POLYNOMIAL PRECONDITIONED ARNOLDI *

MARK EMBREE†, JENNIFER A. LOE‡, AND RONALD B. MORGAN‡

Abstract. Polynomial preconditioning can improve the convergence of the Arnoldi method for computing eigenvalues. Such preconditioning significantly reduces the cost of orthogonalization; for difficult problems, it can also reduce the number of matrix-vector products. Parallel computations can particularly benefit from the reduction of communication-intensive operations. The GMRES algorithm provides a simple and effective way of generating the preconditioning polynomial. For some problems high degree polynomials are especially effective, but they can lead to stability problems that must be mitigated. A two-level "double polynomial preconditioning" strategy provides an effective way to generate high-degree preconditioners.

Key words. eigenvalues, polynomial preconditioning, Arnoldi, GMRES

AMS subject classifications. 65F15, 15A18

1. Introduction.

We seek eigenvalues and eigenvectors of a large (possibly non-symmetric) matrix A. The restarted Arnoldi algorithm [22, 28] (invoked by MATLAB's `eigs` command) is a standard workhorse for such problems, but for some matrices convergence is slow. On can improve convergence via a shift-invert transformation, i.e., applying the algorithm to $(A-\mu I)^{-1}$ to find eigenvalues near $\mu \in \mathbb{C}$. Here we investigate an effective alternative that does not need any explicit inversion of A. The *polynomial preconditioned Arnoldi method* is fairly simple to implement and can accelerate convergence for difficult problems.

When applied to the matrix A and starting vector v, the Arnoldi algorithm approximates eigenvalues using Rayleigh–Ritz estimates from the Krylov subspace

$$\mathcal{K}_m(A,v) \equiv \operatorname{span}\{v, Av, \ldots, A^{m-1}v\}. \tag{1.1}$$

Any vector x in this space, including the approximate eigenvectors, can be written in the form $x = \omega(A)v$ for some $\omega \in \mathcal{P}_{m-1}$, where $\mathcal{P}_s$ denotes the polynomials of degree s or less. The Arnoldi process builds an orthonormal basis for the subspace (1.1) via a Gram–Schmidt process, requiring many inner products as m grows.

Polynomial preconditioning methods [11, 12, 21, 23, 24, 26, 31] apply the Arnoldi algorithm to the matrix $\pi(A)$, for some polynomial $\pi \in \mathcal{P}_d$. Now eigenvalue estimates are drawn from the Krylov subspace

$$\mathcal{K}_m(\pi(A),v) = \operatorname{span}\{v, \pi(A)v, \ldots, \pi(A)^{m-1}v\}, \tag{1.2}$$

a subspace of $\mathcal{K}_{d(m-1)+1}(A,v)$. The large subspace $\mathcal{K}_{d(m-1)+1}(A,v)$ contains better approximations to the desired eigenvectors of A than does $\mathcal{K}_m(A,v)$. A polynomial preconditioner π is effective if the low-dimensional subspace $\mathcal{K}_m(\pi(A),v) \subseteq \mathcal{K}_{d(m-1)+1}(A,v)$ contains such improved estimates of the desired eigenvectors.

More specifically, any $x \in \mathcal{K}_m(\pi(A),v)$ can be written as $x = \omega(\pi(A))v$, where $\omega \in \mathcal{P}_{m-1}$. Since $\omega \circ \pi \in \mathcal{P}_{d(m-1)}$, polynomial preconditioning leads to eigenvector

*The first author was supported by NSF grant DMS-1720257. The third author was supported by NSF grant DMS-1418677.

†Department of Mathematics and Computational Modeling and Data Analytics Division, Academy of Integrated Science, Virginia Tech, Blacksburg, VA 24061 (embree@vt.edu).

‡Department of Mathematics, Baylor University, Waco, TX 76798-7328 (Jennifer_Loe@baylor.edu, Ronald_Morgan@baylor.edu).

estimates that are high-degree polynomials in A. This feature comes at a cost, since $\pi(A)$ must be applied to a vector each time the subspace dimension m is increased. In typical high-performance computing environments these matrix-vector products can be evaluated more efficiently than inner products, which require significant communication and synchronization. Thus the subspace (1.2) can be constructed much more efficiently (in terms of both work and storage) than building out a standard Krylov subspace (1.1) of dimension $d(m-1)+1$. In summary, polynomial preconditioning gives an efficient way to involve high-degree polynomials while controlling the dimension of the subspace and limiting the cost of orthogonalization.

What is a good choice for the preconditioning polynomial π? Section 2 describes one choice for π that is inspired by the GMRES algorithm. By the spectral mapping theorem, every eigenvalue λ of A is mapped to the eigenvalue $\pi(\lambda)$ of $\pi(A)$. As the convergence theory in section 3 illustrates, effective preconditioners separate the desired eigenvalues from the undesired ones.

Polynomial preconditioning is a special kind of *spectral transformation*, in which the Arnoldi algorithm is applied to $f(A)$ for some function f that maps the desired eigenvalues of A to the largest magnitude eigenvalues of $f(A)$; see, e.g., [14]. Typically such transformations involve a matrix inverse. One might seek a polynomial preconditioner π that mimics a more complicated $f(A)$. For example, Thornquist [32] advocates $\pi(A) \approx (A-\mu I)^{-1}$. Here we do not seek π that approximates $(A-\mu I)^{-1}$, but merely one that distances the desired eigenvalues from the rest of the spectrum.

Although methods for polynomial preconditioning have been proposed in the past, they are not generally used in practice. *To become popular, a polynomial preconditioner must be both effective and easy to implement.* We shall also explore stability, an important consideration for practical algorithms.

Section 2 discusses the choice of the GMRES (minimum residual) polynomial for the preconditioning, followed by some convergence theory in Section 3. Numerical experiments begin in Section 4, and suggest several practical issues that a robust algorithm should address. Section 5 studies the sensitivity of π to the choice of GMRES starting vector, while Section 6 shows how to adjust the starting vector to make it more likely for Arnoldi to find the desired eigenvalues. Section 7 addresses numerical stability, suggesting the addition of duplicate roots in π to better control distant unwanted eigenvalues. Finally, Section 8 describes *double polynomial preconditioning*, which enables the use of very large degree polynomials.

2. Minimum Residual Polynomials. We seek the eigenvalues of A nearest the origin.[1] For the polynomial preconditioner π we use the minimum residual polynomial [13, 18] that arises when solving the linear system $Ax = b$ using the GMRES algorithm [27] (or MINRES for symmetric A [20]). This polynomial satisfies

$$\|\pi(A)b\|_2 = \min_{\substack{p\in\mathcal{P}_d \\ p(0)=1}} \|p(A)b\|_2,$$

and hence $\pi \in \mathcal{P}_d$ must satisfy $\pi(0) = 1$; $|\pi(z)|$ will generally be small over the spectrum of A.[2] Denote the eigenvalues of A as $\sigma(A) = \{\lambda_j\}$, so the eigenvalues

[1] If one seeks eigenvalues near $\mu \in \mathbb{C}$, replace A with $A - \mu I$. If μ is in the interior of the spectrum, note that $|\pi(z)|$ is less likely to attain its maximum over the spectrum at μ. The challenge of computing interior eigenvalues arises in Example 2, and is the subject of future work.

[2] This form allows one to write $\pi(z) = 1 - z\varphi(z)$ for some $\varphi \in \mathcal{P}_{d-1}$. Then $0 \approx \pi(A)b = (I - A\varphi(A))b$ suggests that $\varphi(A) \approx A^{-1}$. Thornquist uses this φ as the polynomial preconditioner, replacing A with $A - \mu B$ for generalized eigenvalue problems [32].

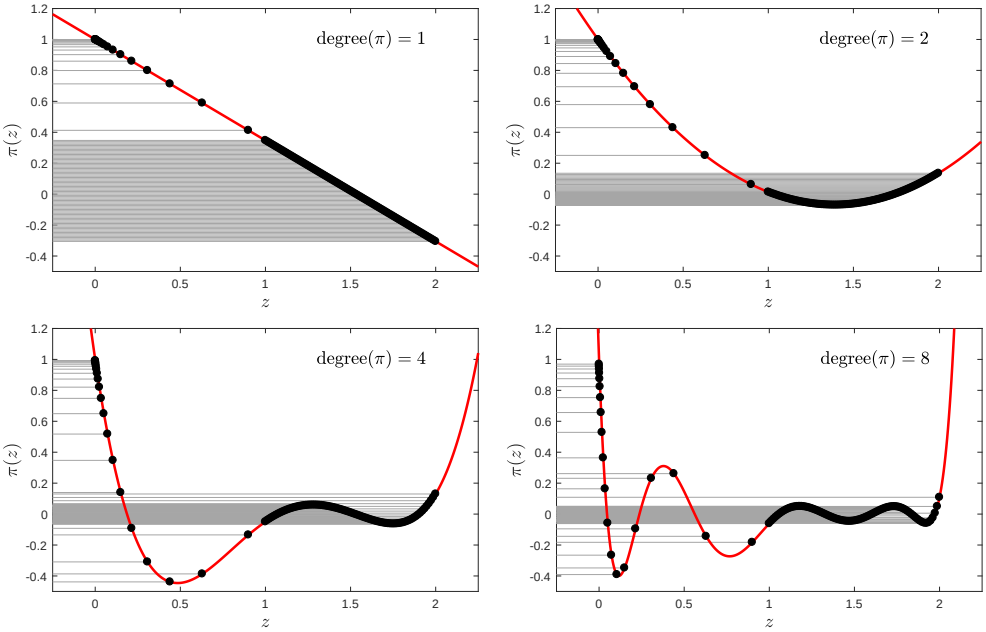


FIG. 2.1. *GMRES polynomials (red lines) tend to separate eigenvalues closest to the origin, while mapping large-magnitude eigenvalues of A close to zero. The black dots and horizontal gray lines show the values of $\pi(\lambda_j)$.*

of $\pi(A)$ are $\pi(\lambda_j)$. If GMRES converges quickly, then $\|\pi(A)b\|$ is small, and the eigenvalues of $\pi(A)$ are typically concentrated near 0. However, the condition $\pi(0) = 1$ means that π is generally not able to map the small eigenvalues of A as close to zero, making these eigenvalues better separated in the spectrum of $\pi(A)$. Figure 2.1 illustrates this point, using a symmetric A with 20 eigenvalues logarithmically spaced in the interval $[10^{-3}, 0.9]$ and 80 eigenvalues uniformly spaced in the interval $[1, 2]$; we seek a few of the smallest eigenvalues. Figure 2.1 shows $\pi(z)$ for degree $d = 1, 2, 4$ and 8 (red lines) and the values of $\pi(\lambda_j)$ (black dots and gray lines). Since the small eigenvalues of A are near the origin (where $\pi(0) = 1$), $\pi(\lambda_j)$ is large for these values, while $\pi(\lambda_j)$ is small for the larger eigenvalues. Moreover, π *separates the tightly clustered eigenvalues near the origin.* The desired eigenvalues of $\pi(A)$ (nearest 1) will be easier for Arnoldi to compute than the corresponding (smallest) eigenvalues of A. Figure 2.1 also hints at a complication: when the degree is large, the map π entangles some of the larger eigenvalues from the interval $[10^{-3}, 0.9]$ with those from $[1, 2]$.

The GMRES polynomial π is easy to construct in factored form. To find its roots, run a cycle of GMRES(d) and compute the harmonic Ritz values [8, 15, 19] (reciprocals of Rayleigh–Ritz eigenvalue estimates for A^{-1} from $A\mathcal{K}_d(A, b)$). For numerical stability, label the roots using the modified Leja ordering [1, alg. 3.1], giving

$$\pi(z) = \prod_{i=1}^{d} \left(1 - \frac{z}{\theta_i}\right). \tag{2.1}$$

A quick listing of the algorithm follows.

