GMRES method for (1.1).

Unlike CG, the computational and storage costs of GMRES grows with the number of iterations. To avoid excessive storage costs, the GMRES method is usually restarted after m iterations for some integer m, i.e., v_m becomes an initial guess for a new GMRES method. The convergence behavior of the GMRES method is difficult to fully characterize and the statements that can be presented for convergence are analogous to those for the matrix GMRES method [40, Chap. 6]. Figure 7 (left) shows the convergence of the preconditioned GMRES on the BVP

$$-(e^x u')' + u' - 10u = \sin(30\pi x), \quad u(\pm 1) = 0$$

for different restarts. As observed in the matrix case the convergence can deteriorate with too frequent restarts, though iterates after restarting are computed more efficiently.

[9]A quasimatrix is a matrix whose columns are functions [35]. The quasimatrix has orthonormal columns if the columns are orthonormal with respect to the L^2 inner-product.

Algorithm 4.2 The preconditioned GMRES method for (1.1), where $\mathcal{T}$ is the operator in (2.4), $\mathcal{R}^* f \in \mathcal{W}_0$ and $0 < \epsilon < 1$ is a tolerance on the norm of the residual.

```
1:  for $k = 1, 2, \ldots,$ do
2:      Compute $\mathcal{Q}_{k+1}$ and $\tilde{H}_k$ in (4.3) using one step of Algorithm 4.1
3:      Compute the QR factorization of $\tilde{H}_k$
4:      Solve $\rho = \min_y \left\| \|\mathcal{R}^* f\| e_1 - \tilde{H}_k y \right\|$
5:      if $\rho < \epsilon$ then
6:          $v = \mathcal{Q}_k y$
7:          $u = \mathcal{R} v$
8:          stop iteration
9:      end if
10: end for
```

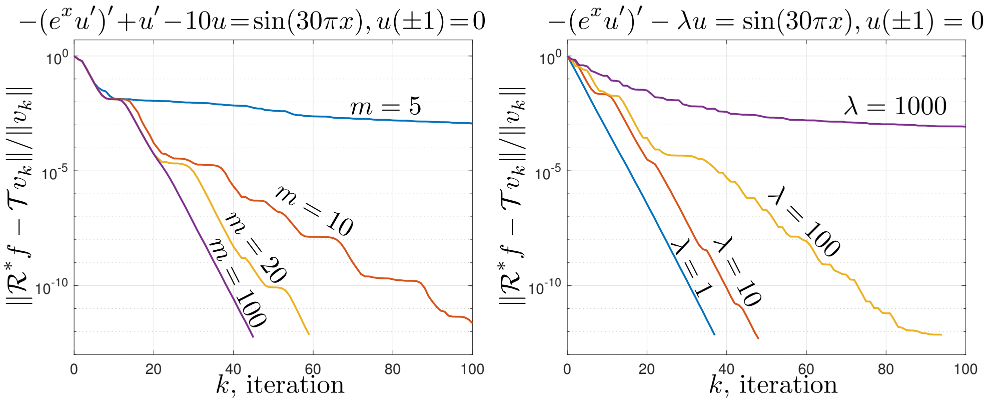

FIG. 7. *Left: Convergence of the restarted GMRES method with restarts every m iterations for $m = 5$ (blue line), $m = 10$ (red line), $m = 20$ (yellow line), and $m = 100$ (purple line). Right: Convergence of the MINRES method for $-(e^x u')' - \lambda u = \sin(30\pi x)$ with $u(\pm 1) = 0$ with $\lambda = 1$ (blue line), $\lambda = 10$ (red line), $\lambda = 100$ (yellow line), and $\lambda = 1000$ (purple line). The quality of the indefinite integral preconditioner in* (2.3) *is reduced as λ increases.*

To study the scaling of the GMRES algorithm against other adaptive spectral methods, we consider the family of BVPs parametrized by ω_1 and ω_2 such that

$$-((2 + \cos(\omega_1 \pi x))u')' + (2 + \sin(\omega_1 \pi x))u' + u = f(x) \text{ on } \Omega = (-1, 1), \qquad u(\pm 1) = 0,$$

where the right-hand side $f(x)$ is chosen so that $u(x) = \sin(\omega_2 \pi x)$ is the exact solution. Similarly to subsection 3.1, we investigate two regimes: (a) ω_1 fixed, $\omega_2 \to \infty$ and (b) ω_2 fixed, $\omega_1 \to \infty$. In the first regime, a high degree polynomial is required to resolve the solution to machine precision while the variable coefficients of the BVP can be resolved by a low degree polynomial. Figure 8 shows the result of the scaling study. We conclude that the GMRES method displays a similar scaling to the CG method and thus GMRES is also particularly beneficial over typical spectral methods when the variable coefficients of the BVP require high degree polynomials to resolve.

