SIAM J. MATRIX ANAL. APPL.
Vol. 42, No. 3, pp. 1248–1267

PROXY-GMRES: PRECONDITIONING VIA GMRES IN POLYNOMIAL SPACE*

XIN YE†, YUANZHE XI‡, AND YOUSEF SAAD§

Abstract. This paper proposes a class of polynomial preconditioners for solving non-Hermitian linear systems of equations. The polynomial is obtained from a least-squares approximation in polynomial space instead of a standard Krylov subspace. The process for building the polynomial relies on an Arnoldi-like procedure in a small dimensional polynomial space and is equivalent to performing GMRES in polynomial space. It is inexpensive and produces the desired polynomial in a numerically stable way. A few improvements to the basic scheme are discussed including the development of a short-term recurrence and the use of compound preconditioners. Numerical experiments, including a test with challenging nonnormal three-dimensional Helmholtz equations and a few publicly available sparse matrices, are provided to illustrate the performance of the proposed preconditioners.

Key words. polynomial preconditioning, polynomial iteration, orthogonal polynomial, short-term recurrence, Helmholtz equation

AMS subject classifications. 15A06, 49M25, 65F08, 65F10, 65F50

DOI. 10.1137/20M1342562

1. Introduction.

We consider solving a large non-Hermitian linear system of equations

$$Ax = b, \tag{1.1}$$

where $A \in \mathbb{C}^{N\times N}$ is non-Hermitian and $x, b \in \mathbb{C}^N$. A Krylov subspace method accelerated by a certain type of preconditioner is often preferred for this type of problems, e.g., GMRES with a form of the incomplete LU (ILU) factorization. However, when the coefficient matrix A is highly indefinite (eigenvalues of A appear on both sides of the imaginary axis) or extremely ill-conditioned, this method may suffer from slow convergence or even stagnation due to stability issues [42]. Furthermore, since both the construction and application phases of the classical ILU preconditioners are intrinsically sequential and difficult to parallelize, they cannot easily take full advantage of modern high-performance computing architectures such as distributed memory machines or graphics processing units (GPUs). Recent efforts to develop fine-grained parallel ILU factorization can be found in [9]. At the same time recent efforts have been devoted to develop preconditioners based on low-rank approximations that are highly parallel [28, 31, 53, 14]. These preconditioners explore the recursive or hierarchical low-rank approximation of the Schur complement or its inverse and only apply ILU to the diagonal blocks in the reordered matrix. These preconditioners are far more elaborate than their ILU counterparts, but they tend to be more robust for indefinite and nonsymmetric linear systems.

*Received by the editors June 2, 2020; accepted for publication (in revised form) by L. Giraud May 27, 2021; published electronically August 5, 2021.
https://doi.org/10.1137/20M1342562
Funding: The work of the authors was supported by the National Science Foundation grants DMS-1521573, DMS-1912048, and OAC-2003720.
†Hewlett Packard Enterprise, Bloomington, MN 55425 USA (xin.ye@hpe.com).
‡Department of Mathematics, Emory University, Atlanta, GA 30322 USA (yxi26@emory.edu).
§Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA (saad@umn.edu).

This paper discusses a new class of polynomial preconditioning techniques for solving (1.1). These preconditioners can be either used in a standalone way, or they can be combined with other types of preconditioners to improve their efficiency. We begin by observing that most classical acceleration schemes, such as the conjugate gradient or GMRES algorithms [42], which are a form of polynomial iteration. Given an initial guess x_0 and residual $r_0 = b - Ax_0$, the approximate solution $\tilde{x}$ in a given iteration is of the form:

$$\tilde{x} = x_0 + p(A)r_0,$$

where p is a polynomial, and its related residual is equal to

$$\tilde{r} = b - A\tilde{x} = (I - Ap(A))r_0 \equiv \rho(A)r_0. \tag{1.2}$$

Note that the approximate solution is a member of the affine Krylov subspace $x_0 + \mathcal{K}_m(A, r_0)$, where $\mathcal{K}_m(A, r_0) \equiv \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\}$. The acceleration procedures based on Krylov subspace methods that have been developed in the literature are all based on polynomial iterations where the iterates are of the form given above, and the polynomials are obtained using various criteria. For example, the criterion employed in GMRES [43] is to select the polynomial p to make the residual norm $\|\tilde{r}\|_2$ as small as possible. The Chebyshev "semi-iterative" method [22, 33, 34, 24] constructs p so that the residual polynomial $\rho(t)$ is an appropriately shifted and scaled Chebyshev polynomial of the first kind. The residual polynomial is built so that it is small in an ellipse that encloses the spectrum of the matrix A. In these methods, the polynomial p can be either used directly to solve linear systems approximately in an iterative scheme as in [22, 24] and other works, or it can be exploited as a preconditioner in combination with an acceleration such as GMRES, for example.

Polynomial preconditioners are quite appealing because they are simple to use and because they can be highly effective for some problems. The construction of the polynomial preconditioner does not involve matrix factorizations and does not rely on any reordering scheme. Perhaps the most attractive feature of polynomial precondioners is that their application relies on one single operation, namely, the matrix-vector multiplication associated with the original coefficient matrix A. This operation has been studied and optimized for decades (see, e.g., [6, 51, 7, 29]) and can be made extremely efficient for sparse matrices on most architectures. In addition, the applications of such preconditioners are completely free of inner product which is communication intensive and limits the performance in a distributed memory environment. The paper brings three main contributions which are summarized below.

Improved numerical stability. In the past, several polynomial preconditioners have been proposed in the literature [41, 36, 35, 18]. However, these methods often suffer from numerical stability issues as they express the polynomial in some canonical basis (e.g., as a Chebyshev series). In contrast, the proposed methods build a polynomial basis via an Arnoldi-like procedure. This procedure represents the polynomial implicitly, and its numerical stability is monitored in order to compute the polynomial preconditioners with good accuracy for arbitrarily high degrees.

Spectrum-based preconditioning. The proposed polynomial preconditioner is constructed by solving a discrete least-squares problem based on minimizing the residual polynomial inside a certain calculated contour that excludes the origin and includes all of the eigenvalues of the coefficient matrix. We note here that polynomial preconditioners based directly on GMRES polynomials are not guaranteed to yield a good preconditioner even when the matrix is not highly nonnormal [50, 18]. Using the

spectrum as a criterion can be quite effective in the common practical situation where the matrix is not highly nonnormal.

Efficient construction and application. The proposed polynomial preconditioners are built in a carefully designed polynomial space which has much smaller dimension compared to the matrix size. As a result, the cost of building the polynomial is essentially negligible. In the application phase, a technique based on short-term recurrence is proposed in section 3.4 which can significantly accelerate the application of the preconditioner on a vector and reduce the storage requirement.

A few comments are in order. First, it is clear that any method based on trying to minimize the residual polynomial on the spectrum of A is bound to be limited in scope to the case where the matrix is not highly nonnormal. However, it is also important to realize that these are the important cases. Highly nonnormal cases are rare when solving linear systems that arise from partial differential equations or from most engineering problems. In fact, one can argue that if a matrix is highly nonnormal, no iterative method will perform well. The simplest example of a case that will fail is an upper triangular matrix with ones on the diagonal and large values above it. No polynomial preconditioner can help for a case like this, and GMRES will typically take the full n steps to produce a solution. Thus, alternatives based, for example, on pseudospectra or field of values are not likely to work and are not explored in this paper.

Second, it may be thought that relying on a contour that encloses eigenvalues may be unreliable because the contour can miss eigenvalues. In fact, this has not been an issue in our experiments. The contour does not need to be very accurate since the polynomial is used for preconditioning and the outer acceleration loop, e.g., GMRES, will eliminate components associated with the missed eigenvalues. A sufficient number of Arnoldi steps are used to produce fairly accurate eigenvalues from which a polygonal region is built. It is important to ensure that the origin is not enclosed in the contour, even at the cost of leaving some eigenvalues outside as these will be captured by the accelerator. Such techniques are discussed in the numerical experiments section.

