POLYNOMIAL PRECONDITIONED GMRES AND GMRES-DR

QUAN LIU[†], RONALD B. MORGAN[‡], AND WALTER WILCOX[§]

Abstract. We look at solving large nonsymmetric systems of linear equations using polynomial preconditioned Krylov methods. We give a simple way to find the polynomial. It is shown that polynomial preconditioning can significantly improve restarted GMRES for difficult problems, and the reasons for this are examined. Stability is discussed and algorithms are given for increased stability. Next we apply polynomial preconditioning to GMRES with deflated restarting. It is shown that this is worthwhile for sparse matrices and for problems with many small eigenvalues. Multiple right-hand sides are also considered.

Key words. linear equations, eigenvalues, polynomial preconditioning, GMRES, GMRES-DR, deflation, QCD

AMS subject classifications. 65F10, 15A06

1. Introduction.

We will look at Krylov subspace methods for solving systems of linear equations $Ax = b$, where A is a large nonsymmetric matrix. Polynomial preconditioning has been considered for Krylov subspace methods; see for example [22, 34, 9, 35, 36, 4, 40, 12, 13, 30, 5, 16, 24, 37, 23]. The system of linear equations is converted to

$$p(A)Ax = p(A)b, \tag{1.1}$$

where p is a polynomial. This new system can then be solved with a Krylov subspace method. If a standard preconditioner is available, it can be used also. Let M^{-1} be an approximate inverse for A. With right preconditioning, first precondition the system as $M^{-1}Ax = M^{-1}b$. Then the polynomial preconditioned version is

$$p(M^{-1}A)M^{-1}Ax = p(M^{-1}A)M^{-1}b. \tag{1.2}$$

We will focus on restarted Krylov methods such as GMRES [39]. Polynomial preconditioning for GMRES has been tested, and it has been particularly noted to be useful for parallel processing. However, it has not become a standard approach. Here we argue that polynomial preconditioned GMRES is worth reconsidering. Difficulty in determining the polynomial may have decreased its use. Many methods require information about the spectrum in order to build the polynomial. Here we find a polynomial in a very simple way that corresponds to a minimum residual. It is shown that this approach is easy and effective for low degree polynomials. For higher degree polynomials, more stable approaches are given.

We will also combine polynomial preconditioning with a modification of GMRES that has deflation of eigenvalues (see for example [26, 20, 10, 6, 27, 32] and references in [15]). Small eigenvalues can slow the convergence of Krylov methods, especially for

*The first author was partially supported by a grant from the Baylor University Research Committee. The second author was supported by the Baylor University Sabbatical Program and by NSF grant DMS-1418677.

[†]Department of Physics, Baylor University, Waco, TX 76798-7316. (quan_liu@mfe.berkeley.edu).

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

[§]Department of Physics, Baylor University, Waco, TX 76798-7316. (Walter_Wilcox@baylor.edu).

restarted methods such as GMRES. Deflating eigenvalues for GMRES requires eigenvectors corresponding to the eigenvalues that are to be eliminated. Once eigenvectors are developed, they can be used to build a preconditioner or they can be put into the subspace. We will consider the latter approach, specifically the method GMRES with deflated restarting (GMRES-DR) [27] which both solves linear equations and computes eigenvalues and eigenvectors. Deflating small eigenvalues can significantly improve convergence. However, if there are too many small eigenvalues, GMRES-DR may not be able to compute all of the corresponding eigenvectors. For this case and also for very sparse matrices, we add polynomial preconditioning to GMRES-DR.

Section 2 of the paper has polynomial preconditioning for restarted GMRES. We discuss why and when polynomial preconditioning is effective. The simple polynomial is given, then it is shown that polynomial preconditioning can be very beneficial for difficult problems. Stability is discussed in Section 3, along with ways to implement the polynomial for greater stability. The new method polynomial preconditioned GMRES-DR is given in Section 4. Then Section 5 has solution of multiple right-hand sides.

2. Polynomial Preconditioned GMRES. Restarting is usually necessary with GMRES to control the orthogonalization expense and storage. However, it limits the degree of the subspace and thus slows the convergence. GMRES(m) is GMRES that restarts when the subspaces reach dimension m. One cycle of GMRES(m) uses the Krylov subspace $Span\{r_0, Ar_0, A^2r_0, A^3r_0, \ldots, A^{m-1}r_0\}$. Because of the structure of this subspace with powers of A, the approximate solution that is extracted from it can be written as $\hat{x} = \pi(A)r_0$, where π is a polynomial of degree $m-1$. Note that higher degree polynomials can be much better for difficult problems.

We now consider adding polynomial preconditioning. Even though the subspace for polynomial preconditioned GMRES(m) is of only dimension m, the same as for regular GMRES(m), the polynomial for forming the approximate solution is higher degree. Let p be a polynomial of degree $deg-1$. Let $s(A) \equiv p(A)A$, so s is degree deg. The approximate solution is now

$$\hat{x} = \pi(s(A))p(A)r_0, \tag{2.1}$$

so the degree of the combined polynomial $(\pi \circ s)p$ actually used to form the approximate solution is $(m-1) * deg + (deg-1) = m * deg - 1$. The minimization used to find the approximate solution is over the subspace of dimension m, not an entire Krylov subspace of dimension $m * deg$. Nevertheless, this high degree polynomial shows potential power of polynomial preconditioned GMRES.