The operator GMRES method is implemented in Chebfun in the `gmres` command and has precisely the same realizations as the operator CG method (see section 3).

4.2. The MINRES method for differential operators. MINRES can be described as a special case of GMRES that applies when the linear system is symmetric.

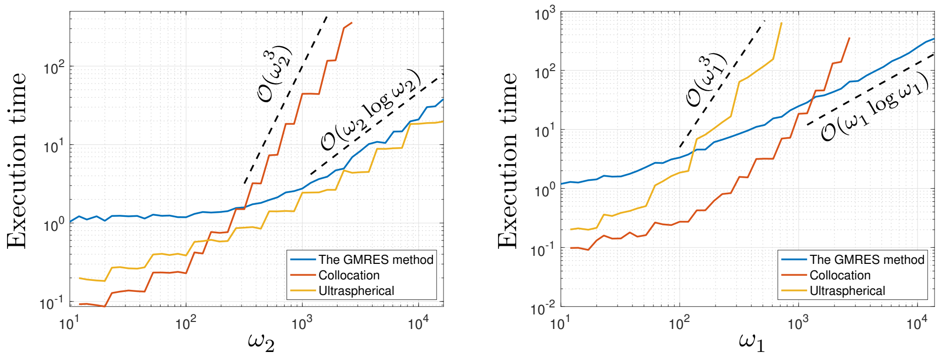

FIG. 8. *Comparison of execution timings of our preconditioned unrestarted GMRES method (blue line), spectral collocation (red line), and the ultraspherical spectral method (yellow line) for the BVP* $-((2+\cos(\omega_1 \pi x))u')' + (2+\sin(\omega_1 \pi x))u' + u = f(x)$, $u(\pm 1) = 0$, *where* f *is chosen so that* $u(x) = \sin(\omega_2 \pi x)$ *is the solution. All spectral methods are implemented in an adaptive manner to automatically resolve the BVP solution to essentially machine precision. Spectral collocation and the ultraspherical spectral method discretize the BVP and then solve the resulting linear system. Left: The parameter* ω_2 *is increased while* $\omega_1 = 10$, *which defines a family of BVPs for which the solution requires a high polynomial degree to resolve to machine precision. Right: The parameter* ω_1 *is increased while* $\omega_2 = 10$, *which defines a family of BVPs for which the variable coefficients require a high polynomial degree to resolve to machine precision. The polynomial degree required to resolve* $\cos(\omega_1 \pi x)$ *and* $\sin(\omega_2 \pi x)$ *on* $[-1, 1]$ *to machine precision is* $\mathcal{O}(\omega_1)$ *and* $\mathcal{O}(\omega_2)$, *respectively.*

In that situation, the matrix $\tilde{H}_k$ reduces to a tridiagonal matrix and a Lanczos procedure is used instead of an Arnoldi iteration [28]. For self-adjoint second-order differential operators, it is analogous. Thus, our operator MINRES method is a GMRES method without restarts that exploits the fact that the operator is self-adjoint. An optimized implementation of MINRES notes that $\mathcal{Q}_k$ and $\mathcal{H}_k$ in (4.3) do not need to be stored and that the solution y can be efficiently updated from previous iterates. The convergence properties of operator MINRES are analogous to the convergence behavior of MINRES for solving linear systems.

We have implemented MINRES in Chebfun in the `minres` command, which has the same practical realizations as the CG method (see section 3). Figure 7 (right) shows the convergence of the preconditioned MINRES method on the family of BVPs $-(e^x u')' - \lambda u = \sin(30\pi x)$ with $u(\pm 1) = 0$ for different values of λ.