Finally, it is important to stress that this work is motivated in a big part by the advantage that polynomial preconditioners present for some types of computing platforms. Polynomial preconditioning was a popular topic in the 1980s and early 1990s; see, e.g., [25, 3, 4, 38, 5, 1, 19, 16, 39], among many references. Research on polynomial preconditioners later faded for a number of reasons. Using a low-degree polynomial, as was often suggested, usually leads to modest gains if any, when compared to just using a larger Krylov subspace. In fact, in terms of the total number of matvecs required to solve a given system, polynomial preconditioning is 'suboptimal' relative to an accelerator like the conjugate-gradient method or (full) GMRES. However, in the specific context of high-performance computing, a known attraction of polynomial precondioners is that they reduce the total number of inner products required, which cause time-consuming sychronization points. Recent work on eigenvalue problems [30] showed that *it is important to be able to use high-degree polynomials when possible*. In the eigenvalue context, the analogue of polynomial preconditioning is known as *polynomial filtering*, and the work [30] as well as [45] demonstrated the effectiveness of resorting to polynomial filters of very high degree, in the thousands, when necessary. In the Hermitian case this is relatively easy to achieve thanks to the stability of Chebyshev bases when used to represent the filters. Indeed, on the interval $[-1, 1]$, Chebyshev polynomials of degrees in the thousands cause no numerical issues and can be safely used. Unfortunately, we do not have a similar tool in the non-Hermitian case. Chebyshev polynomials are associated with confocal ellipses

that have focii at $-1, 1$ (or $-i, i$) in the complex plane, but general complex spectra are rarely well fitted by ellipses. Even when they are, Chebyshev polynomials still face difficulties as they tend to grow fast in the ellipse outside the interval $[-1, 1]$ and become linear dependent when their degrees increase, as can be understood from the condition numbers of their related Gram matrices; see [40]. This limits the possible degree with which polynomial precondioners developed in this way can be applied. One of the main goals of this paper is to answer the following question: *can polynomial preconditioners with high degrees be developed for the non-Hermitian case, and can they lead to effective iterative techniques for solving challenging problems?*

The rest of this paper is organized as follows. Section 2 introduces a few ways to derive polynomial preconditioners based on solving minimax problems. Section 3 presents an Arnoldi-like procedure to generate a stable polynomial basis based on the boundary of the spectrum of the coefficient matrix. Several improvements are discussed in section 4, and numerical examples are provided in section 5. Finally, concluding remarks are stated in section 6.

2. Polynomial construction via an explicit basis. In this section, we will discuss a few ways to derive a polynomial preconditioner when an explicit basis for the polynomial space is given.

2.1. Classical minimax problem. In many applications, the boundary of the spectrum of A is not hard to estimate. For example, this can be done either by analyzing the physical problem [32] from where (1.1) is derived or approximated by methods such as the Arnoldi iteration [2, 17]. Assume that all eigenvalues of A are contained in a simply connected domain $\Omega \subset \mathbb{C}$ and denote by $\Gamma = \partial\Omega$ the boundary of Ω. Here, we further assume that Ω does not contain the origin and that Γ is piecewise smooth; see Figure 2.1 for an illustration. The methods we will consider can be generalized to cases where the region Ω is the union of several disconnected regions $\Omega_1, \Omega_2, \ldots$, but we will not consider this situation in this paper.

From (1.2) we have that

$$\|\tilde{r}\| \leq \|I - Ap(A)\| \|r_0\|.$$

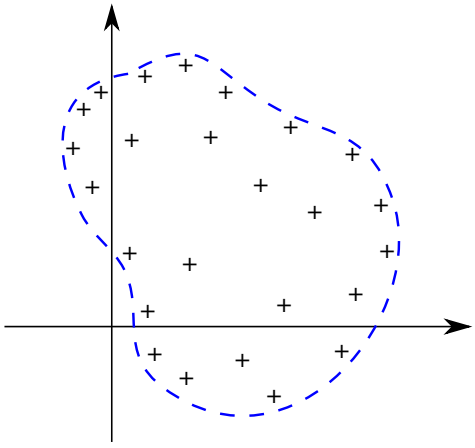


FIG. 2.1. *Eigenvalues of the matrix enclosed by a closed curve.*

In order to make $\|\tilde{r}\|$ small, we could choose p so that $\|I - Ap(A)\|$ is small. A straightforward criterion to ensure this is simply to require that $|1 - zp(z)|$ be small for all $z = \lambda$, where λ is an eigenvalue of A. Unfortunately, this approach involves all the eigenvalues of A, which is not practically feasible, so an alternative is to seek p so that the maximum of $|1 - zp(z)|$ in the region Ω is small. Since we assume the eigenvalues of A are enclosed by Γ, and since $1 - zp(z)$ is holomorphic, the maximum modulus principle [46] tells us that the maximum value of $|1 - zp(z)|$ on Ω is achieved on the boundary Γ. Thus for a fixed $m > 0$, the sought-after polynomial p can be characterized by the following minimax problem:

$$\min_{p \in \mathcal{P}_{m-1}} \max_{z \in \Gamma} |1 - zp(z)|, \tag{2.1}$$

where $\mathcal{P}_{m-1}$ denotes the set of all complex polynomials of degree less than m.

It is important to note that an approach based on this framework can be viewed as a heuristic only because in the highly nonnormal case the norm of $\|I - Ap(A)\|$ is not always tightly related to the maximum of $|1 - zp(z)|$ on the contour Γ that contains the spectrum; see, for example, the articles on the Crouzeix conjecture [11, 12, 13] and the convergence of GMRES on nonnormal matrices [23]. However, for most practical problems minimizing some norm of $|1 - zp(z)|$ on the contour Γ will yield good results.

Defining the Chebyshev norm on any set $\mathcal{D} \subset \mathbb{C}$ of a function f by $\|f\|_{\mathcal{D}} = \max_{z \in \mathcal{D}} |f(z)|$, the minimax problem (2.1) can be rewritten as

$$\min_{p \in \mathcal{P}_{m-1}} \|1 - zp(z)\|_{\Gamma}. \tag{2.2}$$

This is a Chebyshev approximation problem in functional form with a domain that is a continuous subset of the complex plane. The problem can be solved by a Remez-like algorithm [8, 49, 37] or the Lanczos τ-method [26, 10]. However, when the geometry of Γ becomes irregular or the degree of the polynomial increases, these methods might fail. As a result, we will not attempt to solve the minimax problem (2.1) directly.

We can instead solve a discrete version of the problem; i.e., we can simplify (2.1) by replacing the continuous contour Γ by a discrete one. Let $\Gamma_n = \{z_1, z_2, \ldots, z_n\}$ be an n-point discretization of the boundary Γ. This discretization should capture the geometric characteristics of Γ; a uniform discretization of Γ usually suffices in practice. In certain cases when Γ contains a high curvature or discontinuous part, we can either add additional points to refine the discretization in this area or simply replace this part by a smoother curve before the discretization. We then consider the Chebyshev norm on the discrete set Γ_n and define the following *discrete minimax problem*:

$$\min_{p \in \mathcal{P}_{m-1}} \|1 - zp(z)\|_{\Gamma_n}. \tag{2.3}$$

With a given basis $\{\phi_i\}_{i=1}^m$ for $\mathcal{P}_{m-1}$, let $p(z) = \sum_{i=1}^m \alpha_i \phi_i(z)$, and denote by $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_k]^T \in \mathbb{C}^m$ the column vector of all the coefficients; (2.3) becomes

$$\min_{\alpha \in \mathbb{C}^m} \max_{1 \le i \le n} \left| 1 - z_i \sum_{j=1}^m \alpha_j \phi_j(z_i) \right|.$$

Define an $n \times m$ matrix F with entries given by

$$f_{ij} = z_i \phi_j(z_i), \quad 1 \le i \le n,\ 1 \le j \le m,$$

and $e \in \mathbb{C}^n$ the column vector of all ones; (2.3) can be reformulated in the matrix form as

$$\min_{\alpha \in \mathbb{C}^m} \|e - F\alpha\|_\infty.$$

We refer the readers to [48, 47, 52, 27] for some discussions on algorithms for solving the above complex linear programming problem. This problem uses the infinity norm in $\mathbb{C}^n$. We will not consider this approach in the remainder of the paper. Instead we will replace the infinity norm by the 2-norm in $\mathbb{C}^n$. The least-squares polynomial will be computed by a GMRES-like procedure in polynomial space which is described next.

3. Polynomial construction via an Arnoldi process. Define an inner product for the polynomial space as

$$\langle p_1, p_2 \rangle = \sum_{i=1}^{n} p_1(z_i)\overline{p_2(z_i)}. \tag{3.1}$$

This sesqui-linear form is a valid inner product of the space of polynomials $\mathcal{P}_m$ as long as m is smaller than the number of points n . We will denote by $\|\cdot\|_\omega$ the related norm. Then we would like to solve the following discrete least-squares problem instead of (2.2):

$$\min_{p \in \mathcal{P}_{m-1}} \|1 - zp(z)\|_\omega^2. \tag{3.2}$$

Instead of specifying a basis $\{\phi_i(z)\}_{i=1}^m$ in advance as in section 2, we will actually build the polynomial basis dynamically in an Arnoldi-like a process.

3.1. GMRES in polynomial space. The construction procedure for the optimal polynomial is similar to GMRES in vector space and is described in Algorithm 3.1. For the sake of conformity with the notation used in the standard Arnoldi process, the polynomial basis vector of degree ℓ is represented by $q_{\ell+1}$ instead of q_ℓ.