Polynomial Preconditioned Arnoldi,
with GMRES polynomial of degree d

1. **Construction of the polynomial preconditioner, π:**
   (a) Run one cycle of GMRES(d).
   (b) Find the harmonic Ritz values, $\theta_1, \ldots, \theta_d$, which are the roots of the GMRES polynomial: with Arnoldi decomposition $AV_d = V_{d+1}H_{d+1,d}$, find the eigenvalues of $H_{d,d} + h_{d+1,d}^2 f e_d^T$, where $f = H_d^{-*} e_d$ with elementary coordinate vector $e_d = [0, \ldots, 0, 1]^T$.
   (c) Order the GMRES roots with modified Leja ordering [1, alg. 3.1]. (To avoid overflow or underflow for high degree polynomials, one can replace products of absolute values of differences of roots with addition of logarithms of these quantities.)
2. **PP-Arnoldi:** Apply restarted Arnoldi to the matrix $\pi(A) = \Pi_{i=1}^{d}(I - A/\theta_i)$. (The experiments in Sections 4–8 use a thick-restarted Arnoldi method with exact shifts [16, 17, 29, 34].)

3. Convergence theory for polynomial preconditioning. Let $\sigma(A)$ denote the spectrum of A. We seek some subset $\Sigma := \{\lambda_1, \ldots, \lambda_k\}$ of $\sigma(A)$, typically those of smallest magnitude.) The eigenvalues in Σ should be listed with the their full multiplicity, but we presume that the eigenvalues in Σ are *nonderogatory* (i.e., there exists only one linearly independent eigenvector for each distinct eigenvalue in Σ); this assumption is standard for Krylov subspace convergence theory, required because of the concept of *reachable invariant subspaces* [2, 3].

How does the preconditioned Arnoldi algorithm converge as the dimension of the Krylov subspace increases? Especially for nonsymmetric A, the error in the eigenvalue estimates can converge quite irregularly. Thus we prefer to analyze the rate at which the Krylov subspace in the Arnoldi method "captures" the invariant subspace (span of eigenvectors and generalized eigenvectors) associated with Σ.

Let us make this notion precise. Let $\mathcal{U}$ denote the maximal invariant subspace associated with the desired eigenvalues Σ. Since none of these eigenvalues are derogatory, $\dim(\mathcal{U}) = k$. The Arnoldi algorithm approximates $\mathcal{U}$ by some Krylov subspace $\mathcal{V}$, whose dimension will generally differ from k. The *containment gap* (or just *gap*) between $\mathcal{U}$ and $\mathcal{V}$,

$$\delta(\mathcal{U}, \mathcal{V}) \equiv \max_{u \in \mathcal{U}} \min_{v \in \mathcal{V}} \frac{\|u - v\|}{\|u\|}, \tag{3.1}$$

measures the sine of the largest canonical angle between $\mathcal{U}$ and its best k-dimensional approximation from $\mathcal{V}$. Note that $\delta(\mathcal{U}, \mathcal{V}) \in [0, 1]$, with $\delta(\mathcal{U}, \mathcal{V}) = 1$ when $\dim(\mathcal{V}) < \dim(\mathcal{U}) = k$: $\mathcal{V}$ must have dimension at least k in order to approximate all of $\mathcal{U}$. For additional properties of the gap, see [2, 4, 10].

For the polynomially preconditioned Arnoldi algorithm, the Krylov subspace $\mathcal{K}_m(\pi(A), v)$ plays the role of $\mathcal{V}$. How does $\delta(\mathcal{U}, \mathcal{K}_m(\pi(A), v))$ depend on the choice of π, the dimension m, and the starting vector v? We shall apply the convergence theory from [3], which uses the following notation and assumptions.

(a) Label the desired eigenvalues as $\lambda_1, \ldots, \lambda_k$, allowing multiplicities. Assume none of these eigenvalues are derogatory, and these eigenvalues are disjoint from the rest of the spectrum $\{\lambda_{k+1}, \ldots, \lambda_n\}$.
(b) $\mathcal{U}$ denotes the m-dimensional invariant subspace associated with $\lambda_1, \ldots, \lambda_k$.

(c) P_g denotes the spectral projector onto $\mathcal{U}$, so that $P_b \equiv I - P_g$ is the spectral projector onto the invariant subspace associated with $\lambda_{k+1}, \ldots, \lambda_n$.
(d) Ω_b is a compact subset of $\mathbb{C}$ that contains the eigenvalues $\lambda_{k+1}, \ldots, \lambda_n$.
(e) $\alpha_g(z) \equiv (z - \lambda_1) \cdots (z - \lambda_k)$ is the component of the minimal polynomial of A associated with the desired eigenvalues.

For the sake of comparison, we start with Theorem 3.3 of [3], which applies to the standard Arnoldi method for (A, v). For a Krylov subspace of dimension $m \geq 2k$, the theorem gives

$$\delta(\mathcal{U}, \mathcal{K}_m(A, v)) \leq \left(\max_{\psi \in \mathcal{P}_{k-1}} \frac{\|\psi(A) P_b v\|}{\|\psi(A) P_g v\|} \right) \kappa(\Omega_b) \min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b} |1 - \alpha_g(z) p(z)|. \quad (3.2)$$

This bound has three ingredients.

- The starting vector v only appears in the constant

$$C_1 \equiv \max_{\psi \in \mathcal{P}_{k-1}} \frac{\|\psi(A) P_b v\|}{\|\psi(A) P_g v\|}.$$

  If $k = 1$, $C_1 = \|P_b v\| / \|P_g v\|$ gauges the bias of v toward the desired invariant subspace. For $k > 1$, the eigenvalues of A also influence C_1. For additional details about C_1, see [2, sect. 5.1].
- The constant

$$C_2 \equiv \kappa(\Omega_b) \geq 1$$

  is a measure of the nonnormality of A associated with the unwanted eigenvalues. If A is symmetric, then $\kappa(\Omega_b) = 1$. For nonnormal A, enlarging Ω_b to contain points beyond $\lambda_{k+1}, \ldots, \lambda_n$ generally reduces $\kappa(\Omega_b)$. For a detailed discussion of this constant, see [2, sect. 5.2].
- As the Krylov subspace dimension m grows, the approximation problem

$$\min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b} |1 - \alpha_g(z) p(z)|. \quad (3.3)$$

  gives the mechanism for convergence. The minimization problem seeks polynomials that approximate $1/\alpha_g(z) = (z - \lambda_1)^{-1} \cdots (z - \lambda_k)^{-1}$ over $z \in \Omega_b$. The convergence of this polynomial approximation problem depends on the proximity of Ω_b to the desired eigenvalues $\lambda_1, \ldots, \lambda_k$. Better separation between the desired and undesired eigenvalues yields faster convergence.

3.1. Convergence theory for polynomial preconditioning. Polynomial preconditioning alters the convergence bound (3.2), replacing A by $\pi(A)$.

THEOREM 3.1. *Using the notation established above, the gap between the desired invariant subspace $\mathcal{U}$ (associated with the non-derogatory eigenvalues $\lambda_1, \ldots, \lambda_m$ of A) and the polynomially-preconditioned Krylov subspace $\mathcal{K}_m(\pi(A), v)$ satisfies*

$$\delta(\mathcal{U}, \mathcal{K}_m(\pi(A), v)) \leq \left(\max_{\psi \in \mathcal{P}_{k-1}} \frac{\|\psi(\pi(A)) P_b v\|}{\|\psi(\pi(A)) P_g v\|} \right) \kappa(\Omega_b^\pi) \min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b^\pi} |1 - \alpha_g^\pi(z) p(z)|,$$

provided $\{\pi(\lambda_1), \ldots, \pi(\lambda_m)\} \cap \{\pi(\lambda_{m+1}), \ldots, \pi(\lambda_n)\} = \emptyset$. Here Ω_b^π is a compact subset of $\mathbb{C}$ that contains $\pi(\lambda_{k+1}), \ldots, \pi(\lambda_n)$, and

$$\alpha_g^\pi(z) := \big(z - \pi(\lambda_1)\big) \cdots \big(z - \pi(\lambda_k)\big).$$

For this bound to converge, π must map the desired eigenvalues outside Ω_b^π:

$$\{\pi(\lambda_j)\}_{j=1}^k \ \cap \ \Omega_b^\pi \ = \ \emptyset.$$

Since the spectral projectors are invariant under the transformation $A \mapsto \pi(A)$ (provided the desired and undesired eigenvalues remain disjoint under π), P_b and P_g are the same for A and $\pi(A)$.

Let us inspect this bound in the simplest case of symmetric A with $k = 1$. Since $k = 1$, the constants in (3.2) and Theorem 3.1 that involve v are just $C_1 = \|P_b v\|/\|P_g v\|$. Since A is symmetric, $C_2 = \kappa(\Omega_b) = \kappa(\Omega_b^\pi) = 1$. The only difference in the convergence bounds in (3.2) and Theorem 3.1 then comes from the polynomial approximation problems. As suggested in Figure 2.1, the map π can effectively separate the desired eigenvalues from the rest of the spectrum, making it possible that

$$\min_{p\in\mathcal{P}_{m-2k}} \max_{z\in\Omega_b^\pi} |1 - \alpha_g^\pi(z)p(z)| \ \ll \ \min_{p\in\mathcal{P}_{m-2k}} \max_{z\in\Omega_b} |1 - \alpha_g(z)p(z)|.$$

Keep A symmetric but allow general $k \geq 1$. Presuming that π has real coefficients (so $\pi(\sigma(A)) \subset \mathbb{R}$), denote

$$\Omega_b^\pi \equiv \Big[\min_{j=k+1,\ldots,n} \pi(\lambda_j), \ \max_{j=k+1,\ldots,n} \pi(\lambda_j) \Big] \subset \mathbb{R}.$$

Let $\lambda_* \in \{\lambda_1, \ldots, \lambda_k\}$ denote the desired eigenvalue that is mapped closest to Ω_b^π:

$$\operatorname{dist}(\pi(\lambda_*), \Omega_b^\pi) = \min_{1\leq j\leq k} \operatorname{dist}(\pi(\lambda_j), \Omega_b^\pi) = \min_{1\leq j\leq k} \ \min_{z\in\Omega_b^\pi} |\pi(\lambda_j) - z|.$$

Supposing that $\pi(\lambda_*) \notin \Omega_b^\pi$, define

$$K \equiv \frac{\max_{z\in\Omega_b^\pi} \ |\pi(\lambda_*) - z|}{\min_{z\in\Omega_b^\pi} \ |\pi(\lambda_*) - z|} \geq 1. \tag{3.4}$$

Then using standard Chebyshev approximation theory [25, sect. 6.11], the polynomial approximation problem in Theorem 3.1 converges at the asymptotic rate

$$\rho \equiv \frac{\sqrt{K} - 1}{\sqrt{K} + 1} \in [0, 1), \tag{3.5}$$

meaning that there exists some constant $C > 0$ such that for all $m \geq 2k$,

$$\min_{p\in\mathcal{P}_{m-2k}} \max_{z\in\Omega_b^\pi} |1 - \alpha_g^\pi(z)p(z)| \ \leq \ C\rho^m.$$

Table 3.1 compares this convergence rate ρ for the standard Arnoldi method (first row, $\pi(z) = z$) to the convergence obtained for minimum residual polynomials π of increasing degree d for the symmetric A used in Figure 2.1. We seek the $k = 5$ smallest magnitude eigenvalues of A. (When $d = 1$, the shift-invariance property of the Krylov subspace implies that polynomial preconditioning and standard Arnoldi generate the same approximation subspace: $\mathcal{K}_m(\pi(A), v) = \mathcal{K}_m(A, v)$.) Note that the fastest convergence rate *per polynomial degree*, $\rho^{1/d}$, is obtained for the standard case (or $d = 1$). This is expected: the preconditioned space $\mathcal{K}_m(\pi(A), v)$ only contains a small part of the much larger space $\mathcal{K}_{d(m-1)+1}(A, v)$. However, *the preconditioned method obtains this convergence rate with much lower dimensional subspaces.*

TABLE 3.1

Asymptotic convergence rates for various polynomial degrees d, for the example in Figure 2.1 with $k = 5$ desired eigenvalues. The first row is for standard Arnoldi, $\pi(A) = A$. The set Ω_g^π is the smallest interval that contains $\{\pi(\lambda_j)\}_{j=1}^k$, while Ω_b^π is the smallest interval that contains the map of the unwanted eigenvalues, $\{\pi(\lambda_j)\}_{j=k+1}^n$. When Ω_b^π and Ω_g^π overlap, we set $\rho = 1$.