5. An extension to even-order BVPs.

In this section, we describe the extension of continuous Krylov methods to even-order BVPs of the form:

$$\mathcal{L}u = f, \quad \Omega = (-1, 1), \qquad \frac{d^i u}{dx^i}(\pm 1) = 0, \quad 0 \le i \le K-1, \tag{5.1}$$

where $\mathcal{L} : \mathcal{H}_0^K(\Omega) \cap \mathcal{H}^{2K}(\Omega) \to L^2(\Omega)$ and

$$\mathcal{L}u = \sum_{i=0}^{2K} a_i(x) \frac{d^i u}{dx^i}, \qquad a_{2K}(x) > 0. \tag{5.2}$$

Similarly to before, if $\mathcal{L}$ is self-adjoint with positive eigenvalues, then the CG method can be used whereas MINRES is for general self-adjoint operator and GMRES is for general operators. In this setting, our canonical preconditioner $\mathcal{R}$ is the integration preconditioner repeated K times, i.e.,

$$\mathcal{R}u(x) = \int_{-1}^{x}\int_{-1}^{x_1}\cdots\int_{-1}^{x_{K-1}} u(x_K)d_{x_K}\cdots dx_1, \tag{5.3}$$

$$\mathcal{R}^*u(x) = \int_{x}^{1}\int_{x_1}^{1}\cdots\int_{x_{K-1}}^{1} u(x_K)d_{x_K}\cdots dx_1. \tag{5.4}$$

If $\mathcal{L}$ can be written in the form $\mathcal{L}u = \sum_{i=0}^{K}(-1)^i\frac{d^i}{dx^i}\big(\hat{a}_i(x)\frac{d^iu}{dx^i}\big)$ with $\hat{a}_i(x) \geq 0$ for $0 \leq i \leq K$, then $\mathcal{L}$ is self-adjoint and has positive eigenvalues. In this case, the continuous CG method described in Algorithm 2.3 can be employed without change and converges after a finite number of iteration. In particular, the condition number of the preconditioned operator can be bounded from above:

$$\kappa(\mathcal{R}^*\mathcal{L}\mathcal{R}) \leq \frac{\displaystyle\sum_{i=0}^{K}\|\hat{a}_i\|_\infty\left(\frac{\pi}{4}\right)^{2(K-i)}}{\inf_{x\in\Omega}|\hat{a}_K(x)|}\ . \tag{5.5}$$

The orthogonal projection $\Pi_{\mathcal{W}_0}$ onto the space $\mathcal{W}_0 = \{\phi \in L^2(\Omega) : \mathcal{R}\phi \in \mathcal{H}_0^K(\Omega)\}$ can also be easily expressed as

$$\Pi_{\mathcal{W}_0}u = u - p_K^{\text{best}}, \tag{5.6}$$

where $p_{K^{\text{best}}}$ is the best polynomial of degree $\leq K$ to u in $L^2([-1,1])$. The polynomial $p_{K^{\text{best}}}$ can also be simply computed by performing inner-products between u and the Legendre polynomials $P_i(x)$ for $0 \leq i \leq K$.

An auxiliary problem similar to the one in subsection 2.5 needs to be solved to ensure that the modified right-hand side g satisfies $\mathcal{R}^*g \in \mathcal{W}_0$. In this case, the auxiliary problem is to find a function find v_2 such that

$$\left[\frac{d^i}{dx^i}\left(\mathcal{R}\mathcal{R}^*\mathcal{L}\mathcal{R}v_2\right)\right](\pm 1) = \left[\frac{d^i}{dx^i}\left(\mathcal{R}\mathcal{R}^*f\right)\right](\pm 1), \tag{5.7}$$

$$\left[\frac{d^i}{dx^i}\left(\mathcal{R}v_2\right)\right](\pm 1) = 0 \tag{5.8}$$

holds for $0 \leq i \leq K-1$. These equations represent $4K$ discrete constraints and are solved via an $4K \times 4K$ linear system, finding that v_2 can be selected to be a polynomial of degree $\leq 4K-1$.