It is easy to see that the Arnoldi-like process described in Algorithm 3.1 will indeed generate a set of orthonormal polynomial basis $\{q_i\}_{i=1}^m$ with respect to the inner product (3.1); there will be no stability issue even for high degrees due to the full orthogonalization by a modified Gram–Schmidt algorithm as presented in steps 4 to 7. The question now is how to represent the polynomials and how to carry out the actual computations that are involved in Algorithm 3.1. In fact we have a number of choices of which we will only retain one. The simplest choice, a poor one for obvious reasons of stability, is to use the power series representation. In this case, a polynomial $p(z) = \alpha_0 + \alpha_1 z + \cdots \alpha_{m-1} z^{m-1}$ will be represented by the vector $[\alpha_0,\ \alpha_1, \ldots, \alpha_{m-1}]^T \in \mathbb{C}^m$. For example, the polynomial multiplication $q := zq_j$ in step 3 amounts to shifting all components of the representing vector down by one position and putting a zero in the first position; addition, subtraction, and scalar multiplication all translate to the corresponding operation on the vector; inner products are also easy to compute efficiently once the Gram matrix of the power series basis is computed.

However, we will not use any explicit representations because, as we will show later, we are more interested in the coefficients h_{ij} than the polynomials themselves. Therefore, we will represent the polynomials implicitly by the evaluations on the points $\{z_i\}_{i=1}^n$; i.e., a polynomial p is represented by a vector $[p(z_1), p(z_2), \ldots, p(z_n)]^T \in \mathbb{C}^n$.

Algorithm 3.1 *The Arnoldi-like process in polynomial space*

Input: Discretization points $\{z_i\}_{i=1}^n$ on Γ and degree m
Output: Orthogonal polynomial basis $\{q_i\}_{i=1}^{m+1}$

1: Set $q_1 = \mathbb{1}/\|\mathbb{1}\|_\omega$ ▷ *q_1 is of degree 0 and norm 1*
2: **for** $j = 1, 2, \ldots, m$ **do**
3: Compute $q := zq_j$ ▷ *Increase degree*
4: **for** $i = 1, 2, \ldots, j$ **do**
5: Compute $h_{ij} = \langle q, q_i \rangle$
6: Compute $q = q - h_{ij} q_i$
7: **end for** ▷ *Full orthogonalization*
8: Compute $h_{j+1,j} = \|q\|_\omega$
9: Compute $q_{j+1} = q/h_{j+1,j}$ ▷ *Normalize the new basis*
10: **end for**

Under this representation, the polynomial multiplication $q := zq_j$ in step 3 will be translated simply into the entrywise multiplication of two vectors of length n, and the inner products in steps 5 and 8 become standard inner products in vector space $\mathbb{C}^n$.

We now address the solution of the discrete least-squares problem (3.2). Define the $(m+1) \times m$ matrix H_m, where $(H_m)_{ij} = h_{ij}$, for $i \le j+1$ and $(H_m)_{ij} = 0$, for $i > j+1$, so that H_m is an upper-Hessenberg matrix. If we abuse the notation and replace all polynomials by their vector representations in Algorithm 3.1, then the constant 1 in (3.2) becomes βq_1, where $\beta = \|\mathbb{1}\|_\omega = \sqrt{n}$. Define $Q_\ell = [q_1, q_2, \ldots, q_\ell]$ to be the column concatenation of the first ℓ basis vectors; then each Q_ℓ for all $1 \le \ell \le m+1$ has orthonormal columns. If p is expressed linearly in the basis $\{q_1, q_2, \ldots, q_m\}$ as

$$p = \sum_{i=1}^{m} \alpha_i q_i = Q_m \alpha,$$

where $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_m]^T$, then the polynomial zp in (3.2) becomes

$$\begin{aligned} zp &= \sum_{j=1}^{m} \alpha_j (zq_j) = \sum_{j=1}^{m} \alpha_j \sum_{i=1}^{j+1} h_{ij} q_i = \sum_{j=1}^{m} \alpha_j \sum_{i=1}^{m+1} h_{ij} q_i \quad (h_{ij} = 0 \text{ when } i - j > 1) \\ &= \sum_{i=1}^{m+1} \sum_{j=1}^{m} q_i h_{ij} \alpha_j = Q_{m+1} H_m \alpha. \end{aligned}$$

In the end we observe that solving (3.2) amounts to minimizing with respect to $\alpha \in \mathbb{C}^n$ the objective function

$$J(\alpha) = \|\beta q_1 - Q_{m+1} H_m \alpha\|_2^2.$$

Since Q_{m+1} has orthonormal columns, and $q_1 = Q_{m+1} e_1$, where $e_1 = [1, 0, \ldots, 0]^T$ is a vector of length $m+1$, this can be further reduced to

$$J(\alpha) = \|\beta e_1 - H_m \alpha\|_2^2. \tag{3.3}$$

Note that this is a standard least-squares problem, which is the same as the one in GMRES in complex spaces because the approach shares many core ingredients of GMRES.

Once α is found from (3.3), we obtain a polynomial p of degree $m-1$, and the matrix M defined by $M^{-1} = p(A)$ can be used as a preconditioner for solving the linear system $Ax = b$.

To apply M^{-1} to a vector v, note that

$$M^{-1}v = p(A)v = \sum_{i=1}^{m} \alpha_i q_i(A)v := \sum_{i=1}^{m} \alpha_i v_i, \tag{3.4}$$

where we define $v_i \equiv q_i(A)v$ for $1 \le i \le m$. Since $q_1 = \mathbb{1}/\sqrt{n}$ so

$$v_1 = q_1(A)v = Iv/\sqrt{n} = v/\sqrt{n}. \tag{3.5}$$

From the Arnoldi-like process Algorithm 3.1 we have that $zq_i = \sum_{j=1}^{i+1} h_{ji} q_j$; thus

$$Av_i = Aq_i(A)v = \left[\sum_{j=1}^{i+1} h_{ji} q_j(A)\right] v = h_{i+1,i} v_{i+1} + \sum_{j=1}^{i} h_{ji} v_j, \quad 1 \le i \le m-1,$$

and hence

$$v_{i+1} = \frac{1}{h_{i+1,i}} \left(Av_i - \sum_{j=1}^{i} h_{ji} v_j \right), \quad 1 \le i \le m-1. \tag{3.6}$$

The v_i's can be computed recursively from (3.5) and (3.6); and the final result is just a linear combination of v_i's with the coefficient α. Note that the only information needed is the precalculated entries available in H_m; the basis Q_{m+1} is not involved directly in the least-squares problem (3.3) for finding α or in applying the preconditioner (3.4)–(3.6).

We note that the idea of using an Arnoldi-like procedure to generate orthogonal polynomials is not entirely new. In fact, the framework is similar to what was discussed in [41] where the Chebyshev polynomial basis is used to construct a polynomial p that minimizes the residual polynomial $1 - zp(z)$ under some specially defined norm. But, as mentioned in [41], this algorithm suffers from numerical stability issues, and the polynomial degree has to be kept low. The main reason is that, in the complex plane, Chebyshev polynomials on ellipses tend to become linear dependent as their degrees increase, as can be seen from the condition numbers of their related Gram matrices; see [40].

The same argument holds true for other methods that try to construct a polynomial from the span of other bases. Since the algorithm proposed in this manuscript uses only the inner product (3.1) and implicitly constructs the polynomial p, p will be accurately computed for high degrees (as long as $m < n$). Another class of methods constructs the polynomial by finding all of its roots and represents the polynomial by the product of a series of degree-one polynomials, e.g., in [35, 18]. These methods also suffer from stability issues when the degree is high.

3.2. Connection to GMRES. In comparing the proposed approach to the standard GMRES approach, one can observe that (3.3) is exactly the same least-squares problem that we solve in standard GMRES except that the coefficients of H_m are generated in a vector space of dimension n, the number of points on the contour. Looking more carefully at the algorithm, it is also possible to show that in fact *it is equivalent to the standard GMRES algorithm applied to the diagonal matrix whose*

entries are the discretization points $z_1, z_2, \ldots, z_n$. They are equivalent in the sense that they would generate the same Hessenberg matrix H_m and in the end also the same polynomial. For this reason we may refer to this approach as a *proxy-GMRES* algorithm since the original matrix is replaced by a small ("proxy") diagonal matrix whose spectrum captures the original spectrum well.