d	Ω_g^π	Ω_b^π	ρ	$\rho^{1/d}$
standard	$[0.0010, 0.0042]$	$[\ 0.0060, 2.0000]$	0.9416	
1	$[0.9973, 0.9993]$	$[-0.3055, 0.9961]$	0.9416	0.9416
2	$[0.9936, 0.9985]$	$[-0.0705, 0.9908]$	0.9030	0.9503
3	$[0.9840, 0.9962]$	$[-0.2018, 0.9772]$	0.8589	0.9506
4	$[0.9690, 0.9925]$	$[-0.4384, 0.9557]$	0.8232	0.9525
5	$[0.9529, 0.9886]$	$[-0.4155, 0.9329]$	0.7845	0.9526
6	$[0.9292, 0.9829]$	$[-0.4102, 0.8994]$	0.7407	0.9512
7	$[0.9063, 0.9772]$	$[-0.4036, 0.8673]$	0.7057	0.9514
8	$[0.8753, 0.9695]$	$[-0.3909, 0.8240]$	0.6650	0.9503
16	$[0.5563, 0.8830]$	$[-0.3501, 0.4002]$	0.4134	0.9463
24	$[0.1046, 0.7217]$	$[-0.3096, 0.2434]$	1.0000	1.0000

3.2. The effect of restarting on convergence. To obtain accurate eigenvalue estimates while limiting the dimension m of the Krylov subspace (3.4), the Arnoldi algorithm is *restarted* [22, 28]. We begin by describing the standard Arnoldi algorithm with no preconditioning. After taking m steps of the Arnoldi process with the matrix A and starting vector $v_0 \equiv v$ (the first *cycle*), the method runs a fresh set of m steps with the same A but a new starting vector, v_1. In general, cycle $c+1$ takes m Arnoldi steps with A and starting vector v_c. These new starting vectors are formed using polynomial restart methods [22, 23, 28]: given some positive integer $r \leq m - k$, such methods construct $v_c = \phi_c(A)v_{c-1}$, where $\phi_c \in \mathcal{P}_r$. Aggregate these starting vectors to get $v_c = \Phi_c(A)v$, for $\Phi_c \equiv \phi_1 \cdots \phi_c \in \mathcal{P}_{cr}$, so that cycle $c + 1$ of the restarted Arnoldi method uses the Krylov subspace

$$\mathcal{K}_m(A, \Phi_c(A)v) = \text{span}\{\Phi_c(A)v, A\Phi_c(A)v, \ldots, A^{m-1}\Phi_c(A)v\}, \tag{3.6}$$

generally an m-dimensional subspace of $\mathcal{K}_{cr+m}(A, v)$. For any $x \in \mathcal{K}_m(A, \Phi_c(A)v)$ there exists some $\omega \in \mathcal{P}_{m-1}$ such that

$$x = \omega(A)\Phi_c(A)v, \tag{3.7}$$

with $\omega \cdot \Phi_c \in \mathcal{P}_{cr+m-1}$. Like polynomial preconditioning, restarting with polynomial filters gives access to elements in a high-degree Krylov subspace, while keeping the overall subspace dimension low.

The convergence behavior will depend on the polynomial filters [2, 3]. While these filters can be constructed from Chebyshev polynomials [23], filters built from "exact shifts" (unwanted Ritz values) [28] are simpler and more widely used. Such shifts always give convergence for symmetric A [28] (aside from pathological v_0), and usually work well for nonsymmetric A (though they can fail, in theory [6, 7]). Moreover, these filters can be implemented stably using the implicitly restarted Arnoldi [28], thick-restart Arnoldi [16, 17, 29, 34] and Krylov–Schur [30] algorithms. Since these exact-shift filters are built up during each cycle of the restarted Arnoldi procedure, they require numerous inner product evaluations.

Practical computations will combine polynomial preconditioning with polynomial

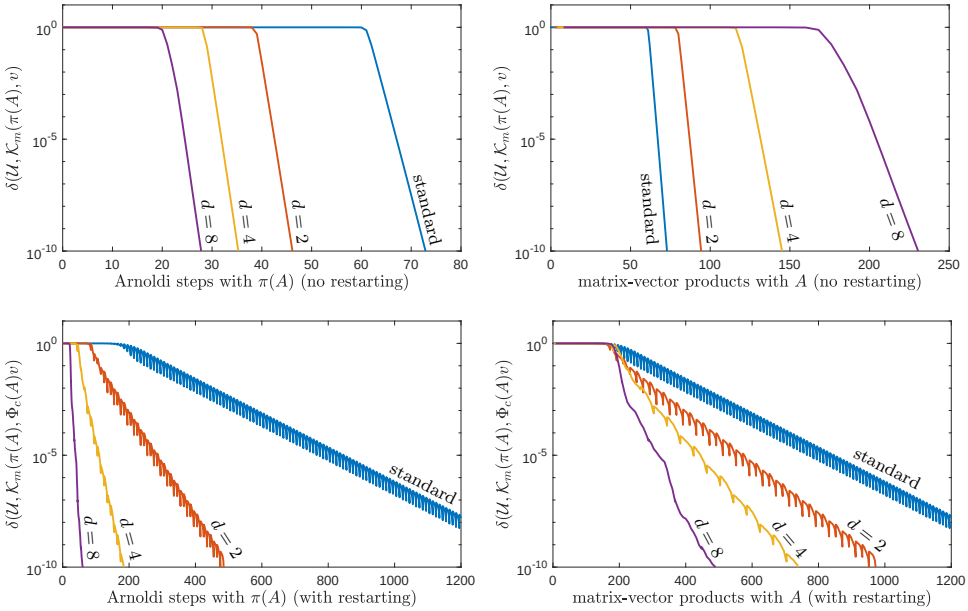


FIG. 3.1. *Arnoldi iterations and matrix-vector products for various choices of d and $k = 5$ eigenvalues, for the example in Figure 2.1 without restarts (top) and with restarts using $m = 2k$ (bottom). Without restarts, standard Arnoldi requires more steps (top left) but fewer matrix-vector products (top right). Restarting can give polynomial preconditioning another advantage, making it faster in terms of both steps (bottom left) and matrix-vector products (bottom right).*

restarting, using the approximation space

$$\mathcal{K}_m(\pi(A), \Phi_c(\pi(A))v) = \\ \text{span}\{\Phi_c(\pi(A))v, \pi(A)\Phi_c(\pi(A))v, \ldots, \pi(A)^{m-1}\Phi_c(\pi(A))v\}. \quad (3.8)$$

Generally (3.8) is an m-dimensional subspace of $\mathcal{K}_{d(cr+m-1)}(A, v)$. Now for any $x \in \mathcal{K}_m(\pi(A), \Phi_c(\pi(A))v)$ there exists a polynomial $\omega \in \mathcal{P}_{m-1}$ such that

$$x = \omega(\pi(A))\Phi_c(\pi(A))v, \quad (3.9)$$

so preconditioning gives access to vectors x that can be written in terms of a polynomial $((\omega \circ \pi) \cdot (\Phi_c \circ \pi))$ of degree $d(cr + m - 1)$, i.e., d times larger than available with restarting alone in (3.7).

How do preconditioning and restarting combine to affect convergence? We first address this question by continuing the experiment started in Figure 2.1, seeking the $k = 5$ smallest magnitude eigenvalues. Figure 3.1 compares convergence of the polynomially-preconditioned Arnoldi algorithm, with and without restarting. In the top figures (no restarts), increasing the preconditioning polynomial degree d improves the convergence, but increases the number of matrix-vector products involving A. The bottom figures incorporate polynomial restarting, limiting the Krylov subspace to have dimension $m = 2k$ and using filter polynomials of degree $r = k$ at each cycle. Polynomial preconditioning improves convergence by a larger margin, enough to also deliver convergence using fewer matrix-vector products.

By adapting [3, eq. (3.10)], we can also provide a bound on the gap between the desired invariant subspace $\mathcal{U}$ and the restarted Krylov subspace using polynomial

preconditioning. Suppose the aggregate polynomial filter $\Psi_c \in \mathcal{P}_{cr}$ has R distinct roots (the shifts), also distinct from the images of the desired eigenvalues, $\Sigma^\pi = \{\pi(\lambda_1), \ldots, \pi(\lambda_k)\}$. Let $\Psi \in \mathcal{P}_{R-1}$ interpolate $1/\alpha_g^\pi$ at these R points. Then

$$\delta(\mathcal{U}, \mathcal{K}_m(A, \Phi_c(A)v)) \leq \left(\max_{\psi \in \mathcal{P}_{k-1}} \frac{\|\psi(\pi(A))P_b v\|}{\|\psi(\pi(A))P_g v\|} \right) \kappa(\Omega_b^\pi) \max_{z \in \Omega_b^\pi} |1 - \alpha_g^\pi(z)\Psi_c(z)|. \quad (3.10)$$

If the roots of Ψ_c are distributed throughout Ω_b^π, we expect $\max_{z \in \Omega_b^\pi} |1 - \alpha_g^\pi(z)\Psi_c(z)|$ to be small. While descriptive bounds on the location of the shifts for nonsymmetric A (or $\pi(A)$) have proved elusive [6, 7], the bound (3.10) provides an indication of how the polynomial preconditioner and the shifts can combine to influence convergence.

4. Experiments. How does polynomial preconditioning perform in practice? In this section we explore three examples involving nonsymmetric A. Here and in future sections, our experiments use the thick restarted Arnoldi method [17, 34], which we refer to as Arnoldi(m, k): m denotes the largest subspace dimension and $k < m$ denotes the number of Ritz values kept at each restart.[3] We seek the $nev < k$ smallest magnitude eigenvalues, leaving a buffer of $k - nev$ eigenvalues to accelerate convergence. Each orthogonalization step is followed by a pass of reorthogonalization. Convergence is tested at each Arnoldi cycle using the original matrix A: Let $\nu_1, \ldots, \nu_m$ denote the Ritz values for $\pi(A)$ at the end of a cycle, ordered by increasing distance from 1 ($|1 - \nu_1| \leq |1 - \nu_2| \leq \cdots \leq |1 - \nu_m|$), and let $y_1, \ldots, y_m \in \mathbb{C}^n$ denote the associated unit-norm Ritz vectors. Then the Rayleigh quotient $\mu_j \equiv y_j^* A y_j$ gives an eigenvalue estimate for A. When seeking $nev < k$ eigenvalues, we require $\|Ay_j - \theta_j y_j\| \leq rtol$ for $j = 1, \ldots, nev$, with $rtol = \|A\|10^{-8}$ unless otherwise stated. The reported operation counts give a rough impression of the matrix-vector products with A, dot products, and other vector operations (scalar multiplications and additions); these counts will vary a bit with implementation details (e.g., reorthogonalization strategy, which residuals are checked at each cycle, etc.). In each case, to explore robustness we use a random starting vector to generate π, and a different random starting vector for the Arnoldi iterations. (In practice one might naturally use the same starting vector to generate π and for the Arnoldi iterations.) We average our results over ten trials, to minimize variation due to these starting vectors.

Example 1. Consider a second-order finite difference discretization of a convection-diffusion equation on the unit square with homogeneous Dirichlet boundary conditions. On the bottom half of the square, the operator is $-u_{xx} - u_{yy} + 20u_x$; on the top half, $-100u_{xx} - 100u_{yy} + 2000u_x$. We use five increasingly fine discretizations that give matrices of size $n = 2500$, 10,000, 40,000, 160,000 and 640,000. We seek the $nev = 15$ smallest magnitude eigenvalues and the corresponding eigenvectors using Arnoldi(50,20), meaning the maximum subspace dimension is $m = 50$, and $k = 20$ Ritz vectors are saved at the restart.

Figure 4.1 compares regular Arnoldi(50,20) to degree $d = 25$ polynomial preconditioned Arnoldi(50,20) for the five matrices, giving both the number of matrix-vector products for convergence (left axis) and an estimate of the total cost (right axis). For $n = 2{,}500$, polynomial preconditioning uses slightly more matrix-vector products. As the matrix size increases and the eigenvalue problem becomes more difficult, polynomial preconditioning becomes increasingly better in comparison. For $n = 640{,}000$,

[3] All examples involve real matrices; to preserve real arithmetic during the restarting process, sometimes k is temporarily reduced to $k - 1$ to avoid splitting a conjugate pair of Ritz values.