Some of the methods that have been studied generate a Chebyshev polynomial [34, 4, 5]. For a symmetric, positive definite problem, this requires estimates of the smallest and largest eigenvalues. With complex spectra, determining the Chebyshev polynomial is difficult. Least squares polynomials [36, 37, 40, 12] have also been proposed, and they also use estimates of the spectrum. We speculate that the complication in finding a polynomial may have discouraged use of polynomial preconditioning and that a simpler way is needed.

2.1. The minimum residual polynomial. In this paper, we will use a polynomial related to the minimum residual polynomial (the GMRES polynomial). Let the Krylov subspace be dimension m. Let the approximate solution from the Krylov subspace be $\hat{x} = p(A)b$, where p is a polynomial of degree $deg-1$ of less. Just above, we called this polynomial π instead of p, but here we are finding the polynomial for

use in Equation (1.1). The residual vector is $r \equiv b - A\hat{x}$. Define polynomial q to be $q(\alpha) \equiv 1 - \alpha p(\alpha)$ and note that q is degree deg or less and $q(0) = 1$. The minimum residual solution is the $\hat{x}$ that minimizes $||b - A\hat{x}||$, and the minimum residual polynomial is the corresponding polynomial q that minimizes $||q(A)b|| = ||b - Ap(A)b|| = ||b - A\hat{x}||$. While the GMRES polynomial q is important, the polynomial p is needed for the right-hand side of Equation (1.1). It can be difficult to determine the coefficients representing p accurately, and this can lead to numerical instability (see Section 3).

We next give a way of finding the p that corresponds to the minimum residual polynomial q. It is also used in [2] for symmetric matrices. This way of finding the polynomial does not seem to have been previously explored in the polynomial preconditioning literature. It was used to find a polynomial for an iteration in [19]. This approach is simple and surprising effective. Let

$$Y = [b, Ab, A^2b, \ldots, A^{deg-1}b]. \tag{2.2}$$

Solving

$$(AY)^T AYg = (AY)^T b \tag{2.3}$$

for g gives the coefficients of the polynomial p. Note that if one desires a general purpose minimum residual polynomial, instead of one tailored for the specific right-hand side vector b, then a random vector can be used in its place. This procedure obviously can be unstable due to ill-conditioning of Y. However it will be shown to be effective for low degree polynomials in the experiments. In Section 3, we give more stable methods.

Sometimes polynomial preconditioning for restarted GMRES is very helpful, but the effectiveness varies. In this section, we try to understand the reasons for this. We begin with three reasons why polynomial preconditioning can be worthwhile.

Point 1: Polynomial preconditioning can be very cost-effective if the several matrix-vector products in $p(A)A$ can be implemented more efficiently than if the matrix-vector products are done singly as in standard GMRES. This depends on the application and how the matrix-vector product is implemented, and we will not consider it further.

Point 2: For restarted methods, polynomial preconditioning can be helpful for difficult problems that need a high degree polynomial. Often small eigenvalues are the reason for the difficulty. As seen in Equation (2.1), polynomial preconditioning allows much higher degree polynomials.

Point 3: Polynomial preconditioning reduces orthogonalization expense relative to matrix-vector products. Each new GMRES basis vector involves deg matrix-vector products and only one vector to be orthogonalized against previous ones. Because of this, polynomial preconditioning has often been studied in the context of parallel processing (see [38] and its references). Polynomial preconditioning is particularly useful for sparse matrices, because then the orthogonalization expense is significant versus the cost for the matrix-vector product. However, if an elaborate standard preconditioner M^{-1} is used, then polynomial preconditioning is less likely to be helpful, because the orthogonalization expense will be less significant compared to the matrix and preconditioner cost. Also, if the standard preconditioning is effective and leads to quick convergence, then Point 2 will not apply.

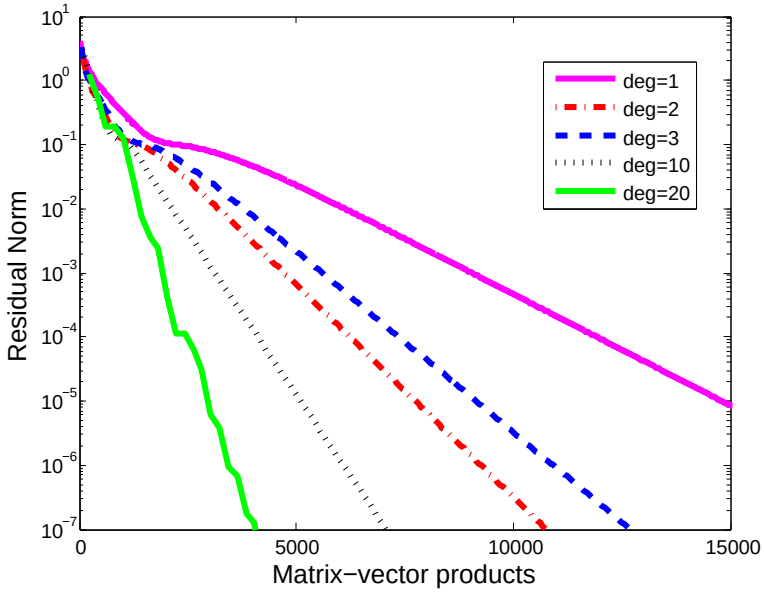

FIG. 2.1. *Convergence of standard restarted GMRES(20) (denoted deg=1) compared to polynomial preconditioned GMRES.*

2.2. Examples. Some of the test matrices will be used for several examples throughout the paper. The first of these will be called the Bidiagonal Matrix. It is bidiagonal of size $n = 5000$ with diagonal elements $0.1, 0.2, 0.3, \ldots 0.9, 1, 2, 3, \ldots 4990, 4991$. The superdiagonal element are all the same and are 0.2 for the first example but will change for the second example.