3.3. Approximating the contour Γ. Sometimes it is possible to have a contour Γ that is available from information obtained from the original physical problem under consideration. When this is not the case, an approximate contour must be obtained from approximate eigenvalues of the coefficient matrix A. First we run several steps of the Arnoldi algorithm which produces Ritz values that are approximations of the eigenvalues of A. These Ritz values tend to approximate those eigenvalues on the boundary of the spectrum. We found that running 60–100 steps of the Arnoldi algorithm is usually good enough for capturing the required information to build the contour in all the numerical examples in this paper. The next step is to find an approximate boundary Γ where a subset of points from the Ritz values forms the vertices of a polygon. Note that Γ is not necessarily convex and that its boundary can shrink towards the interior. Finally, the contour Γ is discretized on each of its continuous parts. Our experience indicates that a uniform discretization with a fixed step size h suffices; this means the number of discretization points is only affected by the length of the approximated boundary Γ and irrelevant of the order of the coefficient matrix. We use $h = 0.005$ in all our tests.

3.4. Short-term recurrence. Because the matrix H_m in Algorithm 3.1 is an upper Hessenberg matrix, computing $p(A)v$ for a degree $m-1$ polynomial p costs $\mathcal{O}(m^2N)$ operations and requires $\mathcal{O}(mN)$ storage. This implies that despite the good numerical stability of the algorithm, its computation cost and storage quickly become unacceptably high as m increases. Motivated by the three-term recurrence for Chebyshev polynomials, we will show in this section that a short-term recurrence can be exploited to significantly reduce these costs.

The basic idea is to replace the full orthogonalization in Steps 4 to 7 in Algorithm 3.1 by a partial orthogonalization. That is, the newly generated polynomial q in Step 3 is only orthogonalized against the most recent k basis, which leads to the following short-term recurrence relation

$$t_{j+1,j}\hat{q}_{j+1} = z\hat{q}_j - \sum_{i=j-k+1}^{j} t_{ij}\hat{q}_i, \quad 1 \le j \le m,$$

where t_{ij} $(1 \le i \le j)$, $t_{j+1,j}$ and $\hat{q}_{j+1}$ are generated in the same way as in steps 5, 8, and 9 in Algorithm 3.1, respectively. The computed basis $\{\hat{q}_i\}_{i=1}^{m+1}$ form the columns of $\hat{Q}_{m+1}$, and t_{ij}'s form an $(m+1) \times m$ matrix T_m. Notice that $\hat{Q}_{m+1}$ doesn't have orthonormal columns anymore, and T_m is a banded matrix with one subdiagonal and $k-1$ superdiagonals. For example, when $k = 2$, we have the three-term recurrence for the computed basis $\hat{q}_i$, and T_m is a tridiagonal matrix. This is similar to the Chebyshev polynomial case. In the extreme case when $k = m$, the partial reorthogonalization becomes equivalent to the full orthogonalization, and all results in section 3.1 are recovered.

Similar to section 3.1, with the new basis $\{\hat{q}_j\}_{j=1}^{m+1}$ from the short-term recurrence we can rewrite (3.2) into minimizing with respect to $\hat{\alpha} \in \mathbb{C}^n$ a new objective function

$$\hat{J}(\hat{\alpha}) = \|\beta\hat{q}_1 - \hat{Q}_{m+1}T_m\hat{\alpha}\|_2^2. \tag{3.7}$$

Solving for $\hat{\alpha}$ in this problem typically needs to compute an orthogonal factorization of the matrix $\hat{Q}_{m+1}T_m$, which requires some additional computation cost and storage (since both $\hat{Q}_{m+1}$ and T_m need to be stored), compared to computing α in (3.3). However, recall that all these computations are still within a vector space of dimension n which is typically much smaller than N. Note that this bears some similarity to the minimization of the quasi residual in QMR [20, 21] and in the direct quasi GMRES (DQGMRES) method [44].

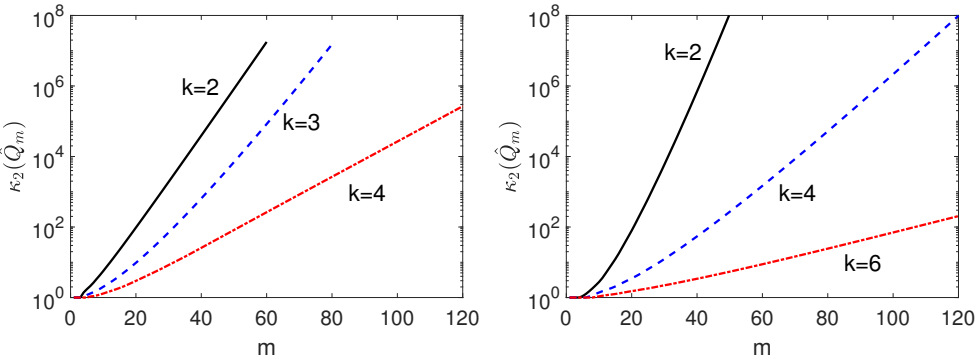
On the other hand, applying the preconditioner $M^{-1} = \hat{p}(A)$, where $\hat{p}$ is represented in the new basis $\hat{Q}_m$ is slightly different. More specifically, (3.6) is replaced by the corresponding short-term version

$$v_{i+1} = \frac{1}{t_{i+1,i}} \left(Av_i - \sum_{j=i-k+1}^{i} t_{ji} v_j \right). \tag{3.8}$$

Due to the above short-term recurrence, the application of the preconditioner $M^{-1} = \hat{p}(A)$ on a vector only requires $\mathcal{O}(mkN)$ operations and $\mathcal{O}(kN)$ storage.

Now we discuss the stability issue associated with this approach. In exact arithmetic, it is easy to see that (3.3) and (3.7) are equivalent and the polynomials $p = Q_m\alpha$ and $\hat{p} = \hat{Q}_m\hat{\alpha}$ obtained from both orthogonalization schemes are exactly the same. This is because enforcing a short-term recurrence is equivalent to a change of basis and an update to the corresponding coefficients. However, in floating-point arithmetic, $\hat{Q}_m$ becomes increasingly ill-conditioned when m increases, and thus the coefficient $\hat{\alpha}$ becomes increasingly hard to compute accurately.

Next we study the relation between the conditioning of the basis matrix $\hat{Q}_m$ and the number of recurrence terms k. Figure 3.1 shows how the 2-norm condition number $\kappa_2(\hat{Q}_m)$ grows for multiple values of k where Γ and Γ_n are drawn from the numerical examples in section 5.1 and section 5.2. For the toy problem in section 5.1, we can see in Figure 3.1a that the conditioning of the basis from short-term recurrence behaves exactly as expected: larger recurrence length k leads to more stable polynomial basis for the same size of dimension. The same effect is more significant as shown in Figure 3.1b for the Helmholtz problem in section 5.2 as the matrix becomes more ill-conditioned and indefinite. The condition numbers for $k = 2$ and 3 are almost the same so we only plotted the curve of $k = 2$; this was done similarly for $k = 4, 5$ and



(a) for problem in Section 5.1 (b) for problem in Section 5.2

FIG. 3.1. *Conditioning of $\hat{Q}_m$ generated with k-term recurrence from two examples.*

$k = 6, 7$. When the recurrence is too short, i.e., $k = 2$ or 3, the condition number rapidly grows beyond 10^8 when m passes 50. In that case, the polynomial $\hat{p}$ extracted with this basis becomes inaccurate, and the resulting preconditioner may become useless. By increasing k to 4 or 5, $\kappa_2(\hat{Q}_m)$ quickly drops from 10^8 to about 10^3 at $m = 50$. By further increasing k to 6, the same framework can admit an even larger degree for the polynomial preconditioner to still remain well conditioned.

The numerical stability of $\hat{Q}_m$ can be monitored inexpensively by its associated Gram matrix. Denote by $\hat{G}_m$ the $m \times m$ Gram matrix of the basis $\hat{Q}_m$ whose entries are defined by

$$\hat{g}_{ij} = \langle \hat{q}_i, \hat{q}_j \rangle, \quad 1 \le i, j \le m,$$

where the inner product is as defined in (3.1). The matrix $\hat{G}_m$ is Hermitian positive definite. Let $\hat{G}_m = \hat{L}_m \hat{L}_m^H$ be the Cholesky factorization where $\hat{L}_m$ is lower-triangular, and note that $\kappa_2(\hat{Q}_m) = \sqrt{\kappa_2(\hat{G}_m)} = \kappa_2(\hat{L}_m)$. As the Arnoldi-like process proceeds, both the Gram matrix $\hat{G}_{m+1}$ and the Cholesky factor $\hat{L}_{m+1}$ can be quickly updated with $\hat{G}_m$ and $\hat{L}_m$ from the previous step. When a high-degree polynomial must be used, the ill-conditioning of $\hat{Q}_m$ can be quickly detected by keeping track of the condition number of the Cholesky factor $\hat{L}_m$. The condition number can either be computed by performing an SVD on $\hat{L}_m$ when m is small or estimated cheaply by, for example, a randomized algorithm [15]. Whenever $\kappa_2(\hat{L}_m)$ goes beyond a certain threshold, we can stop the process and accept the resulting polynomial obtained at that point or restart the same process with a longer recurrence relation. As is indicated in Figure 3.1, we can start from $k = 2$ and increment k every time the process is restarted. This process is repeated until the desired degree can be reached, while $\kappa_2(\hat{L}_m)$ still remains below the given threshold. In our numerical experiments, we find that setting the threshold of the basis condition number at 10^6 usually yields good quality results.