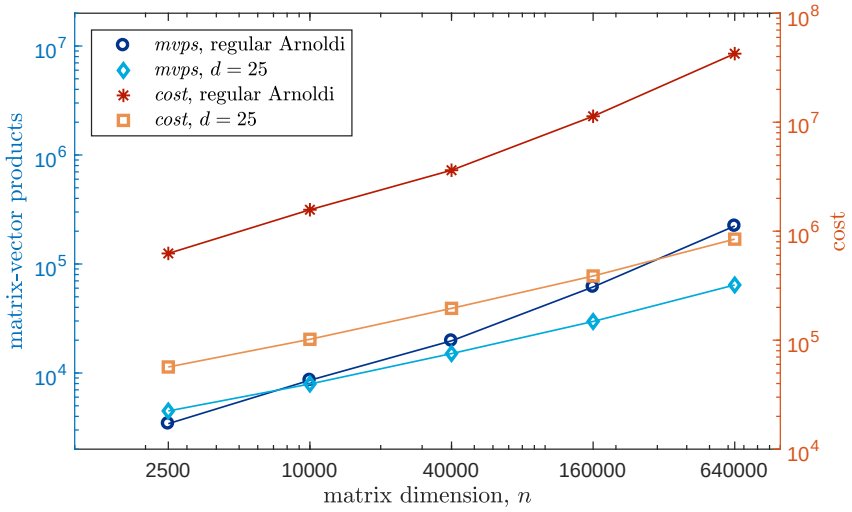


FIG. 4.1. *Example 1 (convection diffusion matrix): Comparison of regular Arnoldi(50,20) to polynomial preconditioned Arnoldi(50,20) with degree $d = 25$ for matrices of size 2,500, 10,000, 40,000, 160,000 and 640,000. Circles (regular Arnoldi) and diamonds (polynomial preconditioned) indicate the number of matrix-vector products (left axis). Asterisks and squares show the corresponding approximate cost ($cost = nnzr \times mvps + vops$, right axis).*

regular Arnoldi averages 223,103 matrix-vector products, while polynomial preconditioned Arnoldi with $d = 25$ only averages 64,227.3.

The computational cost is estimated as $cost = nnzr \times mvps + vops$, where $nnzr \approx 5$ is the average number of nonzeros per row in A, *mvps* is the number of matrix-vector products, and *vops* is the number of length-n vector operations, such as dot products and daxpy's. Of course, the cost of an *mvp* compared to a *vop* depends on the computer and implementation, but this estimate shows the potential for polynomial preconditioning to reduce computational cost. In Figure 4.1, *cost* is associated with the right axis (asterisks for regular Arnoldi, squares for $d = 25$). The axes are scaled so the values on the left axis are one-fifth of the corresponding height on the right axis, allowing one to see what portion of *cost* is due to *mvps*. For regular Arnoldi, most matrix-vector products are accompanied by an orthogonalization step; preconditioning uses more matrix-vector products (to compute $\pi(A)v$) relative to vector operations. For this sparse A, *vops* dominate *mvps* for regular Arnoldi. Polynomial preconditioning with $d = 25$ reduces the *vops* per matrix-vector product by a factor of about 22 for all the n values shown here. With $n = 2{,}500$, the *cost* for regular Arnoldi is 624,768, but only 56,755 for polynomial preconditioning. With $n = 640{,}000$, the comparison is 42,534,104 to 845,416. Increasing the degree to $d = 100$ drops the cost to 524,137: over 80 times cheaper than regular Arnoldi.

We continue the example by looking at the polynomials for the matrix of size 10,000. The upper portion of Figure 4.2 shows representative GMRES polynomials of degree 10 and 25, evaluated at all of the eigenvalues. The steeper slope at the origin of the $d = 25$ polynomial better separates the small eigenvalues from the others, as is clear from the close-up plot on the bottom. The desired first $nev = 15$ eigenvalues come first, followed by the next $k - nev = 5$ eigenvalues that serve as a buffer for the desired ones, and finally the next few eigenvalues: the polynomial mapping separates the small eigenvalues from the others. A gap ratio for the 15th eigenvalue that takes

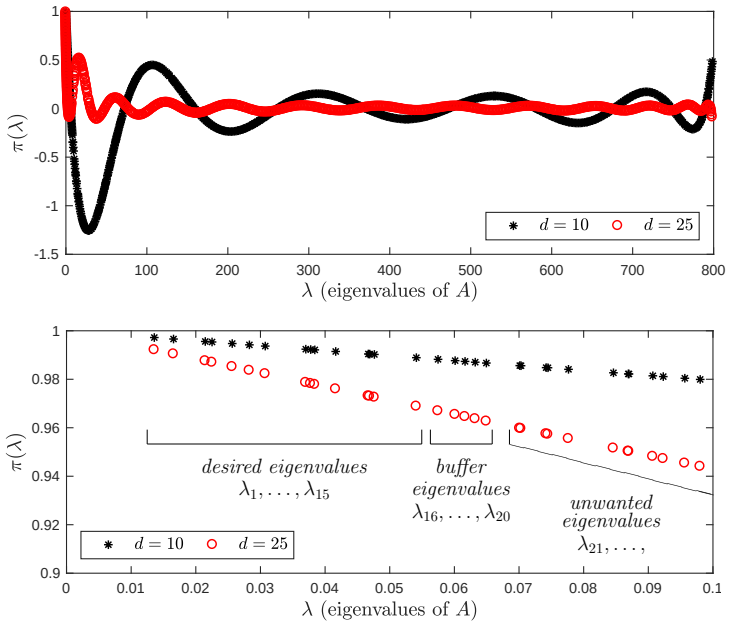


FIG. 4.2. *Example 1 (convection diffusion matrix, $n = 10{,}000$): Polynomials of degree $d = 10$ (black) and $d = 25$ (red) for the convection-diffusion matrix of size $n = 10{,}000$. The upper plot shows the polynomial values $\pi(\lambda)$ plotted for all eigenvalues λ of A. The bottom plot zooms in on the smallest eigenvalues, highlighting the sought-after $nev = 15$ smallest magnitude eigenvalues, the buffer of $k - nev = 5$ additional eigenvalues, and a few of the remaining unwanted eigenvalues.*

into account the buffer Ritz values is $\frac{\lambda_{21}-\lambda_{15}}{\lambda_{10000}-\lambda_{21}} = 2.00 \times 10^{-5}$ for the matrix A, which gives some indication of the eventual convergence of that eigenvalue. With $d = 10$ polynomial preconditioning, the gap ratio improves to $\frac{\pi(\lambda_{21})-\pi(\lambda_{15})}{\pi(\lambda_{5099})-\pi(\lambda_{21})} = 1.14 \times 10^{-3}$, and for $d = 25$, it is still better: $\frac{\pi(\lambda_{21})-\pi(\lambda_{15})}{\pi(\lambda_{5145})-\pi(\lambda_{21})} = 1.62 \times 10^{-2}$. While these ratios suggest that polynomial preconditioning improves the convergence rate in terms of Arnoldi cycles, a larger gap ratio *does not* guarantee a reduction of overall matrix-vector products, since each iteration with a higher degree polynomial requires more of them. As Figure 4.1 shows, for $n = 10{,}000$ the number of matrix-vector products for $d = 25$ is a bit smaller than for regular Arnoldi.

Figure 4.3 shows results for the matrix of size $n = 160{,}000$ using different degree polynomials. Standard Arnoldi(50,20) is plotted at $d = 0$; then polynomial preconditioned Arnoldi(50,20) is applied with $d = 5, 10, 15, \ldots, 50$. The matrix-vector products, denoted on the left axis, hit a minimum for $d = 15$ and increase slightly beyond that. The cost estimate (right axis) decreases. For $d = 50$, *cost* $\approx 316{,}060$. The *cost* can go down a bit further with higher d; for $d = 150$, *cost* $\approx 298{,}308$. See Table 8.1 for further testing with a larger version of this example, including timings and dot product counts.

Example 2. Now consider the matrix Af23560 from Matrix Market, a nonsymmetric matrix of size $n = 23{,}560$ with an average of $nnzr \approx 19.6$ nonzeros per row. Again, we seek the smallest magnitude eigenvalues. The spectrum is complex; see the top left of Figure 4.6. Polynomial preconditioning is less effective for this example than the previous one, since low degree polynomials struggle to sufficiently isolate the smallest magnitude eigenvalues. With less sparsity than the last example, the

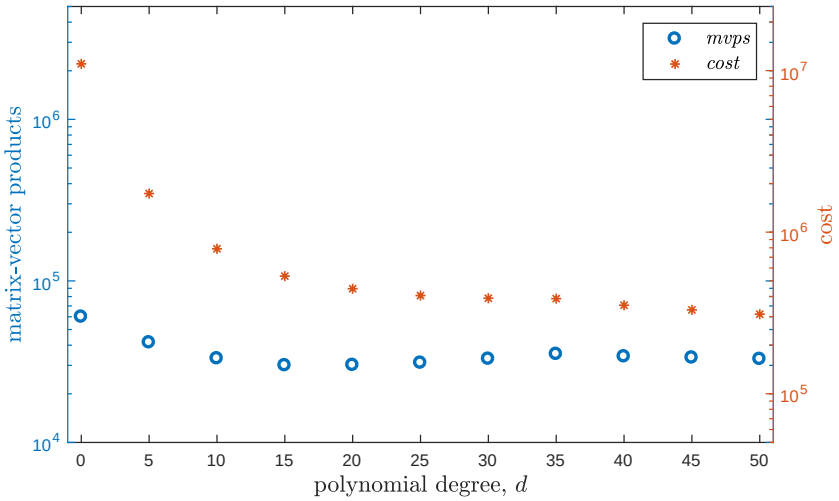


FIG. 4.3. *Example 1 (convection diffusion matrix, $n = 160{,}000$): A comparison of convergence between standard Arnoldi(50,20) ($d = 0$) and polynomial preconditioned Arnoldi(50,20) with $d = 5, 10, 15, \ldots, 50$. Circles indicate the number of matrix-vector products (left axis). The approximate cost ($cost = nnzr \times mvps + vops$) is indicated with asterisks (right axis).*

matrix-vector products form a larger part of the expense for this example.

Figure 4.4 compares convergence for different degree polynomials to no preconditioning ($d = 0$), showing both matrix-vector products (left axis) and the cost estimate $cost = nnzr \times mvps + vops$ (right axis). The right axis is scaled by $nnzr \approx 19.6$, relative to the left axis, to make it easy to see the proportion of cost due to matrix-vector products. Figure 4.4 also shows the matrix-vector products required by the harmonic Arnoldi method [15, 17, 19] (which keeps harmonic Ritz vectors when restarting, rather than standard Ritz vectors). (The associated cost estimate is not shown, but is brought down by a similar proportion.) Each point in Figure 4.4 is the average of 10 trials. We note that while all trials converged to the desired tolerance, some only found a subset of the $nev = 15$ desired (smallest magnitude eigenvalues); a few trials found as few as 11 of these desired eigenvalues.