The first example demonstrates Points 2 and 3.

Example 1. We use the Bidiagonal Matrix just described. This is a fairly difficult problem for restarted GMRES because of the small eigenvalues and the wide range of eigenvalues: the ratio of largest to smallest eigenvalue is $4991/0.1 = 49,910$ (while some matrices from test packages are even more difficult than this one, they generally are preconditioned before restarted GMRES is applied). The right-hand side is chosen random normal(0,1), and we use GMRES(20). The initial phase of determining the polynomial also produces an approximate solution, however in these tests we do not use it to begin solving the system of equations. Figure 2.1 shows that convergence is much faster in terms of matrix-vector products for polynomial preconditioned GMRES (PP-GMRES). $Deg = 1$ corresponds to no polynomial preconditioning. $Deg = 10$ means the polynomial s is degree 10 and so p is degree 9. With this preconditioning, convergence is five times faster. As mentioned in Point 2, the degree of polynomial $(\pi \circ s)p$ used to form the approximate solution is higher with polynomial preconditioning. For instance, PP-GMRES(10) with $deg = 10$ uses degree 199 polynomials instead of degree 19 for non-polynomial preconditioned. The higher degree polynomials are better able to deal with the small eigenvalues. The next paragraph shows that the polynomial preconditioned matrix has an improved spectrum.

The polynomial $s(\alpha) = \alpha p(\alpha)$ has value zero at zero and should be near 1.0 over the rest of the spectrum of the matrix A. This is difficult for a low degree polynomial. Figure 2.2 shows the polynomials of degree 3 and 10. The polynomials have the most difficulty being near one at the small eigenvalues. However, they do improve the ratio of largest to smallest eigenvalue. Matrix A has ratio $49,910$, while for polynomial

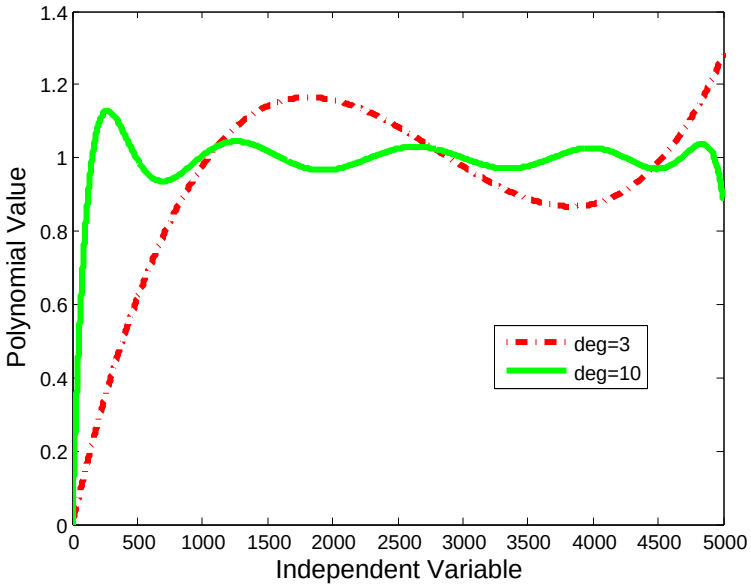

FIG. 2.2. *Polynomials of degree 3 and 10 used for polynomial preconditioning.*

TABLE 2.1
Expenses for PP-GMRES(20) (means only reaches relative residual norm of* $1.2 * 10^{-8}$*)*

deg	1	2	3	5	10	20
mvp's	18,140	11,196	9521	6271	3637	4851*
vector ops	493,000	159,000	94,200	40,000	13,500	11,600*

preconditioning with $deg = 3$, the ratio becomes $1.273/(1.521 * 10^{-4}) = 8367$. The ratio with degree of s equal to 10 is $1.129/(1.185 * 10^{-3}) = 952.9$.

The expense of the method can be reduced much more than is shown by the count of matrix-vector products. With polynomial preconditioning there are several matrix-vector products used for each vector, so orthogonalization expense is reduced relatively more than the matrix-vector products. Table 2.1 shows the number of matrix-vector products and the number of vector operations of length n (such as dot products and daxpy's) that are used for PP-GMRES(20) with different choices of deg. The tolerance is residual norm relative to the initial residual below 10^{-8}. The reduction in vector operations from 493 thousand for regular GMRES(20) down to below 14 thousand with polynomial of degree 10 is quite significant. We note that the polynomial of degree 20 is not as effective. While vector operations go even lower, the matrix-vector products go up. Also the accuracy only reaches relative residual norm of $1.2 * 10^{-8}$, hence the *'s in the table. This is due to instability in the polynomial implementation and is discussed in Section 3.

Example 2. We now use the Bidiagonal Matrix with superdiagonal elements of 0.3. This may appear to be a small change, but it increases the non-normality of the matrix and makes it a more difficult problem. We also switch to GMRES(40), but even with these larger subspaces, the method does not converge without polynomial preconditioning; see Figure 2.3. Here polynomial preconditioning is essential to the success. A more expensive alternative would be to use even larger subspaces for