6. Extension to PDEs.

6. Extension to PDEs. Although the theory presented in section 2 extends to high dimensional problems, an implementation like the one presented in sections 3 and 4 is not straightforward. Indeed, in two dimensions, the solution of a differential equation with smooth coefficients is not necessarily smooth. For example, the solution to $\nabla^2u = 1$ on $[-1,1]^2$ with zero Dirichlet conditions has weak corner singularities. This means that even if we are able to approximate the coefficients of a PDE with a low degree polynomial, we may not be able to approximate the solution with a low degree polynomial. Therefore, looking for a practical iteration based on polynomial expansions which converges to the true solution in two dimensions is misguided as it would necessarily require the degree of iterates to explode as the number of iterations increases. In the face of this challenge, there are two possible ways forward:

- Look for the optimal solution over a finite dimensional subspace by setting $\mathcal{V}_0$ to a finite dimensional subset of $\mathcal{H}_0^1(\Omega)$. This is similar to a typical spectral-Galerkin method [33].
- Choose a different basis to represent functions that are able to resolve weak corner singularities. For example, one could enrich the basis employed by a spectral method so that the output of a preconditioner built from the Laplacian can be adequately resolved (see, for example, [32]).

In what follows, we describe an implementation of the former. We highlight that this option is unsatisfying in this context, as we are giving up several attractive properties of the one dimensional case: it does not converge to the true solution, and it is not adaptive. However, we show that this approach leads to a competitive iterative solver for spectral discretization of PDEs.

We consider a PDE

$$\begin{aligned}&\mathcal{L}u(x,y) = f(x,y), \quad (x,y) \in [-1,1]^2,\\ &u(\pm 1, y) = u(x, \pm 1) = 0,\end{aligned}$$

where $\mathcal{L}$ is a self-adjoint uniformly elliptic operator, i.e.:

$$\mathcal{L}u = -\nabla \cdot (A(x,y)\nabla u(x,y)) + c(x,y)u(x,y)$$

with $[A(x,y)]_{i,j} = a_{ij}(x,y) \in L^\infty$ for $i,j \in \{1,2\}$, $c(x,y) \in L^\infty$, and $c(x,y) \geq 0$. The ellipticity assumptions implies that the spectrum of A is uniformly bounded [8] over $[-1,1]^2$ by, say, $0 < \lambda_{\min}(A) \leq \lambda_{\max}(A) < \infty$. We restrict ourselves to a finite dimensional subspace of $\mathcal{H}_0^1(\Omega)$ and set $\mathcal{V}_{0,n} = \mathcal{P}_{n,0}^2$ where

$$\mathcal{P}_{n,0}^2 := \{\phi(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{n} \alpha_{i,j} x^i y^j \mid \phi(\pm 1, y) = \phi(x, \pm 1) = 0\} \ .$$

In order to implement an iteration similar to the one dimensional case, we need a preconditioner which is bounded, a smoother, and preconditions the Laplacian (see subsection 2.2). For this reason, we define the "square root" of the Laplacian (the analogue of the integration preconditioner for the one dimensional case) with the formal relation

$$-u_{xx} - u_{yy} = \left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right)^* \left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right) u$$

and set $\mathcal{R}u = \left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right)^{-1} u$. This preconditioner defines a preconditioned set $\mathcal{W}_{0,n} = \{\phi \mid \mathcal{R}\phi \in \mathcal{P}_{n,0}^2\}$. In order to run a two dimensional CG method similar to the one in section 3, we need to be able to apply the composition of the operators $\mathcal{R}$ and $\Pi_{\mathcal{W}_{0,n}}$, where $\Pi_{\mathcal{W}_{0,n}}$ is the L^2 orthogonal projector onto $\mathcal{W}_{0,n}$. The operator $\Pi_{\mathcal{W}_{0,n}}\mathcal{R}$ may be written as

$$(6.1) \qquad \left(\Pi_{\mathcal{W}_0}\mathcal{R}\right) u = \underset{v \in \mathcal{V}_{0,n}}{\arg\min} \left\| \left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right) v - u \right\|_{L_2}^2 \ .$$

In subsection 6.1, we describe an algorithm to compute (6.1) in $\mathcal{O}(n^2 \log(n))$ operations based on a Legendre polynomial basis and the technique presented in [10].

6.1. Computation of the two dimensional preconditioner. First, we write $v \in \mathcal{P}^2_{n,0}$ as $v = (1-x^2)(1-y^2)w$, where $w \in \mathcal{P}^2_{n-2}$. With this substitution, (6.1) is equivalent to

$$\min_{w\in\mathcal{P}_{n-2}} \left\| \left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right)(1-x^2)(1-y^2)w - u \right\|^2_{L_2}.$$

The main idea of the algorithm is to compute a preconditioner by expanding the right-hand side w in an ultraspherical basis with parameter $\lambda = 3/2$ (denoted by $C^{(3/2)}$) [20, Tab. 18.3.1] and the solution u in a Legendre basis (denoted by P), and use recurrence relations between these two bases to obtain a sparse and structured representation of the operator that can be solved fast using the alternating direction implicit (ADI) method.