4. Improvements based on compounding preconditioners. In the previous sections, we assumed that Γ excluded the origin. This is important because otherwise, the maximum modulus of the residual polynomial $1 - zp(z)$ would be ≥ 1 on Γ, and the resulting polynomial preconditioner would be ineffective. For ill-conditioned problems, a few eigenvalues of A will be in a small neighborhood of the origin, and the origin will be very close to Γ. In this case, a high-degree polynomial becomes mandatory if we wish to keep the maximum value of $|1 - zp(z)|$ on Γ strictly less than 1. On the other hand, the increased degree will require a longer recurrence, and this will negatively impact the efficiency of the proposed preconditioner. Similarly, we may have a situation where a few eigenvalues are far from all others, and Γ must be quite large to contain all eigenvalues, with big gaps inside that contain no eigenvalues. In this situation the resulting polynomial will also be required to be large. In this section, we will discuss two compounding techniques to overcome this difficulty.

4.1. Compounding two polynomials. Our first polynomial-compounding approach is based on compounding two low-degree polynomials to mimic the effect of a high-degree polynomial. Doing this may result in a slightly higher total number of matrix-vector multiplications associated with A, but the costs associated with vector operations and storage can be significantly reduced. Also as pointed out in [18], this strategy can also reduce the number of inner products performed.

This can be understood from a simple example. Suppose one high-degree polynomial has degree $m - 1$ and the other two low-degree polynomials have degree $m_1 - 1$

and $m_2 - 1$, respectively, with $m = m_1 \times m_2$. For these three polynomials, a k_i-term recurrence is deployed for a polynomial of degree $m_i - 1$ (for $i = 1, 2$), while a k-term recurrence is needed for degree $m-1$. Since m_1 and m_2 are both much smaller than m, the recurrence terms k_i required for stable basis for their corresponding degree $m_i - 1$ polynomial is also much smaller than that for a degree m polynomial; i.e., $k_1, k_2 \ll k$. Thus, applying the preconditioner resulting from compounding the polynomials will entail fewer vector operations and storage.

We now provide some details on how to construct these two low-degree polynomials. First find a contour Γ that encloses all the eigenvalues of A and discretize it as Γ_{n_1}. Based on Γ_{n_1}, construct the first polynomial p_1 of degree $m_1 - 1$, and select the recurrence length k_1 with the procedure discussed in section 3.4. It can be expected that most of the eigenvalues of the preconditioned matrix $A_1 := AM_1^{-1} = Ap_1(A)$ would be clustered around $z = 1$. Therefore, a second contour is then selected as a circle $\mathcal{C}$ centered at $z = 1$ with radius $\theta \in (0, 1)$. Let $\mathcal{C}_{n_2}$ be an n_2-point discretization of $\mathcal{C}$, and apply $\mathcal{C}_{n_2}$ to compute the second polynomial p_2 with degree $m_2 - 1$ and recurrence length k_2. In the end, the compound polynomial has the form $p(z) := p_1(z)p_2(zp_1(z))$, and the resulting preconditioner is $M^{-1} := p(A) = p_1(A)p_2(Ap_1(A))$.

It is clear that the preconditioner M^{-1} is a polynomial in A of degree $m_1 m_2 - 1$. Applying M^{-1} on a vector involves two main operations:

1. Apply $p_1(A)$ to a vector, which follows the formula (3.4), (3.5), and (3.8). This computation costs $m_1 - 1$ matvecs associated with A and $\mathcal{O}(m_1 k_1 N)$ operations from vector operations and $\mathcal{O}(k_1 N)$ storage.
2. Apply $p_2(Ap_1(A))$ to a vector. This operation consists of $m_2 - 1$ matvecs of $Ap_1(A)$, $\mathcal{O}(m_2 k_2 N)$ extra costs from vector operations and $\mathcal{O}(k_2 N)$ extra storage.

Table 4.1 compares the costs of applying one high-degree polynomial preconditioner versus a compound polynomial preconditioner. It is easy to see that when $m = m_1 \times m_2$, even though both preconditioners perform the same number of matvecs associated with A, the operations and peak storage associated with the compound polynomial preconditioner can be much less due to the fact that $k_1, k_2 \ll k$.

4.2. Compounding with other preconditioners. The second approach we discuss will compound the polynomial preconditioner with other types of preconditioners. For ill-conditioned problems, it is suggested to perform an approximate factorization on $A + \sigma I$ for some complex shift σ [54] instead of the original coefficient matrix A. To simplify the discussion, assume that an ILU factorization is utilized:

$$A + \sigma I \approx M_1 = LU.$$

Consider the following new right-preconditioned linear system:

$$AM_1^{-1} y = b,$$

TABLE 4.1

The cost and storage of applying the single and compound polynomial preconditioners, the single polynomial is of degree $m - 1$; the compound polynomial is built with two low-degree polynomials of degree $m_1 - 1$ and $m_2 - 1$.

	Single polynomial	Compound polynomial
Matvec of A	$m - 1$	$m_1 m_2 - 1$
Vector operations	$\mathcal{O}(mkN)$	$\mathcal{O}(m_1 m_2 k_1 N)$
Peak storage	$\mathcal{O}(kN)$	$\mathcal{O}((k_1 + k_2)N)$

where the original solution $x = M_1^{-1}y$, and apply the same procedures discussed in section 3.1 to the new coefficient matrix $A_1 := AM_1^{-1}$. Note here that the contour Γ for A_1 can be estimated by running a few steps of the Arnoldi process. After the polynomial p is constructed, the compound preconditioner takes the form of $M^{-1} = p(AM_1^{-1})$. Suppose the polynomial p is of degree $m-1$; then one application of M^{-1} on a vector consists of $m-1$ matvecs associated with A and $m-1$ applications of the preconditioner M_1^{-1}. As mentioned in section 1, ILU preconditioners may cause performance bottlenecks in a parallel environment due to difficulties in parallelizing the triangular solves. The proposed framework allows to naturally replacing the ILU factorization with more scalable preconditioners such as for Schur complement low rank (SLR), multilevel SLR or generalized MSLR preconditioners [31, 53, 14].

5. Numerical experiments. All numerical tests were run in Matlab R2020b on a desktop PC with AMD Ryzen 7 5800X CPU running at 3.8 GHz and 32 GB memory. We used GMRES with a restart dimension of 50 as the accelerator, the initial guess was set to be the zero vector, and the process was terminated when either the residual was reduced by a prescribed factor τ or the total number of iterations reached 1000. The right-hand side vector was chosen as a random vector.

In the experiments, we use the Matlab built-in function `boundary` to find an approximated boundary Γ where a subset of points from the Ritz values are taken as the vertices of a polygon, and we discretize each edge with a fixed step size $h = 0.005$ in all our tests. In some cases when an analytical approximation of the spectrum boundary is available, the boundary Γ may go through or stay very close to the origin, e.g., in section 5.2. We replace the part of Γ that is within a small radius r to the origin by the arc on the circle of the same radius r; see Figure 5.3b for an example. Then the modified Γ will be used as the new approximated spectrum boundary on which the same discretization step described above will be performed.

The following notation is used in this section:

- p-t: the preprocessing time, or time in seconds to build the preconditioner, including the time to approximate/discretize the contour Γ and build the polynomial or/and other preconditioners depending on the specific tests. The symbol "F" indicates the preconditioner could not be constructed;
- i-t: the iteration time in seconds for GMRES(50) to converge;
- its: the total number of iterations required for GMRES(50) to converge, F indicates GMRES(50) does not converge within 1000 iterations;
- mv: the total number of performed matvecs with the matrix A.

5.1. A diagonal matrix. In the first example, we generated a 2000×2000 diagonal matrix where all the diagonal entries (eigenvalues) were randomly chosen from the semiannular region $\Omega = \{z \in \mathbb{C} \,|\, 0.8 \le |z| \le 2,\, 0 \le \mathrm{Arg}(z) \le \pi\}$. The boundary of this region is shown in Figure 5.1 where the squares are the approximate eigenvalues (Ritz values) computed by running 60 steps of the Arnoldi algorithm. An approximate boundary was obtained by running the Matlab built-in function `boundary` on the approximate eigenvalues. Figure 5.1 shows that the Ritz values from the Arnoldi algorithm can characterize the boundary of the spectrum.