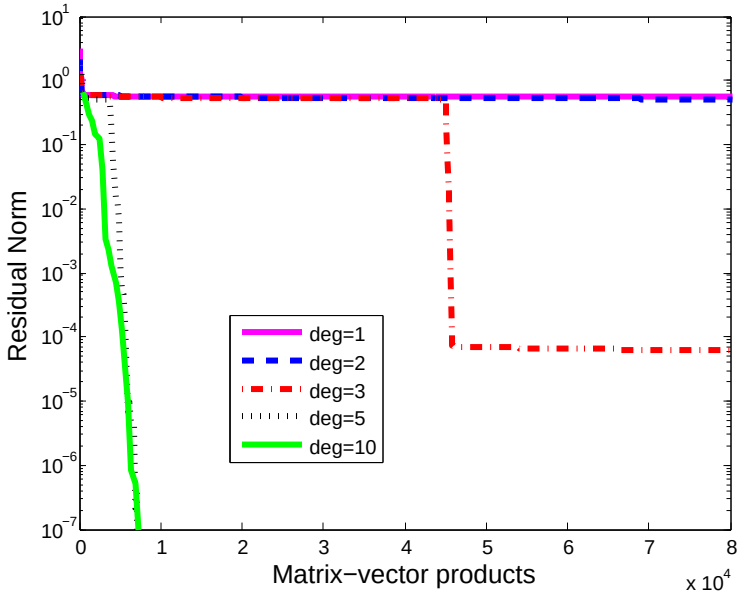

FIG. 2.3. *Convergence of standard restarted GMRES(40) compared to polynomial preconditioned GMRES for more highly non-normal matrix.*

TABLE 2.2
Expenses for PP-GMRES - matrix without small eigenvalues

deg	1	2	3	5	10
mvp's	280	249	249	312	421
vector ops	7621	3514	2431	1960	1576

GMRES.

We next give an example for which polynomial preconditioning is not as helpful.

Example 3. We now adjust the Bidiagonal Matrix to eliminate the smallest eigenvalues. The diagonal elements are $10, 11, 12, \ldots 5009$ and superdiagonal elements are again 0.2. We go back to GMRES(20). Table 2.2 has the number of matrix-vector products and vector op's. This problem is easy enough that high degree polynomials are not as helpful as they were for the previous problems. Thus the polynomial preconditioning either does not reduce the number of matrix-vector products very much or raises it. Vector op's are still significantly reduced.

We now give the next test matrix, which we will call the Helmholtz Matrix. The matrix comes from finite difference descretization of the one-dimensional differential equation

$$-u'' - c^2 u = f,$$

with boundary conditions

$$u(0) = 0 \text{ and } u(1) = 0.$$

This is a very simple version of a Helmholtz equation with wave number c.

Example 4. We first let $c = 10$ and this gives a symmetric matrix with three negative eigenvalues. We let $n = 500$, so the subinterval length for the descretization

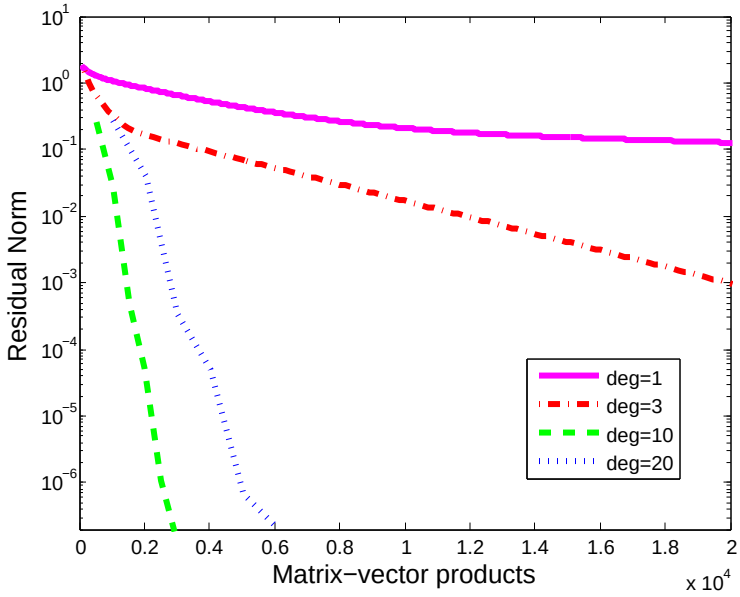

FIG. 2.4. *Convergence for simple 1-D Helmholtz matrix with wave number of 10 and* $n = 500$.

TABLE 2.3
Expenses for PP-GMRES - indefinite matrix with 3 negative eigenvalues

deg	1	2	3	5	10	20
mvp's	404,000	68,582	49,533	26,866	3025	6045
vector ops	23,100,000	2,010,000	985,000	332,000	20,200	23,400

is $h = 1/501$. The right-hand side is again generated random normal. Figure 2.4 and Table 2.3 compare GMRES(50) with and without polynomial preconditioning and show that there can be a great improvement using polynomials. With $deg = 10$, the number of vector operations is improved by a factor of over 1000. The polynomials s of degree 3, 10 and 20 are shown in Figure 2.5. Then Figure 2.6 shows a close-up of these polynomials evaluated at the small eigenvalues. Also shown is the identity polynomial that corresponds to no polynomial preconditioning. The higher degree polynomials shift the small eigenvalues further away from the origin.