First, we use the relationship [20, (18.9.20)]:

$$\frac{d}{dx}\left(\left(1-x^2\right)C_n^{(3/2)}(x)\right) = -(n+1)(n+2)P_{n+1}(x)\,. \tag{6.2}$$

This relationship implies that the matrix representation of the operation $u \mapsto ((1-x^2)u)''$ when the domain is represented in an $C^{(3/2)}$ basis and the output is represented in a P basis is:

$$\hat{D} := \begin{bmatrix} 0 & & & & \\ -2 & & & & \\ & -6 & & & \\ & & \ddots & & \\ & & & -(n+1)(n+2) & 0 \end{bmatrix}.$$

Next, we use recurrence relation [20, (18.9.8)]

$$(2n+3)\left(1-x^2\right)C_n^{3/2}(x) = -(n+1)(n+2)P_{n+2}(x) + (n+1)(n+2)P_n(x)\,, \tag{6.3}$$

which implies that the matrix representation of the operation $u \mapsto (1-x^2)u$ when the domain is represented in an $C^{(3/2)}$ basis and the output is represented in a P basis has the form

$$\hat{M} := \begin{bmatrix} \frac{1}{3} & & & & \\ 0 & \frac{6}{5} & & & \\ -\frac{1}{3} & 0 & \frac{12}{7} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{(n-1)n}{2n-1} & 0 & \frac{(n+1)(n+2)}{2n+3} \end{bmatrix}.$$

Using the normalized Legendre polynomials $\hat{P}_n := \frac{\sqrt{2n+1}}{2}P_n$, which are orthonormal with respect to the standard L^2 inner product, we find that the continuous minimization problem in (6.1) reduces to a classical linear discrete least square problem. Thus, we diagonally scale the matrices $D := S\hat{D}$ and $M := S\hat{M}$ where S is the diagonal matrix of scaling factors $[S]_{i+1,i+1} = \sqrt{2/(2i+1)}$. Using this notation, and denoting the matrix of $C^{(3/2)}$ coefficients of w by $W \in \mathbb{C}^{n-2\times n-2}$ and the matrix of $\hat{P}$ coefficients by $u \in \mathbb{C}^{n\times n}$, (6.1) is reduced to the following matrix linear equation:

$$\min_{W\in\mathbb{C}^{n-2\times n-2}} \|DUM^T - iMUD^T - W\|_F^2 = \min_{W\in\mathbb{C}^{n-2\times n-2}} \|A\mathrm{vec}(U) - \mathrm{vec}(W)\|_2^2\,.$$

Here, $\mathrm{vec}(X) \in \mathbb{C}^{n^2}$ denotes the vector obained by stacking the columns of $X \in \mathbb{C}^{n\times n}$, and $A := (D \otimes M) - i\,(M \otimes D)$. The normal equations of this least squares problem are

$$A^* A \mathrm{vec}(U) = A^* \mathrm{vec}(W)\ .$$

Noting that $D^* M$ is a skew-adjoint matrix, we find that

$$A^* A = \big(D^* D \otimes M^* M\big) + \big(M^* M \otimes D^* D\big)\,.$$

Let $R \in \mathbb{R}^{(n-2)\times(n-2)}$ be the square diagonal matrix that satisfies $RR = D^* D$, then

$$A^* A = (R \otimes R)\left[(I \otimes K) + (K \otimes I)\right](R \otimes R)\ .$$

Here, $R \otimes R$ is a $(n-2)^2 \times (n-2)^2$ diagonal matrix. Therefore, it is trivial to solve linear systems involving $R \otimes R$ in $\mathcal{O}(n^2)$. The matrix K is positive definite, banded, and $\kappa(K) = \mathcal{O}(n^4)$, which means $\left[(I \otimes K) + (K \otimes I)\right] x = b$ can be solved in $\mathcal{O}(n^2 \log(n) \log(1/\epsilon))$ operations using the ADI method with an accuracy tolerance of $0 < \epsilon < 1$ [10]. Thus, the total cost of solving (6.1) is $\mathcal{O}(n^2 \log(n))$ operations.