Focusing on $d = 5$ (with standard, not harmonic, Arnoldi), polynomial preconditioning uses 2.81 times as many matrix-vector products as with no preconditioning. Nevertheless, the cost estimate is reduced by about 20%, because the preconditioned method uses many fewer orthogonalization steps. For $d = 40$, the matrix-vector product count is nearly 25% higher than for no preconditioning, but *cost* is reduced by 84%. Figure 4.5 shows the magnitude of the $d = 5$ and $d = 40$ minimum residual polynomials $\pi(\lambda)$ on the eigenvalues λ of A in the complex plane for a typical example. In the top plot ($d = 5$) this polynomial is 1 at the origin, $\pi(0) = 1$, but the degree is not large enough for $|\pi(\lambda)|$ to be small at all the eigenvalues of A: in particular, $|\pi(\lambda)|$ is large at some large-magnitude eigenvalues on the periphery of the spectrum, with $|\pi(\lambda)| > 3$ for one conjugate pair of eigenvalues. The middle left portion of Figure 4.6 shows the resulting spectrum of $\pi(A)$. We seek the eigenvalues of A near the origin, and expect π to map these close to 1; this is the case, but now these eigenvalues are actually in the interior of the spectrum, due to other eigenvalues λ for which $|\pi(\lambda)| > 1$, complicating the eigenvalue computation. This situation motivates our use of the harmonic Arnoldi method (with the same $d = 5$ polynomial preconditioning), since

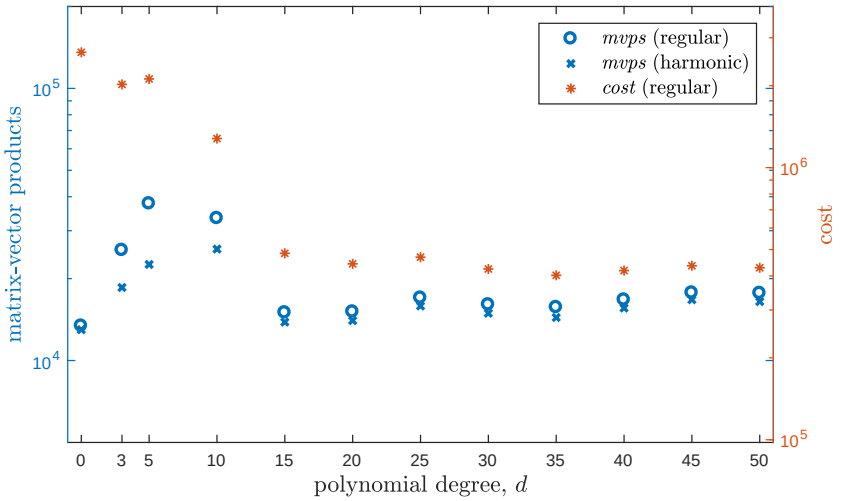


FIG. 4.4. *Example 2 (matrix Af23560): A comparison of convergence between standard Arnoldi(50,20) ($d = 0$) and polynomial preconditioned Arnoldi(50,20) with $d = 3, 5, 10, 15, \ldots, 50$. Circles indicate the matrix-vector products for regular Arnoldi restarted with Ritz vectors; crosses show the matrix-vector products for Arnoldi restarted with* harmonic *Ritz vectors (left axis). The approximate cost ($cost = nnzr \times mvps + vops$) for regular Arnoldi is shown by asterisks (right axis).*

the harmonic Arnoldi variant is often better for interior eigenvalue problems [17]. For the experiments shown in Figure 4.4, the harmonic Arnoldi method reduces the matrix-vector products by 40%. The lowest portions of Figures 4.5 and 4.6 show similar spectral information for the minimum residual polynomial with $d = 40$. In this case π does a better job of being small at the eigenvalues not near the origin, but it still has some trouble at the eigenvalues with largest imaginary part, which get mapped close to 1. Nevertheless, these eigenvalues are no longer mapped so that an interior eigenvalue problem is created. The right side of Figure 4.6 shows close-up views of the spectrum of A on top and the polynomial preconditioned spectra of $\pi(A)$ for $d = 5$ and $d = 40$ below. Though the overall spectra are vastly changed by the polynomial preconditioning, these close-ups are very similar.

Example 3. The matrix E20r0100 from Matrix Market also has a complex spectrum, but this time polynomial preconditioning is more effective, reducing the overall matrix-vector products. This matrix has dimension $n = 4241$ with an average of nearly 31 nonzeros per row. As before, we seek the $nev = 15$ eigenvalues nearest the origin, which are in the interior of the spectrum (A has 1199 eigenvalues with (quite small) negative real parts), though now we use Arnoldi(100,30) iterations to access larger subspaces. Figure 4.7 shows how polynomial preconditioning changes the spectrum, and Table 4.7 gives results for different degree polynomials. A degree $d = 15$ polynomial reduces the number of matrix-vector products by a factor of almost 9 and vector operations by a factor of more than 125. Matrix-vector products are not further reduced with higher degree polynomials, but the vector operations are.

5. Two Starting Vectors. For the examples in the last section we randomly generated the starting vector used to create π, an approach that seems to work quite well in practice. However, here we consider the possibility that an unusual or skewed starting vector for π can give bad results, and show how two starting vectors can be used to generate π to minimize the risk of a bad starting vector.

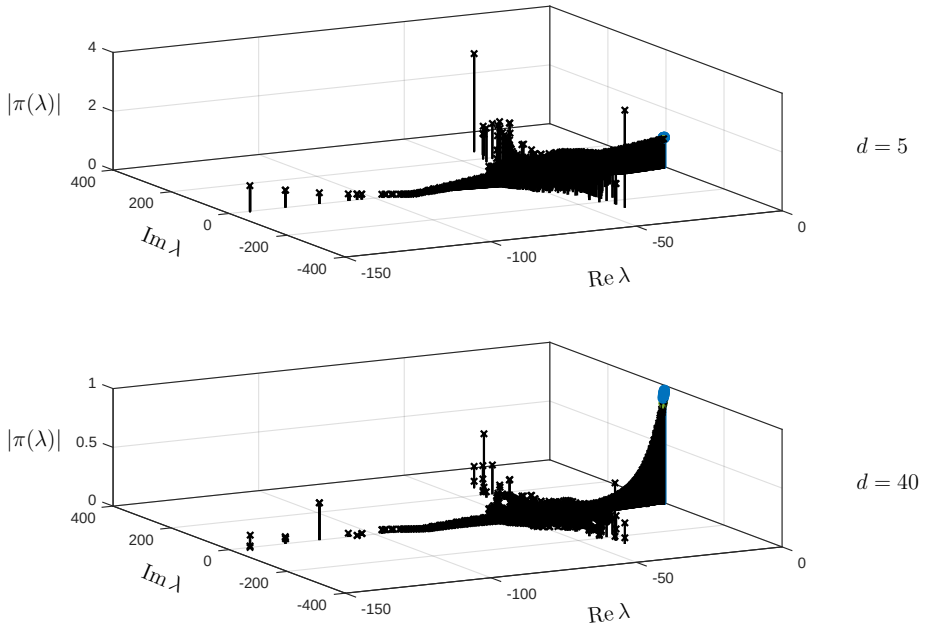


FIG. 4.5. *Example 2 (matrix Af23560): The magnitude of the GMRES polynomial π at each eigenvalue of A, for degrees $d = 5$ (top) and $d = 40$ (bottom).*

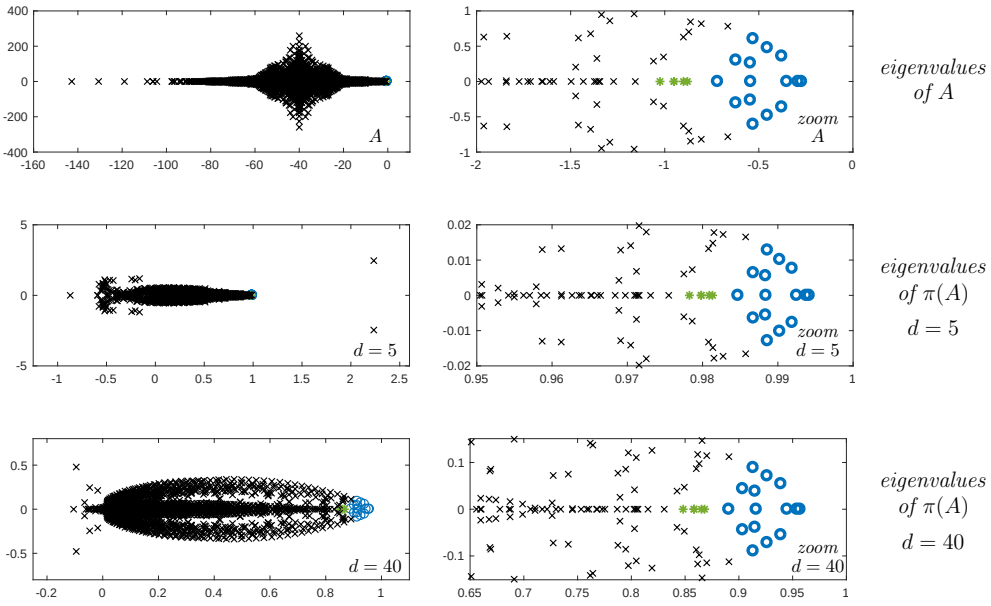


FIG. 4.6. *Example 2 (matrix Af23560). The top plots show the eigenvalues of A, with the $nev = 15$ desired eigenvalues (nearest 0) as blue circles and the $k - nev = 5$ buffer eigenvalues as green stars; the undesired eigenvalues are shown as black pluses. The middle plots show the eigenvalues of the preconditioned matrix $\pi(A)$ for degree $d = 5$; the desired eigenvalues are mapped near 1, but these are interior eigenvalues of $\pi(A)$. The bottom plots show the eigenvalues of $\pi(A)$ for $d = 40$: the desired eigenvalues are now on the exterior of the spectrum.*

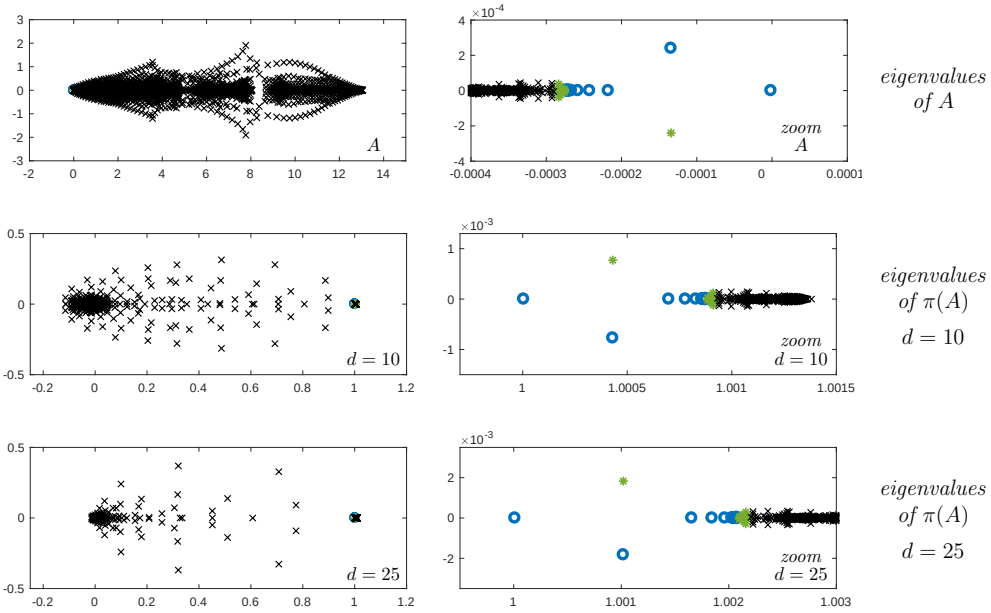


FIG. 4.7. *Example 3 (matrix E20r0100). The top plots show the eigenvalues of A, with the nev = 15 desired eigenvalues (nearest 0) as blue circles and the k − nev = 15 buffer eigenvalues as green stars; the undesired eigenvalues are shown as black pluses. The middle plots show the eigenvalues of the preconditioned matrix π(A) for degree d = 10; the desired eigenvalues are mapped near 1. The bottom plots show the eigenvalues of π(A) for d = 25.*

TABLE 4.1
Example 3 (Matrix e20r0100, n = 4241): results for polynomial preconditioning with different degree polynomials using Arnoldi(100,30) to seek nev = 15 eigenvalues, averaged over 10 trials.

degree d	cycles	*mvps*	*vops* (millions)	dot products (millions)	*cost* (millions)	# correct eigenvalues
0	7,959.1	558,258.0	171.80	74.41	189.10	14.8
5	308.6	109,067.6	6.82	2.89	10.20	14.0
10	97.4	68,926.6	2.19	0.91	4.33	14.0
15	58.9	62,740.8	1.35	0.55	3.29	14.0
20	46.0	65,510.1	1.07	0.43	3.10	14.0
25	43.2	76,936.4	1.02	0.41	3.41	14.0
50	44.5	158,253.2	1.14	0.42	6.04	14.7
75	27.9	149,730.0	0.77	0.27	5.41	14.8
100	14.8	107,274.7	0.46	0.15	3.78	14.0

Example 4. Let A be diagonal with main diagonal elements $1, 2, 3, \ldots, 1000$. We run Arnoldi(50,20) to calculate $nev = 15$ eigenvalues to residual norm of 10^{-8} and use a different starting vector for the polynomial preconditioned Arnoldi loop than was used for developing the polynomial (with $d = 10$). It takes only one cycle to find all 15 correct eigenpairs. Next, we make the last 100 components of the starting vector for the polynomial small by multiplying them by 0.01. The resulting π is not small much past $\lambda = 930$; $\pi(\lambda)$ goes up to at least 7 at $\lambda = 1000$, transforming the problem of finding the eigenvalues near 1 into an interior eigenvalue problem. Convergence is

much slower, with 16.8 cycles needed (average of 10 trials), and then only the first four desired eigenvalues ($\lambda = 1, 2, 3, 4$) are found. The remaining computed eigenvalues fall in $\{938, 939, \ldots, 956\}$, depending on the run: eigenvalues λ that π maps closer to the target point 1 than the desired eigenvalues $\lambda = 5, \ldots, 15$.