Example 5. Next we use the Helmholtz Matrix, but with a much higher wave number $c = 1000$. Also, n is increased to 1000. This gives a much more indefinite matrix with 333 negative eigenvalues and 667 positive ones. This problem is well known to be difficult and often multigrid preconditioning is used for its solution [8, 21, 11]; see the next example. We will show that polynomial preconditioning can help, but must be used with caution, because it can also hurt convergence. GMRES(50) is used. In Figure 2.7, a polynomial of degree 15 works very well, but polynomials of degree 2 and 20 are worse than if there is no polynomial preconditioning. Figure 2.8 shows these polynomials and it can be seen that the degree 15 polynomial converts the spectrum to be almost definite (there are six negative eigenvalues of $p(A)A$). However, the degree 20 polynomial gives a much poorer spectrum that is indefinite and spreads from approximately -2 to 4. This polynomial suffers from instability in its computation. The degree 2 polynomial produces a spectrum for $p(A)A$ that is

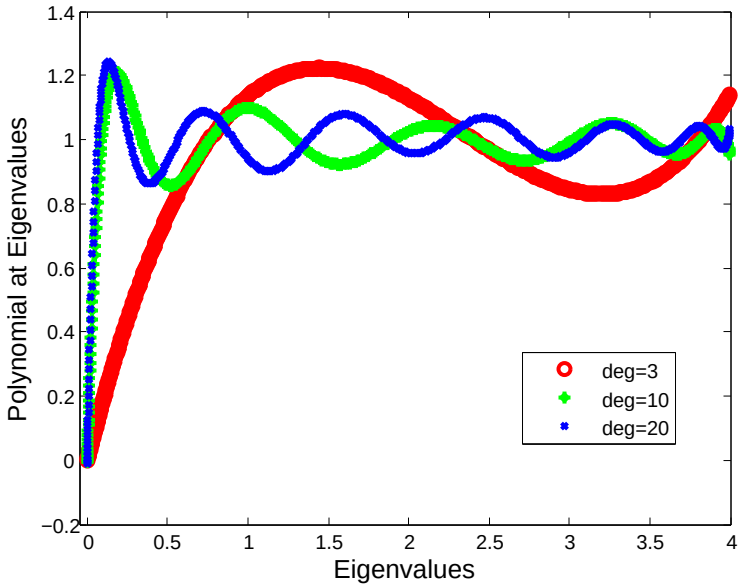

FIG. 2.5. *Polynomials plotted at eigenvalues of Helmholtz matrix; degree 3, 10 and 20*

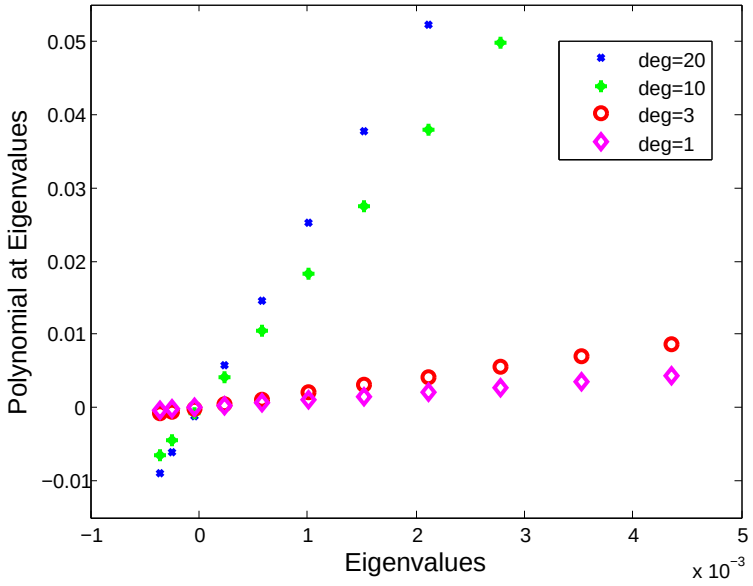

FIG. 2.6. *Closeup of polynomial values at eigenvalues for degree 3, 10, 20 and also with no polynomial preconditioning (deg = 1).*

somewhat less indefinite with 129 negative eigenvalues, but there are many eigenvalues mapped near zero (43 eigenvalues are within 0.01 of zero, compared to four in the spectrum of A). The degree 15 polynomial happens to not pass through the x-axis near an eigenvalue A, which would create a very small eigenvalue of $p(A)A$. We tried polynomials of other degrees and the results vary greatly, so there is a certain amount of randomness for this example.

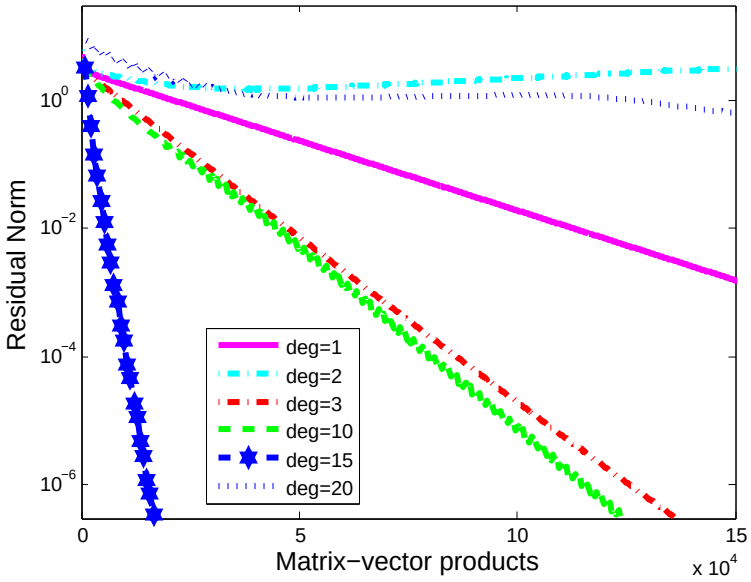

FIG. 2.7. *Convergence for simple 1-D Helmholtz matrix with wave number of 1000 and* $n = 1000$.