The remaining operations necessary to run the CG method can also be computed fast: differentiation in a Legendre basis costs $\mathcal{O}(n^2)$ operations thanks to the sparse recurrence relations in [20, (18.9.19)] and products of Legendre series can be computed in $\mathcal{O}(n^2 \log(n)^2)$ operations via a fast Legendre-to-Chebyshev transform [13]. A summary of all of the operations required to implement the two dimensional algorithm based on Legendre polynomials is given in Table 2. The cost of one CG iteration is $\mathcal{O}(n^2 \log(n)^2)$ operations.

The same argument as found in Theorem 2.6 shows that

$$\kappa_{\mathcal{W}_{0,n}}(\mathcal{R}^* \mathcal{L} \mathcal{R}) \le \frac{\lambda_{\max}(A) + \|c\|_\infty \|\mathcal{R}\|_{\mathrm{op}}^2}{\lambda_{\min}(A)}\ .$$

This implies that the number of iterations required to solve the system to some prescribed accuracy is independent of n. Consequently, the computational complexity of this spectral solver is $\mathcal{O}(n^2 \log(n)^2)$ operations.

We demonstrate the scaling of the algorithm and compare its running time to the two dimensional extension of the ultraspherical spectral method [36]. We study two regimes: (1) the variable coefficients can be approximated with low degree polynomials, and (2) the variable coefficients require a high degree polynomials to be approximated. In the first regime illustrated in Figure 9, the ultraspherical spectral method produces sparse matrices and the resulting linear system can be solved by

TABLE 2
Implementation and complexity of all operations required to implement the PCG method in two dimensions based on a Legendre basis.

Operations	Implementation	Complexity
Product	Legendre–Chebyshev transform [13] and fast Chebyshev product using FFT	$\mathcal{O}(n^2 \log(n)^2)$
Differentiation	recurrence [20, (18.9.19)]	$\mathcal{O}(n^2)$
Transform between $C^{(3/2)}$ and P	recurrence [20, (18.9.7)]	$\mathcal{O}(n^2)$
Inner product	standard $\mathbb{R}^n$ inner product	$\mathcal{O}(n^2)$
Preconditioner and projector	ADI [10]	$\mathcal{O}(n^2 \log(n))$

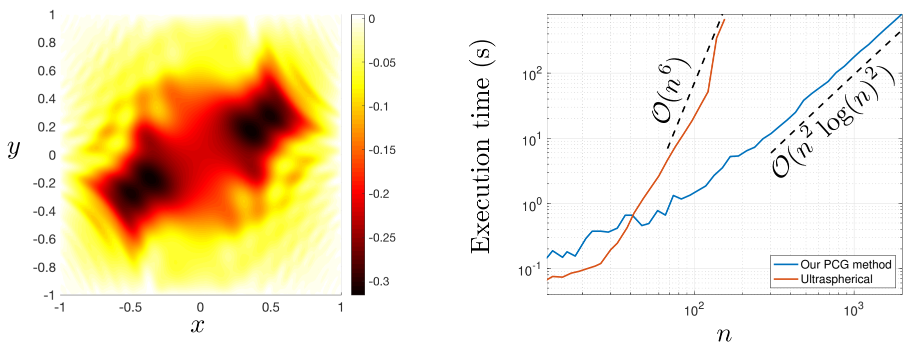

FIG. 9. *Numerical experiment on a PDE with variable coefficients that require high degree polynomials to approximate:* $-\frac{\partial}{\partial x}((2+\sin(5xy\pi))\frac{\partial}{\partial x}u) - \frac{\partial}{\partial y}((2+\cos(50\pi x)\sin(10\pi y))\frac{\partial}{\partial y}u) = -100x\sin(20\pi x^2 y)\cos(4\pi(x+y))$ *with zero Dirichlet conditions. Right: plot of the solution for* $n = 1000$. *Left: Comparison of execution timings of our PCG method and the ultrapsherical spectral method* [36] *for different values of* n. *The CG iteration was stopped when* $\|r\|_2 < 10^{-13}$.