We give an algorithm that uses two starting vectors to determine one polynomial, applying GMRES to a 2×2 block diagonal system of dimension $2n$.

Polynomial Determined by Two Starting Vectors

1. **Set-up:** Generate random vectors b_1 and b_2, with $\|b_1\| = \|b_2\| = 1/\sqrt{2}$. Let
$$\widehat{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \qquad \widehat{A} = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix}.$$
2. **Generate polynomial:** Run a cycle of GMRES(d) with starting vector $\widehat{b}$ and matrix $\widehat{A}$, and find the roots of the GMRES polynomial π.

This approach essentially uses two Krylov subspaces, one each with b_1 and b_2, and so takes into account both starting vectors. We tested Example 4 with the skewed starting vector for b_1, but b_2 a random vector. The results are good: we now need only one or two cycles to compute all the desired small eigenvalues.

6. Damped Polynomials. Sometimes the GMRES polynomial π is not an ideal preconditioner for eigenvalue calculations. In this section we examine several scenarios that can lead to poor preconditioners, and propose techniques to tame the extreme behavior of the GMRES polynomial; we call the result a "damped" polynomial. Section 6.1 addresses the case were $\pi(\lambda)$ is too large at undesired eigenvalues λ; Section 6.2 investigates problems that are *too easy*, making $\pi(\lambda)$ too small at desired λ. Both cases can be improved with a damped polynomial.

6.1. Overenthusiastic polynomials. The next example considers a case where $|\pi(\lambda)|$ is large at unwanted eigenvalues; it also shows how polynomial preconditioning can be effective even for a matrix with less sparsity than the earlier examples.

Example 5. Consider S1rmq4m1 from Matrix Market, a symmetric, positive definite matrix of size $n = 5489$ with an average of $nnzr \approx 47.8$ nonzeros per row. Finding the small eigenvalues is difficult because they are packed closely together relative to the whole spectrum. The first 15 range from 0.3797 to 1.9027, the 21st is 2.2630 and the largest is 6.87×10^5. Figure 6.1 shows the performance of Arnoldi(50,20) applied to find the $nev = 15$ smallest magnitude eigenvalues of A. The gap ratio $\frac{|\lambda_{21}-\lambda_{15}|}{|\lambda_{5489}-\lambda_{21}|} \approx 5.24 \times 10^{-7}$ roughly describes the eventual convergence for the 15th eigenvalue. For a typical case of polynomial preconditioning with $d = 20$, $\frac{|\pi(\lambda_{21})-\pi(\lambda_{15})|}{|\pi(\lambda_{2737})-\pi(\lambda_{21})|} = \frac{|0.99728-0.99771|}{|-1.82056-0.99728|} \approx 1.54 \times 10^{-4}$ (though recall each preconditioned Arnoldi iteration requires 20 matrix-vector products). Nevertheless, there is an improvement in matrix-vector products by a factor of 6.5 over these ten trials, and $cost = nnzr \times mvps + vops$ is reduced by a factor of 25.51. Though A has a relatively large number of nonzeros, $nnzr \approx 47.8$, the cost of vector operations still dominates for regular Arnoldi; in contrast, with polynomial preconditioning the matrix-vector products are the bigger expense. Moreover, for $d = 20$ polynomial preconditioning decreases the dot products by a factor of 130.34 over regular Arnoldi.

Figure 6.1 shows that polynomials of degree $d > 20$ can cause problems. The performance starts to vary widely over our 10 trial runs; for each $d > 20$, at least one run failed to find the correct $nev = 15$ smallest eigenvalues. Figure 6.2 indicates the problem, showing $\pi(\lambda)$ for $d = 10$, 20, and 30 for the first of our ten trials. The

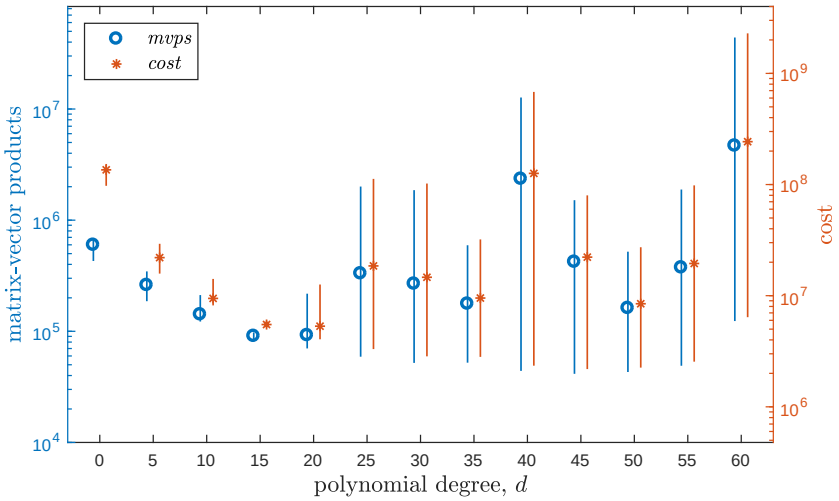
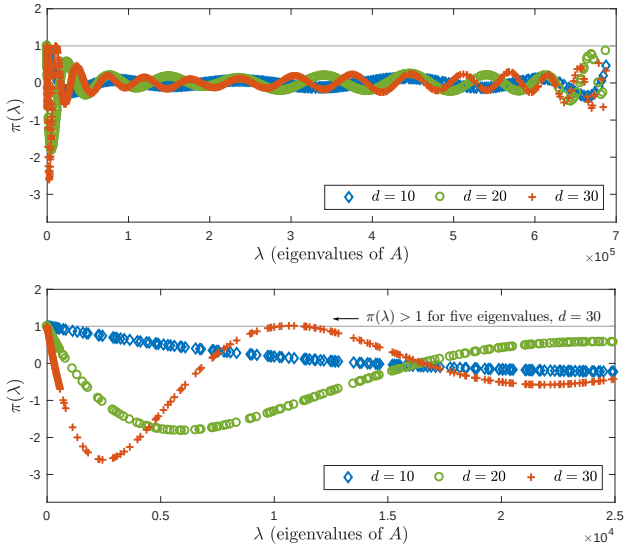


FIG. 6.1. *Example 5 (S1rmq4m1 matrix): Convergence of standard Arnoldi(50,20) ($d = 0$) and polynomial preconditioned Arnoldi(50,20) with $d = 5, 10, 15, \ldots, 60$. Blue circles indicate the average number of matrix-vector products (left axis); red asterisks show the average approximate cost ($cost = nnzr \times mvps + vops$, right axis). These results vary widely over ten trials; vertical bars show the minimum and maximum matrix-vector products and cost over these trials.*

slope of π at the origin is much steeper for $d = 20$ than for $d = 10$, explaining the faster convergence. Degree $d = 30$ is even steeper, but has a problem near 10,000, where the $\pi(\lambda) > 1$ for five eigenvalues: π maps eigenvalues of A from the interior of the spectrum to the exterior of $\pi(A)$, mixing seven of them amongst or above the $nev = 15$ desired smallest eigenvalues of $\pi(A)$ near 1. The resulting interior eigenvalue problem can lead to slow convergence and spurious eigenvalues. Of the converged Ritz values for this example, only 11 fall among the nev desired smallest magnitude eigenvalues; the others are unwanted interior eigenvalues of A. As Figure 6.1 shows, erratic convergence continues for larger values of d. (One run with $d = 60$ failed to compute *any* of the eigenvalues correctly, and took 24,287 cycles to converge; in a run not included in the plot, a $d = 30$ run failed to converge in 30,000 cycles.)

We call the polynomials that jump up too high "overenthusiastic". This matrix seems prone to such polynomials because about half its spectrum is near 0 (2703 eigenvalues are less than 600; the next 2786 go from 700 up to 6.87×10^5): the GMRES polynomial seems to concentrate on these small eigenvalues, while focusing less precisely on the others.

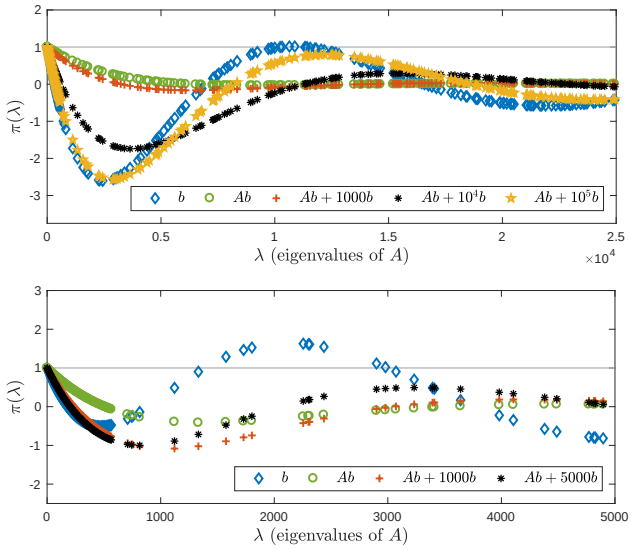
Damping the polynomial provides a possible remedy to overenthusiasm. Damping techniques are considered in [12] for symmetric matrices and for polynomials that approximate the Dirac delta function. Here we take a different approach. We first damp by changing the starting vector for GMRES(d) from a random vector b to Ab: premultiplication by A will generally reduce the components of b in the eigenvectors corresponding to the small eigenvalues. (Think of performing one step of the power method.) The GMRES polynomial for Ab is then less likely to be overenthusiastic, because it does not try as hard to be small at the small eigenvalues. The top portion of Figure 6.3 shows polynomials of degree $d = 30$. Diamonds show the GMRES polynomial generated from b; circles show the damped GMRES polynomial generated from Ab. The damped polynomial no longer jumps too high in the middle of the spectrum, but its slope is much less steep at the origin than for the standard polynomial: thus



$d = 30$

$d = 30$ *zoom*

FIG. 6.2. *Example 5 (S1rmq4m1 matrix): Polynomials of degree $d = 10$, 20 and 30. The top plot shows $\pi(\lambda)$ for all eigenvalues λ of A; the bottom plot zooms in near the origin.*



$d = 30$ *zoom*

$d = 60$ *zoom*

FIG. 6.3. *Example 5 (S1rmq4m1 matrix): Close-ups near the origin of polynomials of degree $d = 30$ (top) and $d = 60$ (bottom), using different damping strategies.*

it yields slower than desired convergence. For the run shown here, $cost = 4.28 \times 10^6$, about the same as for undamped $d = 20$ ($cost = 4.23 \times 10^6$); of course, both are better than $cost = 97.4 \times 10^6$ required without polynomial preconditioning. Next we try starting vectors of the form $Ab + \alpha b$. Figure 6.3 shows results for a few choices of α. For $\alpha = 10^5$, $cost = 3.34 \times 10^6$, but this π is on the verge of being overenthusiastic.

The bottom of Figure 6.3 repeats this experiment with $d = 60$. The undamped polynomial is very enthusiastic; it goes far above 1.0. Again this can be controlled with the starting vector $Ab + \alpha b$, but now a higher proportion of the Ab term is needed. With $\alpha = 5000$, the *cost* goes down to 2.50×10^6.

6.2. Easy problems. The GMRES polynomial π starts at $\pi(0) = 1$ at the origin and tries to drop down quickly to be near zero over the spectrum. For easy problems (where the spectrum is well separated from the origin) and high degree polynomials, π can drop down too fast, mapping some desired eigenvalues near zero, mixed amongst the undesired eigenvalues. Figure 2.1 shows such behavior: for $d = 8$ the first 10 eigenvalues of $\pi(A)$ are separated from the others, but the next few are mixed with the rest of the spectrum. The preconditioner π will be effective if $\pi(\lambda) \approx 1$ for the desired eigenvalues, and $|\pi(\lambda)|$ is small for the undesired ones. To promote such behavior one can use small d, or operate on $A - \mu I$ for μ near the desired eigenvalues. Here we show how damping can also help.