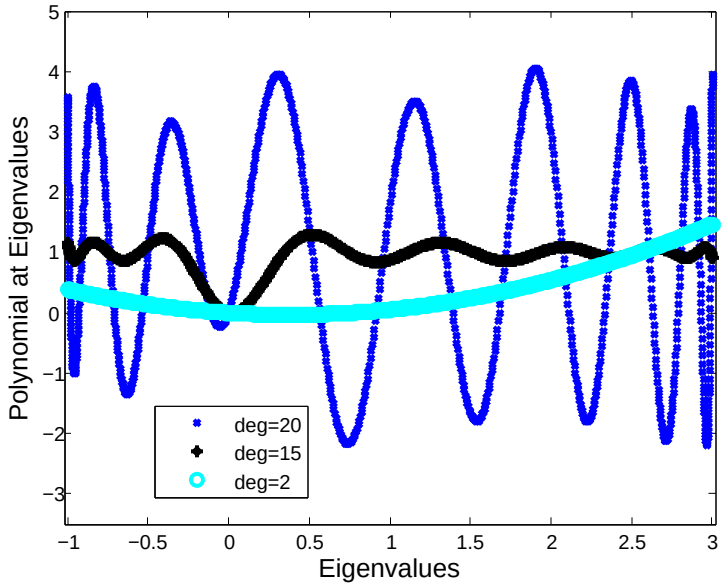

FIG. 2.8. *Polynomials plotted at eigenvalues of Helmholtz matrix with wave number of 1000; degree 2, 15 and 20*

2.3. With standard preconditioning. As mentioned in the Introduction, one of the strengths of polynomial preconditioning is that it can be used along with standard preconditioning. Many real world linear equations problems are very difficult and require standard preconditioning to be tractable. If even after that preconditioning the problem is still challenging for restarted GMRES, then polynomial preconditioning may assist.

TABLE 2.4
Helmholtz matrix with wave number of 1000. Standard preconditioning combined with polynomial preconditioning.

deg. of poly. prec.	mvp's	vector ops	GMRES cycles	GMRES iterations
1	72,750	4,150,000	1455	72,750
3	40,926	773,000	271	13,550
5	25,360	288,000	101	5050
10	12,544	71,500	25	1250
15	3784	15,400	5	250
20	2041	5,940	2	100

Example 6. The matrix is the same as in the previous example, the Helmholtz Matrix with a large wave number $c = 1000$. We use right-preconditioning as in Equation (1.2). A preconditioner is suggested in [11] that uses the matrix that corresponds to the differential equation $-u'' - (1-i)c^2u = f$. This matrix is then the same as the matrix A except that it is shifted by a imaginary multiple of the identity matrix. The systems of linear equations for the preconditioning with this matrix are solved with a multigrid approach (since our goal is to show effectiveness of the polynomial preconditioning, not to evaluate standard preconditioners, we simulate the multigrid solution). This shifted matrix is easier to solve with multigrid than the original matrix. Table 2.4 has results. The first line with no polynomial preconditioning (degree of poly is 1) shows that the standard preconditioning needs 73 thousand matrix-vector products and preconditioning solves. This is down from 325 thousand matrix-vector products with no preconditioning. However, whether this reduces the overall expense depends on the cost of the multigrid solve. Next, adding on the polynomial preconditioning gives remarkable results. With the degree 20 polynomial, the number of matrix-vector products is reduced to just over two thousand. Also, this is more work than really needed, since the method converges early in the second cycle of GMRES(50), but the results are shown with the cycle finishing up. If full GMRES is used, it takes only 53 iterations with 1100 matrix-vector products. For this example, the combination of standard and polynomial preconditioning creates a better method.

3. Stability. In the tests in the previous section, we have seen that the polynomial of degree 20 can be unstable. In this section, we first discuss other ways of finding the coefficients of the polynomial p that corresponds to the GMRES polynomial q. Then methods are given that avoid finding the standard coefficients.

3.1. Computation of the polynomial coefficients. The method used in Section 2 for computing the coefficients of the polynomial could be called a Normal Equations approach. The first alternative will be called the QR Factorization approach. It uses the power basis from (2.2), but instead of Equation (2.3), it solves

$$min||b - AYg|| \tag{3.1}$$

with a QR factorization of AY.

The next way of finding the coefficients of polynomial p corresponding to a minimum residual solution is to run *deg* iterations of GMRES and compute the harmonic Ritz values [25, 31] which are also the roots of the GMRES residual polynomial q. Then we find the coefficients of the GMRES polynomial. From these, the coefficients of p can be determined. (In Matlab, apply the *poly* function to the harmonic

Ritz values to get coefficients of q, then scale so that the last coefficient is one, so $q(0) = 1$. The negatives of the first deg coefficients represent p). This will be called the Harmonic Ritz approach.

Another way of finding the coefficients of p is given by Nachtigal, Reichel and Trefethen in [29], and we call it the NRT approach. See Section 4 of [29] for details of computing p. They use the polynomial for a Richardson iteration instead of for polynomial preconditioning.

Example 7. We use the Bidiagonal Matrix with superdiagonals of 0.2 and compare the four methods for computing the coefficients of the polynomial p. They work the same for polynomials of degree 10 or less, so we will compare for degree 20. All ways are somewhat unstable for degree 20. Example 1 used the Normal Equations approach. While the residual norm for the polynomial preconditioned problem in Equation (1.1) continues to improve, the residual norm for the solution of $Ax = b$ stalls out after reaching relative residual norm of $1.2 * 10^{-8}$. The QR Factorization approach of Equation (3.1) reaches relative residual norm of only $2.8 * 10^{-3}$. Tests with other matrices gave the same surprising result of the normal equations giving more accurate solutions than when the polynomial coefficients are computed with QR factorization. Sometimes the initial convergence is faster when using the QR factorization, but the final result is less accurate. This should be the subject of future study. Next with the Harmonic Ritz approach, the method stalls at relative residual norm of $5.5 * 10^{-7}$. The NRT way gives similar results with relative residual norm reaching $5.7 * 10^{-7}$.