We first constructed polynomials of degree 29 ($m = 30$) with a recurrence length $k = 2$. Figure 5.2 shows the contour maps for the function $|1 - zp(z)|$ in log scale based on both exact (left) and approximate (right) boundaries. Since the estimated boundary approximates the exact one very well, the two maps look almost identical. Table 5.1 tabulates the numerical results for solving the linear system with these constructed polynomial preconditioners; the tolerance was set at $\tau = 10^{-12}$. It took

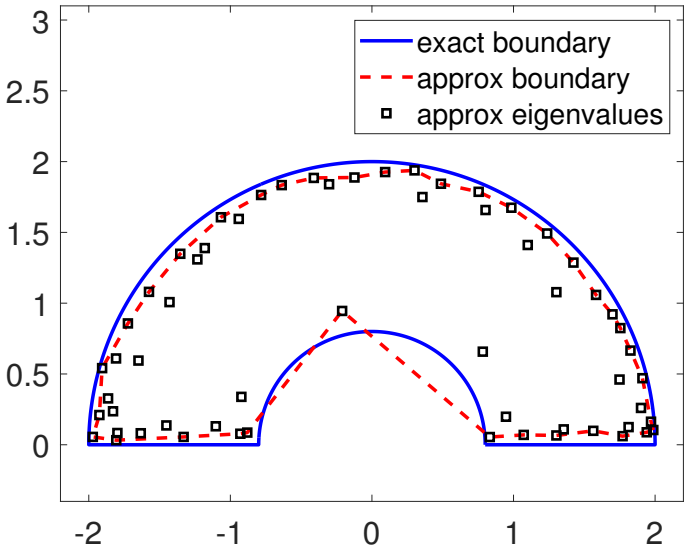
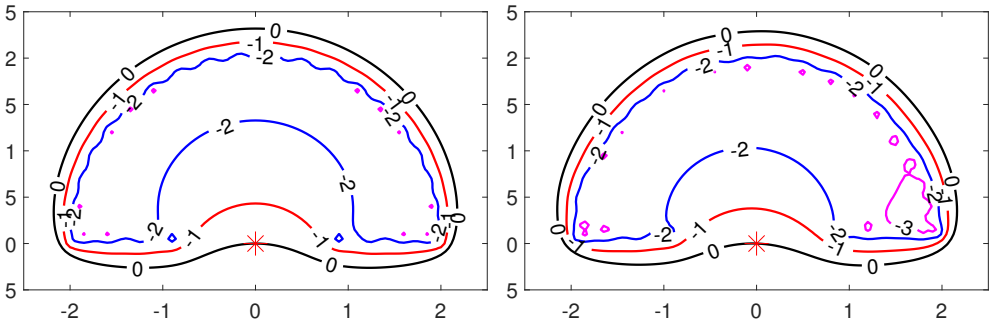


FIG. 5.1. *The exact and approximate boundaries of the spectrum and the approximate eigenvalues obtained from* 60 *steps of the Arnoldi algorithm for the* 2000×2000 *diagonal matrix in section* 5.1.



(a) using the true boundary (b) using the approximate boundary

FIG. 5.2. *Contour maps of* $|1 - zp(z)|$ *in log scale with different choice of* Γ *for the* 2000×2000 *diagonal matrix in section* 5.1. *The asterisk marks the origin.*

TABLE 5.1
Convergence results of GMRES(50) for the 2000×2000 *diagonal matrix test in section* 5.1 *with tolerance* $\tau = 10^{-12}$.

		p-t	i-t	its	mv
No precond.		\	0.6525	237	237
With precond.	Exact boundary	0.0045	0.5366	8	240
	Approx. boundary	0.0040	0.5238	8	240

237 iterations for GMRES(50) to converge without any preconditioner. On the other hand, GMRES(50) with the polynomial preconditioners converged in 8 iterations in both cases. Although the preconditioned methods performed 3 more matvecs, they actually took less time to converge. Similar observations can also be made in other

examples in this section. This is due to the fact that a reduced iteration number leads to a much smaller subspace for GMRES and far fewer inner products during the computation. This performance gap can be expected to become more pronounced when running the experiments on high-performance computing architectures.

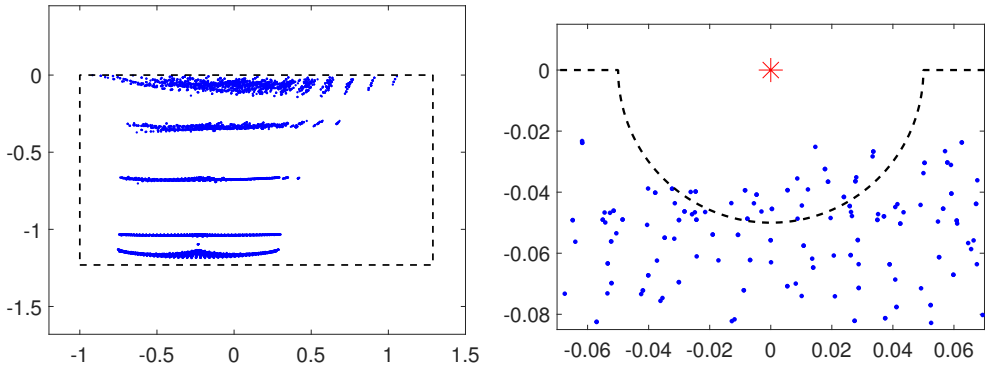
We also want to emphasize that the preconditioner construction time was only a tiny fraction of the iteration time. This is because the number of discretization points for both the exact and approximate boundaries and the corresponding polynomial space has much smaller dimension compared to the matrix size $N = 2000$.

5.2. Helmholtz problem. The second example is the three-dimensional Helmholtz equation

$$-\Delta u - \frac{\omega^2}{c^2(x)} u = s,$$

where ω is the angular frequency and $c(x)$ is the wavespeed. Here we chose the wavesped function $c(x)$ with eight high-wavespeed anomolies as considered in Figure 7.1 of [32]. The computational domain was the unit cube, and the equation was discretized with 7-point stencil finite difference method. Perfectly matched layer (PML) boundary conditions were imposed to reduce the artificial reflections near the boundaries of the computational domain. We chose the minimum sampling rate as 9 outside the anomolies and 18 inside the anomolies similar to [32]. The resulting linear system is sparse complex symmetric with dimension $N = N_x \times N_y \times N_z$. Moreover, the spectrum of the matrix is contained in a rectangle area $\{z \in \mathbb{C} \,|\, \text{real}(z) \in [-1, \rho_1 - 1], \text{imag}(z) \in [-\rho_2, 0]\}$, where the two parameters ρ_1 and ρ_2 are given in [32, Lemma 3.1]. Figure 5.3a shows all the eigenvalues and the rectangular boundary from [32, Lemma 3.1] for a discretized Helmholtz operator of size $N = 20^3$. A zoom-in view of Γ near the origin is shown in Figure 5.3b. We fixed the tolerance at $\tau = 10^{-3}$ for the Helmholtz equation tests in this section. Note that this test matrix is nonnormal.

5.2.1. Compounding polynomial preconditioners. Compared with the toy example of the first test example, this problem is much harder to solve. First, there are many eigenvalues near the origin. Second, the theoretical spectrum boundary (the rectangular area) overlaps with the origin. In order to construct an effective polynomial preconditioner, we have to



(a) Eigenvalues and the rectangle boundary (b) Modified Γ near the origin

FIG. 5.3. *The theoretical rectangular spectrum boundary from* [32, Lemma 3.1] *and the zoomedin view of the modified* Γ *near the origin for a discretized Helmholtz operator of size* $N = 20^3$.

TABLE 5.2
Convergence results of various preconditioned GMRES(50) on the Helmholtz equation test with size $N = 100^3$, the tolerance is fixed at $\tau = 10^{-3}$.

Preconditioner type	p-t	i-t	its	mv
No preconditioner	\	\	F	\
ILUT	F	\	\	\
ILUT with diagonal shift $\sigma = -0.4i$	220.96	\	F	\
Single polynomial of degree $600 - 1$	3.49	1484.87	16	9600
Compound polynomial of degree $60 \times 10 - 1$	0.05	906.41	18	10,800

1. modify the theoretical rectangular contour [32, Lemma 3.1] to exclude the origin;
2. use a high-degree polynomial.