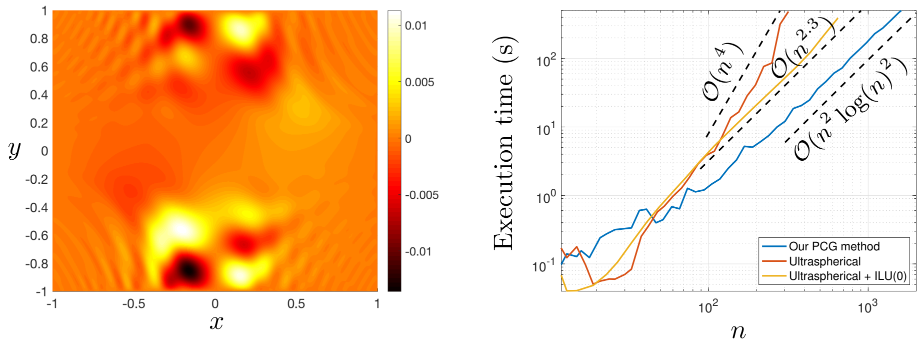

FIG. 10. *Numerical experiment on a PDE with variable coefficients that can be resolved with low degree polynomials:* $-\frac{\partial}{\partial x}((2+\sin(\pi x)y^2)\frac{\partial}{\partial x}u) - \frac{\partial}{\partial y}((2+\cos(\pi x)\sin(\pi y))\frac{\partial}{\partial y}u) = 10y^2\sin(20\pi x^2 y)\cos(4\pi(x+y))$ *with zero Dirichlet conditions. Left: plot of the solution for* $n = 1000$. *Right: Comparison of execution timings of our preconditioned CG method (blue line), the ultrapsherical spectral method using a sparse direct solver (the red line), and the ultrapsherical spectral method using GMRES preconditioned with ILU(0). The iterative methods were stopped when* $\|r\|_2 < 10^{-13}$.

a direct solver in $\mathcal{O}(n^4)$ operations. We also compare against a GMRES method preconditioned with an incomplete LU factorization (ILU(0)) [30, Chap. 10]. The execution time of the preconditioned GMRES method is observed to have complexity $\mathcal{O}(n^{2.3})$. In the second regime illustrated in Figure 10, the matrix producted by the ultraspherical spectral method is dense and a direct solver requires $\mathcal{O}(n^6)$ operations. In this case, we did not consider an iterative solver as the matrix is dense.

Conclusion. Operator analogues of the CG method, MINRES, and GMRES are derived for solving BVPs on $(-1, 1)$ that employ operator-function products. An operator preconditioner ensures that only a finite number of Krylov iterations are necessary to compute an approximate solution, and an orthogonal projection operator guarantees that the computed Krylov subspace imposes the boundary conditions of the BVP. The resulting iterative solvers are able to compute solutions from $\mathcal{H}_0^1(\Omega)$ and are competitive BVP solvers when a fast operator-function product is available.

Acknowledgements. The authors tried several different approaches to developing an operator analogue of Krylov methods, and we thank Jiahao Chen and Alan Edelman for interesting discussions on several unfruitful attempts. We are grateful to Nick Trefethen for popularizing the idea of operator analogues of Krylov subspace methods in 2012, and we thank Jared Aurentz, Sheehan Olver, and Marcus Webb for sharing their thoughts on the topic. We also thank David Bindel, Toby Driscoll, Dan Fortunato, Hadrien Montanelli, and Heather Wilber for comments and reactions on different versions of this work. We also thank the referees for a careful reading of the manuscript and valuable comments.

REFERENCES

[1] J. L. Aurentz and L. N. Trefethen, *Chopping a Chebyshev series*, ACM Trans. Math. Softw., 43 (2017), p. 33.

[2] O. Axelsson and J. Karátson, *Equivalent operator preconditioning for elliptic problems*, Numer. Algorithms, 50 (2009), pp. 297–380.

[3] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods: Fundamentals in Single Domains*, Springer, New York, 2010.

[4] J. W. Daniel, *The conjugate gradient method for linear and nonlinear operator equations*, SIAM J. Numer. Anal., 4 (1967), pp. 10–26.

[5] T. A. Driscoll, F. Bornemann, and L. N. Trefethen, *The chebop system for automatic solution of differential equations*, BIT, 48 (2008), pp. 701–723.

[6] T. A. Driscoll and N. Hale, *Rectangular spectral collocation*, IMA J. Numer. Anal., 36 (2015), pp. 108–132.

[7] T. A. Driscoll, N. Hale, and L. N. Trefethen, *Chebfun Guide*, Pafnuty Publications, Oxford, 2014.