Since none of the ways of computing the polynomial p works ideally for higher degree polynomials, we next give methods that avoid finding the regular coefficients of p. While a standard polynomial is represented as a combination of monomials, now p will be represented as a combination of certain polynomials. First we use Arnoldi polynomials.

3.2. Arnoldi polynomials. We compute Arnoldi [3, 38] basis vectors and find coefficients that combine these basis vectors to produce the minimum residual solution. These coefficients come from an intermediate step in the standard GMRES solution. Then in order to implement $p(A)z$, for some vector z, it is necessary to apply the same Arnoldi recurrence. It is as if we are running Arnoldi with starting vector z, but instead of computing scalars that lead to orthonormal vectors, we use the same recursion coefficients as were used in the original Arnoldi iteration that found the polynomial information. To summarize, in the first phase we generate Arnoldi polynomials and express the polynomial p as a combination of these. Then to implement p, the same Arnoldi polynomials are applied to vector z. The vectors thus formed are combined to get $p(A)z$. In [18, 17], polynomials from the first cycle of GMRES are similarly reused, although for subsequent cycles of GMRES rather than for a polynomial preconditioner. The same approach is used in [42] for polynomial preconditioning of the eigenvalue problem.

Use Arnoldi basis vectors to implement $p(A)z$

1. *Initial formulation of p.*
   Compute the standard Arnoldi recurrence with starting vector $b/||b||$: $AV_{deg} = V_{deg+1}H_{deg+1,deg}$. The columns of V_{deg} are the Arnoldi basis vectors.
   Let e_1 be the first coordinate vector of length $deg + 1$. Solve $\min||e_1 - H_{deg+1,deg}g||$ for g (this is most of the way towards finding the GMRES solution to $Ax = b/||b||$). The entries of g are the scalars that represent the

polynomial p in terms of the Arnoldi basis.

2. *Implement $y = p(A)z$, for any vector z.*
   Apply the same Arnoldi recurrence from Part 1 to z, using the same values from $H_{deg+1,deg}$ computed in Part 1. Combine the basis vectors thus generated with coefficients from g:
   $y = g_1 * z$
   $w_1 = z$
   For $j = 1 : deg - 1$
   $t = A * w_j$
   $t = t - \sum_{i=1}^{j} h_{ij} * w_i$
   $w_{j+1} = t/h_{j+1,j}$
   $y = y + g_{j+1} * w_{j+1}$
   End

One can consider the polynomials that correspond to the Arnoldi vectors generated by the Arnoldi iteration. For example, the second Arnoldi vector (the second column of V_{deg}) is $v_2 = (1/h_{21})(Av_1 - h_{11}v_1)$ and the corresponding polynomial of A is $\tau(A) = (1/h_{21})(A - h_{11}I)$. When these Arnoldi polynomials are applied to the starting vector v_1, they produce orthogonal vectors. In the algorithm above, we apply these same polynomials to another vector z in Step 2. However, the w_j vectors that are produced are not orthogonal. The hope is that they will be close to orthogonal or at least further from being linearly dependent than the power method vectors used for the basis in Equation (2.2).

The number of matrix-vector products to compute $p(A)z$ is $deg - 1$ (same as before). The number of vector operations is about $\frac{deg^2}{2} + 1.5deg$. This extra cost will sometimes be significant.

3.3. Newton polynomials with Ritz value shifts. Motivated by the significant costs for forming the w_j vectors in Step 2 of the previous algorithm, we next propose a different way to build a basis for representing the polynomial. Here we use a Newton polynomial approach. The basis vectors in the first phase of determining the polynomial are not orthogonal as Arnoldi vectors are, but the later phase of computing $p(A)$ times a vector requires less vector operations.

We apply Arnoldi with starting vector v_1 (the normalized version of b) out to a dimension $deg-1$ subspace. Then we compute the $deg-1$ Ritz values, θ_i, and reorder with the modified Leja ordering [7]. These are used to build this basis for a Krylov subspace:

$$\{v_1,\ (A-\theta_1 I)v_1,\ (A-\theta_2 I)(A-\theta_1 I)v_1,\ \ldots \\ (A-\theta_{deg-1}I)(A-\theta_{deg-2}I)\cdots(A-\theta_2 I)(A-\theta_1 I)v_1\} \quad (3.2)$$

The polynomial p is then represented in terms of this basis. A similar approach is used in [7] to cheaply build a basis for a modified GMRES. See [33] for further investigation including on the conditioning of Newton polynomial bases.

This choice in some way resembles the Arnoldi process. For Newton, the θ_i's are determined first, then used to build the basis. The Arnoldi basis has a similar form

to (3.2), but changes the θ_i values for every vector. So the basis is

$$\{v_1,\ (A-\theta_1^1 I)v_1,\ (A-\theta_2^2 I)(A-\theta_1^2 I)v_1,\ \ldots \\ (A-\theta_{deg-1}^{deg-1} I)(A-\theta_{deg-2}^{deg-1} I)\cdots(A-\theta_2^{deg-1} I)(A-\theta_1^{deg-1} I)v_1\}, \quad (3.3)$$

where the superscripts show the θ_i's are changing for each vector. The θ_i^{deg-1} values used for the last vector of (3.3) are the same as the θ_i values in (3.2). So the final Newton vector is the same as the last Arnoldi vector in exact arithmetic. While the Newton basis is not orthogonal, the final vector is orthogonal to the previous ones.