For this test, the problem size was $N = 100^3 = 1,000,000$, and the boundary Γ was the same as shown in Figure 5.3b. We compared the performance of two polynomial preconditioners on this test matrix. The first one was a single polynomial of degree 599 ($m = 600$). In order to ensure numerical stability, we used a recurrence length of $k = 10$. The second one was a compound polynomial with $m_1 = 60$ and $m_2 = 10$ so that $m_1 \times m_2 = m$. Since m_1 and m_2 are relatively small, the recurrence lengths were set to be $k_1 = k_2 = 2$. The convergence results with these two preconditioners are shown in Table 5.2. In addition, we also show threshold-based ILU (ILUT) preconditioners on A and $A + \sigma I$ with a complex diagonal shift $\sigma = -0.4i$ for comparisons.

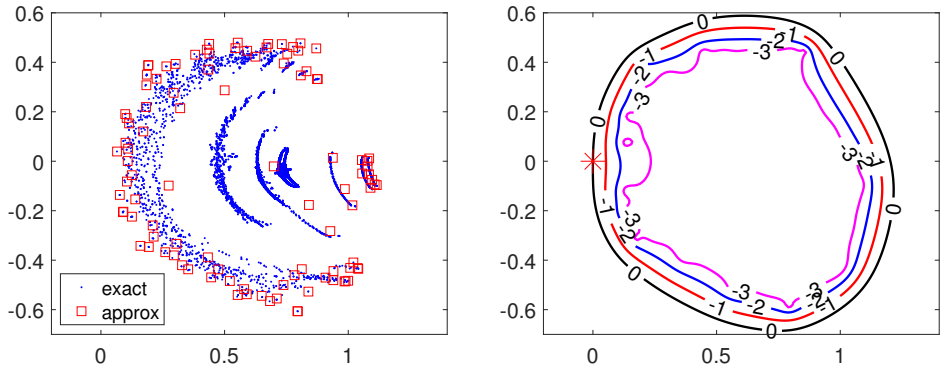
Due to the ill-conditioning and indefiniteness of the test matrix, the first three methods in Table 5.2 failed to converge. In particular, ILUT even failed to finish the factorization. On the other hand, both polynomial preconditioned methods converged within 18 iterations. The compound polynomial preconditioner took much less time to construct than the single polynomial preconditioner, and it also reduced the iteration time by more than a half, even though 1200 more matvecs with A were performed.

5.2.2. Compounding with SLR. We also compounded the polynomial preconditioner with the nonsymmetric SLR preconditioner [31] and tested its preconditioning effect on the Helmholtz problem. The problem size was still kept at $N = 100^3$, and the tolerance was set at $\tau = 10^{-3}$. We applied the SLR preconditioner to the shifted system $M_1 \approx A + \sigma I$ with a complex shift $\sigma = -0.4i$ (pulling the eigenvalues away from the origin) and then chose a polynomial preconditioner of degree 29 for the matrix $A_1 = AM_1^{-1}$. The approximate spectrum boundary of A_1 was obtained by running 80 steps of the Arnoldi process; then it was uniformly discretized with $h = 0.005$ which resulted in 727 discretization points. The convergence results as well as the comparison with SLR preconditioner are shown in Table 5.3. Note that the diagonal shift $\sigma = -0.4i$ is the same as one of the ILUT tests in Table 5.2. We see that the SLR preconditioned GMRES(50) failed to converge in 1000 iterations while the SLR compound polynomial preconditioner converged in only 29 iterations and required the least iteration time among all seven methods tested in Table 5.2 and Table 5.3. Also notice that the construction time of this compound preconditioner is higher than other methods because it includes both the SLR preconditioner construction time and the time to perform 80 steps of the Arnoldi process.

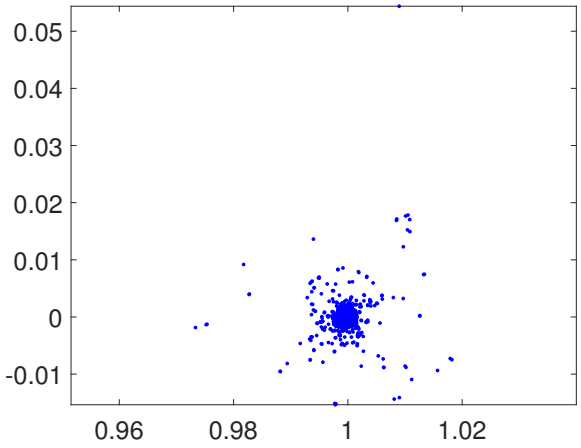
In order to visualize the spectrum of the preconditioned matrix across different stages with this compound preconditioner, we also ran the experiment on a smaller Helmholtz problem of size $N = 20^3$ so that we were able to compute all eigenvalues

TABLE 5.3
Convergence results of the SLR and SLR compound polynomial preconditioned GMRES(50) on the Helmholtz equation test with size $N = 100^3$; the tolerance was fixed at $\tau = 10^{-3}$.

Preconditioner type	p-t	i-t	its	mv
SLR preconditioner	86.94	\	F	\
SLR with polynomial of degree $30-1$	145.85	308.03	29	870



(a) Exact and approximate eigenvalues of AM_1^{-1} (b) Contour map of $|1 - zp(z)|$ in log scale



(c) Eigenvalues of the final preconditioned matrix $A_1 p(A_1)$

FIG. 5.4. *Illustration of the preconditioning effect of both stages of the SLR compound preconditioner $p(AM_1^{-1})$ on a small discretized Helmholtz equation test of size $N = 20^3$, where M_1 denotes the SLR preconditioner and p has degree 29.*

of the matrices. Let M_1 denote the SLR preconditioner for the discretized Helmholtz operator A. The spectrum of $A_1 = AM_1^{-1}$ as well as the approximate eigenvalues from running 80 steps of the Arnoldi algorithm are shown in Figure 5.4a. A polynomial preconditioner $p(A_1)$ of degree 29 was constructed, and the contour map of the corresponding residual polynomial $|1-zp(z)|$ is drawn in Figure 5.4b. Compared with Figure 5.3b, it is easy to see that the SLR preconditioner pushed the eigenvalues of A_1 further away from the origin. Thus, a low-degree polynomial of p has already led to an efficient preconditioner, which is supported by both the contour map Figure 5.4b as well as the spectrum of the preconditioned matrix $A_1 p(A_1)$ in Figure 5.4c.

TABLE 5.4
Information on the general test matrices from the SuiteSparse collection: N is the size of the matrix and nnz the number of nonzero elements.

Group/matrix name	N	nnz	Origin
Rajat/rajat09	24,482	105,573	circuit simulation
Dehghani/light_in_tissue	29,282	406,084	electromagnetics
Goodwin/Goodwin_127	178,437	5,778,545	CFD problem
Kim/kim2	456,976	11,330,020	3D problem
Bourchtein/atmosmodd	1,270,432	8,814,880	CFD problem

TABLE 5.5
Convergence results of general sparse matrices with GMRES(50) and tolerance $\tau = 10^{-10}$, all polynomials were of degree 39; column n shows the number of discretization points used on the spectrum boundary of A_1.

Matrix	ILUT				ILUT compound polynomial				
	p-t	i-t	its	mv	n	p-t	i-t	its	mv
Rajat09	F	\	\	\	771	0.21	1.57	46	1,840
Light_in_tissue	0.055	1.23	213	213	572	0.31	0.33	6	240
Goodwin_127	F	\	\	\	462	1.27	171.97	261	10,440
Kim2	3.50	6.98	38	38	1488	10.97	4.58	2	80
Atmosmodd	0.83	114.90	397	397	568	23.30	26.35	6	240

5.3. General sparse matrices. We also tested the ILUT compound polynomial preconditioner on several general sparse matrices obtained from the SuiteSparse Matrix Collection.[1] All of the test problems are nonsymmetric real or non-Hermitian complex. After the ILUT preconditioner M_1 was constructed, we ran 60 steps of the Arnoldi algorithm to obtain the approximate eigenvalues; then we built the approximate spectrum boundary, a polygon uniformly discretized on each of its continuous parts with step size $h = 0.005$. All polynomials were of degree 39 ($m = 40$) with a recurrence of length $k = 2$. The tolerance for GMRES(50) was set at $\tau = 10^{-10}$. Some information about these matrices is provided in Table 5.4. Convergence results are shown in Table 5.5 together with those from ILUT alone for comparison. Note that the preconditioning set-up time for the ILUT compound polynomial preconditioner includes time for ILUT preconditioner construction, 60 steps of Arnoldi algorithm on $A_1 = AM_1^{-1}$, and time for building the polynomial. Despite a slightly more expensive construction costs, ILUT compound polynomial preconditioner outperformed ILUT on all of these 5 tests in the iteration phase.