[8] L. C. Evans, *Partial Differential Equations*, Graduate Studies in Mathematics 19, 2nd ed., American Mathematical Society, 2010.

[9] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, vol. 1, Cambridge university Press, Cambridge, 1998.

[10] D. Fortunato and A. Townsend, *Fast Poisson solvers for spectral methods*, preprint, arXiv:1710.11259, 2017.

[11] W. M. Gentleman, *Implementing Clenshaw-Curtis quadrature,* ii *computing the cosine transformation*, Commun. ACM, 15 (1972), pp. 343–346.

[12] N. Gould, D. Orban, and T. Rees, *Projected Krylov methods for saddle-point systems*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1329–1343.

[13] N. Hale and A. Townsend, *A fast, simple, and stable chebyshev–legendre transform using an asymptotic formula*, SIAM J. Sci. Comput., 36 (2014), pp. A148–A167.

[14] P. R. Halmos, *A Hilbert Space Problem Book*, vol. 19, Springer Science & Business Media, New York, 2012.

[15] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, vol. 49, J. Res. Nat. Bureau Stand., 1952.

[16] R. Hiptmair, *Operator preconditioning*, Comput. Math. Appl., 52 (2006), pp. 699–706.

[17] R. C. Kirby, *From functional analysis to iterative methods*, SIAM Rev., 52 (2010), pp. 269–293.

[18] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-state and Time-dependent Problems*, vol. 98, SIAM, Philadelphia, 2007.

[19] J. Liesen and Z. Strakoš, *Krylov Subspace Methods: Principles and Analysis*, Oxford University Press, Oxford, 2013.

[20] D. W. Lozier, *Nist digital library of mathematical functions*, Ann. Math. Art. Intell., 38 (2003), pp. 105–119.

[21] J. Málek and Z. Strakoš, *Preconditioning and the Conjugate Gradient Method in the Context of Solving PDEs*, SIAM, Philadelphia, 2014.

[22] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials*, CRC Press, Boca Raton, FL, 2002.

[23] G. Meurant, *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations*, SIAM, Philadelphia, 2006.

[24] S. Olver, *GMRES for the differentiation operator*, SIAM J. Numer. Anal., 47 (2009), pp. 3359–3373.

[25] S. Olver and A. Townsend, *A fast and well-conditioned spectral method*, SIAM Rev., 55 (2013), pp. 462–489.

[26] E. L. Ortiz, *The tau method*, SIAM J. Numer. Anal., 6 (1969), pp. 480–492.

[27] R. Pachón, R. B. Platte, and L. N. Trefethen, *Piecewise-smooth chebfuns*, IMA J. Numer. Anal., 30 (2009), pp. 898–916.

[28] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[29] B. P. Rynne and M. A. Youngson, *Linear Functional Analysis*, Springer Science & Business Media, New York, 2000.

[30] Y. Saad, *Iterative methods for sparse linear systems*, vol. 82, SIAM, Philadelphia, 2003.

[31] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[32] W. W. Schultz, N. Lee, and J. P. Boyd, *Chebyshev pseudospectral method of viscous flows with corner singularities*, J. Sci. Comput., 4 (1989), pp. 1–24.

[33] J. Shen, *Efficient spectral-Galerkin method* i. *direct solvers of second-and fourth-order equations using Legendre polynomials*, SIAM J. Sci. Comput., 15 (1994), pp. 1489–1505.

[34] J. Shen, T. Tang, and L.-L. Wang, *Spectral Methods: Algorithms, Analysis and Applications*, vol. 41, Springer Science & Business Media, New York, 2011.

[35] G. W. Stewart, *Afternotes goes to graduate school: Lectures on advanced numerical analysis*, SIAM, Philadelphia, 1998.

[36] A. Townsend and S. Olver, *The automatic solution of partial differential equations using a global spectral method*, J. Comput. Phys., 299 (2015), pp. 106–123.

[37] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.

[38] L. N. Trefethen, *Approximation Theory and Approximation Practice*, vol. 128, SIAM, Philadelphia, 2013.

[39] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[40] H. A. Van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, vol. 13, Cambridge University Press, Cambridge, 2003.

[41] J. A. C. Weideman and L. N. Trefethen, *The eigenvalues of second-order spectral differentiation matrices*, SIAM J. Numer. Anal., 25 (1988), pp. 1279–1298.