Example 6. Let A be the diagonal matrix of dimension $n = 10{,}000$ with diagonal entries $1, 2, 3, \ldots, 9999, 10000$. We run Arnoldi(50,20) to calculate the $nev = 15$ smallest magnitude eigenvalues to residual norm $rtol = 10^{-8}$, averaging over ten trials. Without preconditioning, the calculation takes 1,625 *mvps* and 287,282 *vops*. With $d = 40$, more matrix-vector products are needed (2,775.6 *mvps*), but other costs are much lower (15,390.8 *vops*). Increasing to $d = 50$ changes the results dramatically: π is too small at some desired eigenvalues; all 10 trials miss $\lambda = 13, 14, 15$; some runs miss more. (The various trials compute unwanted eigenvalues among $\{66, 67, \ldots, 72\}$.) Furthermore, convergence is much slower, with many eigenvalues mapped close together: 12,621.7 *mvps* are needed. For $d = 50$, the GMRES problem is too easy.

We next try the damped polynomial with Ab as the GMRES starting vector for $d = 50$. The performance is better; the correct eigenvalues are found in 2,565.0 *mvps*, a single cycle. If we continue increasing d, even the damped polynomial runs into trouble by coming down too quickly. When $d = 80$ most of the trials miss at least one eigenalue, but this can be fixed by damping more, using starting vector $A^2 b$.

6.3. A heuristic toward automation. We outline an attempt to automatically determine when to damp, for both too easy and overenthusiastic situations.

Heuristic for Damping b and Adjusting d

1. Apply one Arnoldi(m, k) cycle to $\pi(A)$ with starting vector b.
2. Compute the Ritz values $\nu_1, \ldots, \nu_m$ for $\pi(A)$ and associated (unit-length) Ritz vectors $y_1, \ldots, y_m$ from $\mathcal{K}_m(\pi(A), b)$, ordered by increasing distance from 1:
$$|1 - \nu_1| \le |1 - \nu_2| \le \cdots \le |1 - \nu_m|.$$
3. Compute Rayleigh–Ritz eigenvalue estimates *for* A: $\mu_j \equiv y_j^* A y_j$.
4. Check the *Ideal Order Condition*:
$$|\mu_1| \le |\mu_2| \le \cdots \le |\mu_{nev}| < \min_{nev+1 \le j \le k} |\mu_j|.$$
5. If the Ideal Order Condition holds, proceed with the computation;
   If the Ideal Order Condition fails:
   - Generate a new π using starting vector Ab. Repeat steps 1–3.
   - If the Ideal Order Condition still fails, replace d with $\lfloor d/2 \rfloor$ and try again.
   - Continue halving d until the condition holds (or $d = 1$).

Example 6 (continued). For the matrix in Example 6 with $d = 50$, a cycle of Arnoldi(50,20) applied to $\pi(A)$ leads to these Rayleigh quotients $\mu_1, \ldots, \mu_k$ for A:

$$1,\ 2, 3,\ \ldots, 12,\ \underline{68.83},\ \underline{67.62},\ \underline{68.72},\ 71.74,\ 67.26,\ 67.86,\ 84.61,\ 72.80.$$

(Continuing the Arnoldi(50,20) algorithm with this $\pi(A)$ only finds 10 of the desired $nev = 15$ eigenvalues.) The Ideal Order Condition requires these Rayleigh quotients to increase monotonically in magnitude and be dominated by the last $k - nev = 5$ values; the underlined values violate one or both of these requirements, so the heuristic replaces b with Ab and computes a fresh π. This damping is successful: Arnoldi(50,20) converges to all $nev = 15$ desired eigenvalues in just one cycle.

We tested 100 runs each of degrees $d = 30, 31, \ldots, 60$ (a total of 3100 tests). The problem is sometimes "too easy" starting at $d = 39$, with increasing prevalence as d increases. Of the 3100 tests, 1425 failed the first damping test. (We ran Arnoldi(50,20) anyway: *all 1425 cases failed to find all* $nev = 15$ *correct eigenvalues.*) With damping (replacing random b by Ab), all 1425 cases passed the second damping test; Arnoldi(50,20) proceeded to compute all $nev = 15$ desired eigenvalues.

The damping test is not perfect, however. Of the 1675 tests that passed the first damping test, about 95% of the cases successfully computed the $nev = 15$ eigenvalues. The remaining 82 cases failed to find some of the desired eigenvalues (although all found at least 8 correct eigenvalues). While this simple damping test is not perfect, in many cases it signals the need for a modified approach.

7. Stability. Several earlier examples hinted that high degree preconditioners can be temperamental. Here we investigate how such polynomials, with their steep slopes, can lead to a computational instability, and propose a way to cope.

Example 7. Let A be the diagonal matrix of order $n = 10{,}000$ with diagonal entries $0.1, 0.2, 0.3, \ldots, 9.9, 10, 11, 12, \ldots, 9908, 9909, 20000$, giving 100 relatively small eigenvalues and one outlying eigenvalue, $\lambda = 20{,}000$. Using Arnoldi(50,20) with $d = 5$, the $nev = 15$ computed eigenvalues all reach a residual norm at or below $rtol = 2.1 \times 10^{-11}$, marginally better than the 6.1×10^{-11} obtained without polynomial preconditioning. With $d = 10$, the accuracy degrades to 7×10^{-9}, while $d = 15$ only reaches 2.3×10^{-6}. Figure 7.1 shows the $d = 15$ residual convergence (top), and the corresponding preconditioner π (bottom, solid line): π has a root at $\theta_{15} = 20{,}000.0000000000036379$, which of course is very near the large eigenvalue $\lambda_{10000} = 20{,}000$. When $\pi(A)v$ is computed for some vector v, the factored form $\pi(A) = \Pi_{j=1}^{15}(I - A/\theta_j)$ is used. The component of $\pi(A)v$ in the direction of the eigenvector z_{10000} that corresponds to the large eigenvalue is $\gamma_{10000}\Pi_{i=1}^{15}(1 - \lambda_{10000}/\theta_i)z_{10000}$, where γ_{10000} is the coefficient for z_{10000} in an eigenvector expansion of v. Fourteen of the $(1 - \lambda_{10000}/\theta_i)$ terms magnify this component and the fifteenth reduces it back down, but with substantial cancellation error: indeed, in this case $1 - \lambda_{10000}/\theta_{15} \approx 2.22 \times 10^{-16}$ (machine epsilon).

We can monitor the possible loss of accuracy due to this cancellation error by computing how large the component is blown up by the other terms. Define

$$pof(j) \equiv \prod_{\substack{i=1 \\ i \neq j}}^{d} |1 - \theta_j/\theta_i|;$$

"*pof*" stands for "product of other factors evaluated at Ritz value." (This definition uses θ_j where we might like to use the jth desired eigenvalue, but θ_j will approximate it in the case of interest.) The quantity $pof(j)$ gauges the slope of π at θ_j, since

$$|\pi'(\theta_j)| = pof(j)/|\theta_j|.$$

(Unlike $\pi'(\theta_j)$, $pof(j)$ is scale-invariant.) Large $pof(j)$ values signal points where π changes rapidly, warning of ill-conditioning in related computations.

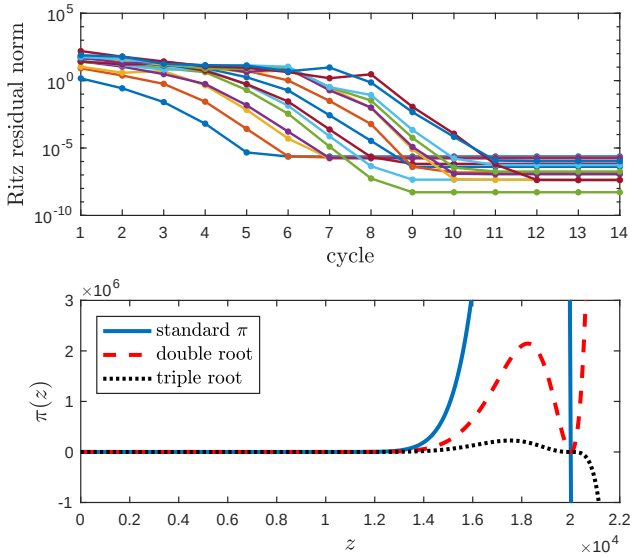


FIG. 7.1. *Example 7, with $d = 15$: The top plot shows the residual norms, cycle by cycle: the large eigenvalue in A limits the attainable accuracy. The bottom plot shows how adding extra roots at the largest harmonic Ritz value tames the polynomial.*

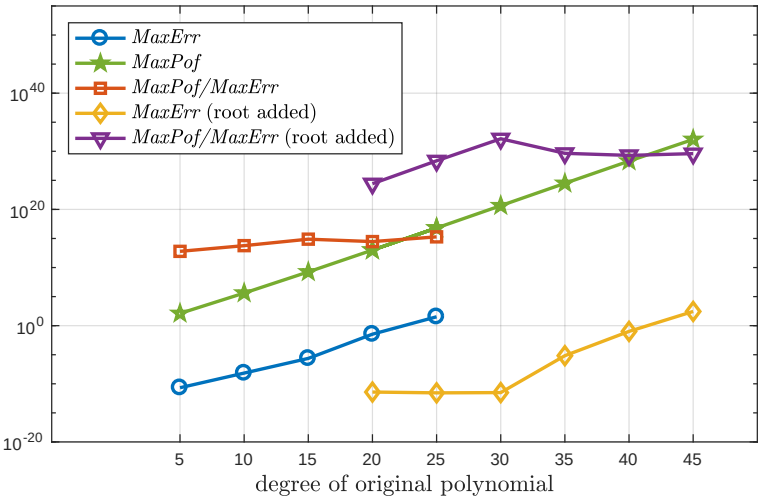


FIG. 7.2. *Example 7, test of instability: A large outlying eigenvalue gives a large pof(j); the attainable accuracy (blue circles) degrades as the degree d increases. An extra root to π at the largest harmonic Ritz value can improve the attainable accuracy for larger d values (yellow diamonds).*

Figure 7.2 shows that as the degree d increases, the accuracy steadily degrades. For this example, the maximum Ritz residual norm grows with the maximum $pof(j)$ value. Let *MaxErr* be the maximum eventual residual norm of the 15 computed eigenvalues and let *MaxPof* be the maximum $pof(j)$ value. For the matrix in Example 7, Figure 7.2 plots *MaxPof* (which occurs at the largest harmonic Ritz value) with stars: it steadily increases as d increases. Once the instability is the main source of error, starting at degree 10, the ratio *MaxPof*/*MaxErr* is on the order of 10^{15}.

To improve stability when some $pof(j)$ is large, we add an additional root at θ_j to π, making θ_j a double root. When θ_j is almost an eigenvalue λ of A, this makes $\pi(\lambda)$

so near zero that even if the component of $\pi(A)v$ in the direction of this eigenvector is off by several orders of magnitude, it is not significant relative to the other terms. If θ_j is a double root of π, then the slope $\pi'(\theta_j) = 0$, suggesting better conditioning. However, the extra root increases the degree of π and the number of matrix-vector products with A needed to apply $\pi(A)$. When is an extra root worth adding, and how many should be included? In the test described above, an extra root added little benefit when $pof(j) \leq 10^4$. In further testing, we have found that a new root is roughly needed for every factor of 10^{14} that *MaxPof* exceeds 10^4. Thus we suggest making θ_j a double root if $pof(j)$ exceeds 10^4, a triple root if $pof(j)$ exceeds 10^{18}, a quadruple root if $pof(j)$ exceeds 10^{32}, etc., incrementing by 10^{14} each time.

Adding Roots for Stability

1. **Setup:** Assume the d roots $(\theta_1, \ldots, \theta_d)$ of π have been computed and then sorted according to the modified Leja ordering [1, alg. 3.1].
2. **Compute** $pof(j)$**:** For $j = 1, \ldots, d$, compute $pof(j) = \prod_{i \neq j} |1 - \theta_j/\theta_i|$.
3. **Add roots:** Compute least integer greater than $(\log_{10}(pof(j)) - 4)/14$, for each j. Add that number of θ_j values to the list of roots. We add the first to the end of the list and if there are others, they are spaced into the interior of the current list, evenly between the occurrence of that root and the end of the list (keeping complex roots together).

Other choices are possible for placing the new roots into the list of roots. We also tried a second Leja sorting with the distance between identical roots defined to be a small amount such as $\alpha|\theta_j|$ for $\alpha = 10^{-15}$. This was sometimes better and sometimes worse than the approach listed above. This topic could use further investigation.