6. Conclusions. The primary distinction between the polynomial preconditioning techniques introduced in this paper and existing techniques is the emphasis on controlling the numerical stability of the polynomial construction and the resulting iterative process. This is important because, contrary to a common misperception, polynomial *preconditioners are appealing when used with relatively high degrees.* High-degree polynomials result in good quality preconditioners that will yield convergence in a smaller number of outer iterations. We showed how to significantly improve the performance of a basic polynomial preconditioning method by a process that relies on a short-term recurrence as well as by a strategy of compounding two consecutive preconditioners. It is clear that a big appeal of the proposed methods is their potential for producing big gains in speed when used in highly parallel and distributed environments such as massive clusters of many computing nodes with multicore CPUs

[1] SuiteSparse Matrix Collection: https://sparse.tamu.edu.

and GPUs/accelerators. The numerical experiments show that even in a scalar environment, these methods can be effective in solving difficult problems in situations where classical techniques fail.

REFERENCES

[1] L. M. Adams and E. G. Ong, *Additive polynomial preconditioners for parallel computers*, Parallel Comput., 9 (1989), pp. 333–345.

[2] W. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[3] S. F. Ashby, *Minimax polynomial preconditioning for Hermitian linear systems*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 766–789.

[4] S. F. Ashby, T. A. Manteuffel, and J. S. Otto, *A comparison of adaptive Chebyshev and least-squares polynomial preconditioning for Hermitian positive definite linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1–29.

[5] S. F. Ashby, T. A. Manteuffel, and P. E. Saylor, *Adaptive polynomial preconditioning for Hermitian indefinite linear systems*, BIT, 29 (1989), pp. 583–609.

[6] M. M. Baskaran and R. Bordawekar, *Optimizing Sparse Matrix-Vector Multiplication on GPUs Using Compile-Time and Run-Time Strategies*. Technical report, 2008.

[7] N. Bell and M. Garland, *Implementing sparse matrix-vector multiplication on throughput-oriented processors*, in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, 2009, pp. 1–11.

[8] E. W. Cheney, *Introduction to Approximation Theory*, AMS, Providence, Rhode Island, 1982.

[9] E. Chow and A. Patel, *Fine-grained parallel incomplete lu factorization*, SIAM J. Sci. Comput., 37 (2015), pp. C169–C193.

[10] J. P. Coleman, *Polynomial approximations in the complex plane*, J. Comput. Appl. Math., 18 (1987), pp. 193–211.

[11] M. Crouzeix, *Bounds for analytic functions of matrices*, Integral Equations Operator Theory, 48 (2004), pp. 461–477, https://doi.org/10.1007/s00020-002-1188-6.

[12] M. Crouzeix, *Numerical range and functional calculus in Hilbert space*, J. Funct. Anal., 244 (2007), pp. 668–690.

[13] M. Crouzeix and C. Palencia, *The numerical range is a* $(1+\sqrt{2})$*-spectral set*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 649–655.

[14] G. Dillon, V. Kalantzis, Y. Xi, and Y. Saad, *A hierarchical low rank Schur complement preconditioner for indefinite linear systems*, SIAM J. Sci. Comput., 40 (2018), pp. A2234–A2252.

[15] J. D. Dixon, *Estimating extremal eiganvalues and condition numbers of matrices*, SIAM J. Numer. Anal., 20 (1983), pp. 812–814.

[16] S. C. Eisenstat, J. M. Ortega, and C. T. Vaughan, *Efficient polynomial preconditioning for the conjugate gradient method*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 859–872.

[17] H. C. Elman, Y. Saad, and P. E. Saylor, *A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 840–855.

[18] M. Embree, J. A. Loe, and R. B. Morgan, *Polynomial Preconditioned Arnoldi*, preprint, arxn:1806.08020, 2018, https://arxiv.org/abs/1806.08020.

[19] R. Freund, *On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices*, Numer. Math., 57 (1990), pp. 285–312.

[20] R. W. Freund and N. M. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[21] R. W. Freund and N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.

[22] G. H. Golub and R. S. Varga, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods*, Numer. Math., 3 (1961), pp. 147–156.

[23] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.

[24] M. H. Gutknecht and S. Röllin, *The Chebyshev iteration revisited*, Parallel Comput., 28 (2002), pp. 263–283.

[25] O. G. Johnson, C. A. Micheli, and G. Paul, *Polynomial preconditioners for conjugate gradient calculations*, SIAM J. Numer. Anal., 20 (1983), pp. 362–376.

[26] C. Lanczos, *Trigonometric interpolation of empirical and analytical functions*, J. Math. Phys., 17 (1938), pp. 123–199.
[27] D. P. Laurie and L. M. Venter, *A two-phase algorithm for the Chebyshev solution of complex linear equations*, SIAM J. Sci. Comput., 15 (1994), pp. 1440–1451.
[28] R. Li and Y. Saad, *Divide and conquer low-rank preconditioners for symmetric matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A2069–A2095.
[29] R. Li and Y. Saad, *GPU-accelerated preconditioned iterative linear solvers*, J. Supercomput., 63 (2013), pp. 443–466.
[30] R. Li, Y. Xi, L. Erlandson, and Y. Saad, *The eigenvalues slicing library (EVSL): Algorithms, implementation, and software*, SIAM J. Sci. Comput., 41 (2019), pp. C393–C415, https://doi.org/10.1137/18M1170935.
[31] R. Li, Y. Xi, and Y. Saad, *Schur complement-based domain decomposition preconditioners with low-rank corrections*, Numer. Linear Algebra Appl., 23 (2016), pp. 706–729.
[32] X. Liu, Y. Xi, Y. Saad, and M. V. de Hoop, *Solving the 3D high-frequency Helmholtz equation using contour integration and polynomial preconditioning*, preprint, arxiv:1811.12378, 2018, https://arxiv.org/abs/1811.12378.
[33] T. A. Manteuffel, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327.
[34] T. A. Manteuffel, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 183–208.
[35] T. A. Manteuffel and G. Starke, *On hybrid iterative methods for nonsymmetric systems of linear equations*, Numer. Math., 73 (1996), pp. 489–506.
[36] N. M. Nachtigal, L. Reichel, and L. N. Trefethen, *A hybrid GMRES algorithm for nonsymmetric linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 796–825.
[37] R. Pachón and L. N. Trefethen, *Barycentric-Remez algorithms for best polynomial approximation in the chebfun system*, BIT, 49 (2009), pp. 721–741.
[38] L. Reichel, *The application of Leja points to Richardson iteration and polynomial preconditioning*, Linear Algebra Appl., 154–156 (1991), pp. 389–414.
[39] Y. Saad, *Practical use of polynomial preconditionings for the conjugate gradient method*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 865–881.
[40] Y. Saad, *On the condition numbers of modified moment matrices arising in least squares approximation in the complex plane*, Numer. Math., 48 (1986), pp. 337–347.
[41] Y. Saad, *Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems*, SIAM J. Numer. Anal., 24 (1987), pp. 155–169.
[42] Y. Saad, *Iterative methods for sparse linear systems*, 2nd ed., SIAM, Philadelphia, 2003.
[43] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[44] Y. Saad and K. Wu, *DQGMRES: A direct quasi-minimal residual algorithm based on incomplete orthogonalization*, Numer. Linear Algebra Appl., 3 (1996), pp. 329–343.
[45] J. Shi, R. Li, Y. Xi, Y. Saad, and M. V. de Hoop, *Computing planetary interior normal modes with a highly parallel polynomial filtering eigensolver*, in SC18: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, IEEE, 2018, pp. 894–906.
[46] E. M. Stein and R. Shakarchi, *Complex Analysis, Princeton Lectures in Analysis,* II, Princeton University Press, Princeton, NJ, 2003.
[47] R. L. Streit, *Solution of systems of complex linear equations in the l_∞ norm with constraints on the unknowns*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 132–149.
[48] R. L. Streit and A. H. Nuttall, *A note on the semi-infinite programming approach to complex approximation*, Math. Comp., 40 (1983), pp. 599–605.
[49] P. T. P. Tang, *A fast algorithm for linear complex Chebyshev approximations*, Math. Comp., 51 (1988), pp. 721–739.
[50] H. K. Thornquist, *Fixed-polynomial approximate spectral transformations for preconditioning the eigenvalue problem*, Ph.D. thesis, Rice University, Houston, TX, 2006.
[51] F. Vázquez, E. M. Garzón, J. A. Martínez, and J. J. Fernández, *The sparse matrix vector product on GPUs*, technical report, 2009.
[52] G. A. Watson, *A method for the Chebyshev solution of an overdetermined system of complex linear equations*, IMA J. Numer. Anal., 8 (1988), pp. 461–471.
[53] Y. Xi, R. Li, and Y. Saad, *An algebraic multilevel preconditioner with low-rank corrections for sparse symmetric matrices*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 235–259.
[54] Y. Xi and Y. Saad, *A rational function preconditioner for indefinite sparse linear systems*, SIAM J. Sci. Comput., 39 (2017), pp. A1145–A1167.