Example 7 (continued) We now apply the procedure just given to Example 7, for increasing values of d. The test adds a root all $d \geq 8$. Figure 7.2 shows *MaxErr* for different degree polynomials with an added root (so the degree of the preconditioner is actually one more than the degree shown in the plot). The accuracy for degree 25 is far better than without the added root (2.8×10^{-12} compared to 3.4×10^{1}). However, even with a double root accuracy is lost for $d > 30$; for large d, *MaxPof*/*MaxErr* is roughly 10^{30}. For $d = 40$ with one root added, $MaxErr = 1.0 \times 10^{-1}$. However, at that point $MaxPof = 2.0 \times 10^{28}$, so according to the plan given above another extra root is needed. With this triple root, *MaxErr* improves vastly to 4.5×10^{-12}. The bottom of Figure 7.1 compares the original $d = 15$ polynomial to those with one and two added roots at the large harmonic Ritz value. The slope of the original π is large at $\lambda = 20{,}000$. Adding a root causes the polynomial to level off briefly there. The polynomial with a triple root is not needed at this degree, but notice how it would add considerable stability near the extreme eigenvalue.

We have tried this procedure of adding roots for realistic problems, including matrices used earlier in this paper and others, and it seems to work well. The number of needed additional roots varies considerably. For example, with the degree $d = 100$, the Af23560 matrix from Example 2 uses 24 added roots (16 double and 4 triple), while the damped polynomial for S1rmq4m1 from Example 5 needs only 1 added root. We do suspect there will be problems for which this procedure does not work; in fact, we have been able to devise such an example by skewing the starting vector against an outlying eigenvalue so severely that the associated harmonic Ritz value is far from that eigenvalue: extra roots in the wrong place do not much help.

8. Double Polynomial Preconditioning. Communication-avoiding methods minimize operations that transfer data across processors (and perform related syn-

chronizations) such as dot products, potentially at the cost of extra communication-free work on local processors. We have already seen (e.g., Table 4.1) that polynomial preconditioning can significantly reduce the number of dot products required to compute eigenvalues. In fact, dot products can be even more significantly reduced by combining two levels of polynomial preconditioning, giving access to very high degree polynomials (which can permit lower subspace dimensions, and hence less memory).

Example 8. We revisit the convection-diffusion matrix from Example 1 of size $n = 640{,}000$, again using Arnoldi(50,20) to compute the $nev = 15$ smallest eigenvalues. Table 8.1 reports results averaged over 10 trials, all of which converge toward all $nev = 15$ of the desired eigenvalues.) Large degree preconditioning polynomials accelerate convergence, in terms of time and dot products. However, two concerns emerge as d increases: construction of π becomes increasingly expensive (e.g., the $d = 150$ computation takes 60,127.8 dot products, 27,941.2 of which come from the GMRES run used to construct π), and larger values of d can cause stability problems. We seek to avoid these limitations, while still reaping the benefits of high degree polynomials.

These observations motivate *Double Polynomial Preconditioning*. Start by generating the GMRES polynomial π_1 of degree d_1 for A. As before, we expect the smallest magnitude eigenvalues of A to be mapped to the eigenvalues of $\pi_1(A)$ nearest 1. Thus define $\tau(\alpha) \equiv 1 - \pi_1(\alpha)$: we seek the smallest magnitude eigenvalues of $\tau(A)$. To compute these smallest magnitude eigenvalues of $\tau(A)$, apply polynomial preconditioning to this matrix, i.e., apply GMRES to $\tau(A)$ to generate a new GMRES polynomial π_2 of degree d_2. Now apply Arnoldi(m, k) to compute the *nev* eigenvalues of $\pi_2(\tau(A))$ nearest 1. The composite polynomial $\pi_2(\tau(\cdot))$ has degree $d_1 d_2$, making use of extremely high degree polynomials more practical.

Example 8 (continued). The bottom half of Table 8.1 shows the effectiveness of double polynomial preconditioning for the convection-diffusion problem. The first column reports the polynomial degrees d_1 and d_2; e.g., $15 \times 20 = 300$ means that τ has degree $d_1 = 15$ and π_2 has degree $d_2 = 20$, so the composite polynomial $\pi_2(\tau(\cdot))$

TABLE 8.1

Example 8 (convection–diffusion, $n = 640{,}000$). Work required for Arnoldi(50,20) to compute the $nev = 15$ smallest magnitude eigenvalues. Double polynomial preconditioning (bottom part of table) can significantly reduce the number of communication-intensive dot products required for standard polynomial preconditioning (top part of table).

degree d or $d_1 \times d_2$	cycles	*mvps* (thousands)	*cost* (thousands)	time (minutes)	dot products (thousands)
Polynomial Preconditioned Arnoldi					
0	7436.1	223.1	42534.1	246.3	17178.2
10	259.0	78.3	1876.8	18.7	574.6
25	84.3	64.2	845.4	11.6	188.6
50	41.2	63.6	612.2	10.0	95.4
100	20.6	64.8	524.1	9.5	57.9
125	16.5	65.3	519.9	9.8	56.3
150	14.0	67.0	535.0	9.9	60.1
Double Polynomial Preconditioning					
$15 \times 20 = 300$	3.8	41.0	273.6	1.7	9.7
$15 \times 40 = 600$	2.0	48.9	314.8	2.0	6.9
$15 \times 50 = 750$	2.0	61.2	392.0	2.5	7.9
$25 \times 40 = 1000$	1.0	51.0	321.0	2.1	5.2
$25 \times 60 = 1500$	1.0	76.5	481.4	3.1	8.0

has degree 300. Because high degree composite polynomials can be formed without the need for much GMRES orthogonalization, the dot products are greatly reduced (but other costs can go up). For composite degree $25 \times 40 = 1000$, only 5,231.2 dot products are needed, a ten-fold reduction from the lowest number given for single polynomial preconditioning. Arnoldi(50,20) needs only one cycle with this high degree polynomial. (In these tests, we only check residual norms at the end of cycles. To further reduce operations, we could check residuals mid-cycle and terminate early.)

Example 9. We revisit the matrix in Example 7. In this case, double polynomial preconditioning can help cure the instability, though this is not guaranteed in general. Let the first polynomial have degree $d_1 = 6$, which has a root near 19,991.2, close enough to $\lambda = 20{,}000$ so that the spectrum of $\pi_2(\tau(A))$ with $d_2 = 20$ does not have an outstanding eigenvalue, and there is no instability: Arnoldi(50,20) finds the $nev = 15$ smallest eigenvalues in two cycles (composite degree $6 \times 20 = 120$). Next, we change to $d_1 = 5$, for which π_1 has a root only at 19,700.5: not close enough to $\lambda = 20{,}000$, so for $d_2 = 20$ the matrix $\pi_2(\tau(A))$ has an outstanding eigenvalue, and no progress is made in 50 Arnoldi cycles. However, the *MaxPof* test described in Section 7 suggests that a double root be added here: that is sufficient to give convergence in two cycles, finding most of the $nev = 15$ desired eigenvalues.

Further investigation is needed of stability for double polynomial preconditioning and whether the same test we applied for one polynomial remains effective here.

9. Conclusions. Polynomial preconditioning can vastly improve eigenvalue calculations for difficult problems, giving the benefit of working with high-degree polynomials in A without requiring high-dimensional subspaces.

We have focused on preconditioning with the GMRES (residual) polynomial, which is easy to compute and adapts to the spectrum of A. Our computational experiments illustrated the success of this method and identified a few scenarios that can be remedied with special handling: using multiple starting vectors for GMRES; damping the GMRES starting vector; adding extra copies of certain roots to enhance stability. While polynomial preconditioning often reduces matrix-vector products for difficult problems, the reduction in vector operations such as dot products is even greater, so this approach holds great promise for communication-avoiding eigenvalue computation on high performance computers. *Double Polynomial Preconditioning* gives access to high degree polynomials in A, and can further reduce dot products. Techniques from [5, 9] can potentially aid parallel implementations.

Future research should include computation of interior eigenvalues, generalized eigenvalue problems, and application to computing stable eigenvalues in matrices that exhibit a significant departure from normality [33, chap. 28]. Stability control for the double polynomial preconditioning should also be investigated.

REFERENCES

[1] Z. Bai, D. Hu, and L. Reichel, *A Newton basis GMRES implementation*, IMA J. Numer. Anal., 14 (1994), pp. 563–581.

[2] C. Beattie, M. Embree, and J. Rossi, *Convergence of restarted Krylov subspaces to invariant subspaces*, SIAM J. Matrix Anal. Appl., 25 (2004), pp. 1074–1109.

[3] C. A. Beattie, M. Embree, and D. C. Sorensen, *Convergence of polynomial restart Krylov methods for eigenvalue computations*, SIAM Review, 47 (2005), pp. 492–515.

[4] F. Chatelin, *Spectral Approximation of Linear Operators*, Academic Press, New York, 1983.

[5] J. Demmel, M. Hoemmen, M. Mohiyuddin, and K. Yelick, *Avoiding communication in sparse matrix computations*, in 2008 IEEE International Symposium on Parallel and Distributed Processing, IEEE, 2008.

[6] J. Duintjer Tebbens and G. Meurant, *Any Ritz value behavior is possible for Arnoldi and GMRES*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 958–978.
[7] M. Embree, *The Arnoldi eigenvalue iteration with exact shifts can fail*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1–10.
[8] R. W. Freund, *Quasi-kernel polynomials and their use in non-Hermitian matrix iterations*, J. Comput. Appl. Math., 43 (1992), pp. 135–158.
[9] M. F. Hoemmen, *Communication-avoiding Krylov subspace methods*. PhD Thesis, EECS Dept., University of California at Berkeley, 2010.
[10] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, Berlin, corrected second ed., 1980.
[11] C. Lanczos, *Chebyshev polynomials in the solution large-scale linear systems*, Proc. ACM, (1952), pp. 124–133.
[12] R. Li, Y. Xi, E. Vecharynski, C. Yang, and Y. Saad, *A thick-restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems*, SIAM J. Sci. Comput., 38 (2016), pp. A2512–A2534.
[13] Q. Liu, R. B. Morgan, and W. Wilcox, *Polynomial preconditioned GMRES and GMRES-DR*, SIAM J. Sci. Comput., 37 (2015), pp. S407–S428.
[14] K. Meerbergen, A. Spence, and D. Roose, *Shift-invert and Cayley transforms for detection of rightmost eigenvalues of nonsymmetric matrices*, BIT, 34 (1994), pp. 409–423.
[15] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl., 154-156 (1991), pp. 289–309.
[16] ———, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1230.
[17] R. B. Morgan and M. Zeng, *A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity*, Linear Algebra Appl., 415 (2006), pp. 96–113.
[18] 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.
[19] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–133.
[20] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[21] H. Rutishauser, *Theory of gradient methods*, in Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, M. Engeli, T. Ginsburg, H. Rutishauser, and E. Stiefel, eds., Birkhauser, Basel, 1959, pp. 24–49.
[22] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.
[23] Y. Saad, *Chebychev acceleration techniques for solving large nonsymmetric eigenvalue problems*, Math. Comp., 42 (1984), pp. 567–588.
[24] ———, *Least squares polynomials in the complex plane and their use for solving sparse nonsymmetric linear systems*, SIAM J. Numer. Anal., 24 (1987), pp. 155–169.
[25] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, second ed., 2003.
[26] Y. Saad, *Numerical Methods for Large Eigenvalue Problems, 2nd Edition*, SIAM, Philadelphia, PA, 2011.
[27] Y. Saad and M. H. Schultz, *GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[28] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[29] A. Stathopoulos, Y. Saad, and K. Wu, *Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods*, SIAM J. Sci. Comput., 19 (1998), pp. 227–245.
[30] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601 – 614.
[31] E. L. Stieffel, *Kernel polynomials in linear algebra and their numerical applications*, U. S. Nat. Bur. Standards, Appl. Math. Ser., 49 (1958), pp. 1–22.
[32] H. K. Thornquist, *Fixed-Polynomial Approximate Spectral Transformations for Preconditioning the Eigenvalue Problem*, PhD thesis, Rice University, 2006. Technical report TR06-05, Department of Computational and Applied Mathematics, Rice University.
[33] L. N. Trefethen and M. Embree, *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*, Princeton University Press, Princeton, NJ, 2005.
[34] K. Wu and H. Simon, *Thick-restart Lanczos method for symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602 – 616.