Use Ritz values and Newton basis vectors to implement $p(A)z$

1. *Find Ritz values.*
   Run standard Arnoldi iteration with subspace of dimension $deg-1$. Compute the Ritz values, $\theta_1, \ldots \theta_{deg-1}$ (so generate the Arnoldi recurrence with starting vector b: $AV_{deg-1} = V_{deg} H_{deg,deg-1}$, and compute the eigenvalues θ_i of $H_{deg-1,deg-1}$). Reorder the Ritz values with the modified Leja ordering [7].
2. *Initial formulation of p.*
   $v_1 = b/||b||$
   $j = 1$
   While $j < deg$
   $t = A * v_j; \quad d_j = t$
   if θ_j real:
   $t = t - \theta_j * v_j; \quad \gamma_{j+1} = ||t||; \quad v_{j+1} = t/\gamma_{j+1}$
   $j = j + 1$
   if θ_j imaginary:
   $t = t - real(\theta_j) * v_j; \quad \gamma_{j+1} = ||t||$
   $s = A * t; \quad d_{j+1} = s/\gamma_{j+1}$
   $s = s - real(\theta_j) * t + (imag(\theta_j))^2 * v_j$
   $v_{j+1} = t/\gamma_{j+1}$
   $\gamma_{j+2} = ||s||; \quad v_{j+2} = u/\gamma_{j+2}$
   $j = j + 2$
   End
   $d_{deg} = A * v_{deg}$
   Let D be the matrix with columns d_1 through d_{deg} and note that $D = A * V_{deg}$.
   Solve $D^T D g = D^T v_1$, for g.
3. *Implement $y = p(A)z$, for any vector z.*
   $y = g_1 * z$
   $w_1 = z$
   $j = 1$
   While $j < deg$
   $t = A * w_j$
   if θ_j real:
   $t = t - \theta_j * w_j; \quad w_{j+1} = t/\gamma_{j+1}$
   $y = y + g_{j+1} * w_{j+1}$
   $j = j + 1$
   if θ_j imaginary:
   $t = t - real(\theta_j) * w_j$
   $s = A * t$
   $s = s - real(\theta_j) * t + (imag(\theta_j))^2 * w_j$
   $w_{j+1} = t/\gamma_{j+1}; \quad y = y + g_{j+1} * w_{j+1}$

TABLE 3.1
Bidiagonal Matrix. Stability and expense with three bases for representing the polynomial (means only reaches relative residual norm of $1.2 * 10^{-8}$, - means cannot form polynomial)*

	deg	3	10	20	30	40	50
Power	rel accur	4.1e-14	3.4e-10	1.2e-8	5.5e-9	5.5e-9	-
	mvp's	9521	3637	4851*	9675	12,094	-
	vector ops	94,200	13,500	11,600*	18,900	21,000	-
Arnoldi	rel accur	1.4e-14	2.1e-14	3.9e-14	5.4e-14	6.1e-14	6.7e-14
	mvp's	9521	3637	3648	3064	4084	3102
	vector ops	116,000	33,700	47,100	53,800	91,300	85,100
Newton	rel accur	9.7e-15	3.6e-14	1.2e-13	2.3e-13	3.0e-13	3.5e-13
	mvp's	9523	3646	3667	3093	4123	3151
	vector ops	107,000	20,100	16,100	13,100	17,300	14,700

$w_{j+2} = s/\gamma_{j+2}$; $\quad y = y + g_{j+2} * w_{j+2}$
$j = j + 2$
End

Example 8. We compare determining the polynomial p with the original basis involving powers of A to using the Arnoldi and Newton bases. GMRES(20) is applied to the Bidiagonal Matrix with superdiagonal elements 0.2. The method is now very stable. Table 3.1 has results for the three bases. The table gives the minimum relative residual norm that the solution attains and has the cost for solving to relative residual norm of $1. * 10^{-8}$ (the number of matrix-vector products and of vector operations). Using the Arnoldi basis allows for relative residual norm under $1. * 10^{-13}$ even for a degree 50 polynomial. It also requires about a third as many matrix-vector products as does the power basis method for polynomials of degree 30 and 40. The vector operations count gets large for the high degree polynomials. We next examine the conditioning of the matrix W whose columns w_j are generated while computing $p(A)z$ in the algorithm above. The matrix V found in Step 1 of the algorithm is orthonormal and has ratio of smallest to largest singular value of 1. W is not orthogonal because is has a different starting vector. However for polynomial of $deg = 20$, the ratio of smallest to largest singular values of W does not get very small. It hits a minimum of $1.7 * 10^{-2}$. The ratio for the power basis quickly approaches zero. Next for the Newton polynomial basis approach, results in Table 3.1 show that this approach is almost as stable as the Arnoldi basis, but uses far less vector operations for higher degree polynomials. The minimum ratio of smallest to largest singular values of W is $5.6 * 10^{-3}$. This is not much worse than the minimum value for Arnoldi basis.

Example 9. For the Helmholtz Matrix with $c = 10$ and $n = 500$, the degree 10 polynomial was better than degree 20 in Example 4. The Newton basis gives much better results for degree 20. Only one cycle of GMRES(50) is needed and this takes 1059 matrix-vector products and 6710 vector operations. This compares well with the 6045 matrix-vector products and 17,500 vector operations with the power basis. Figure 3.1 shows the convergence. Power and Newton give the same results for polynomial of degree 10. Newton with degree 20 polynomial is considerably better than with the degree 10 polynomial.

Example 10. We now use the Helmholtz Matrix with $c = 1000$ and $n = 1000$.

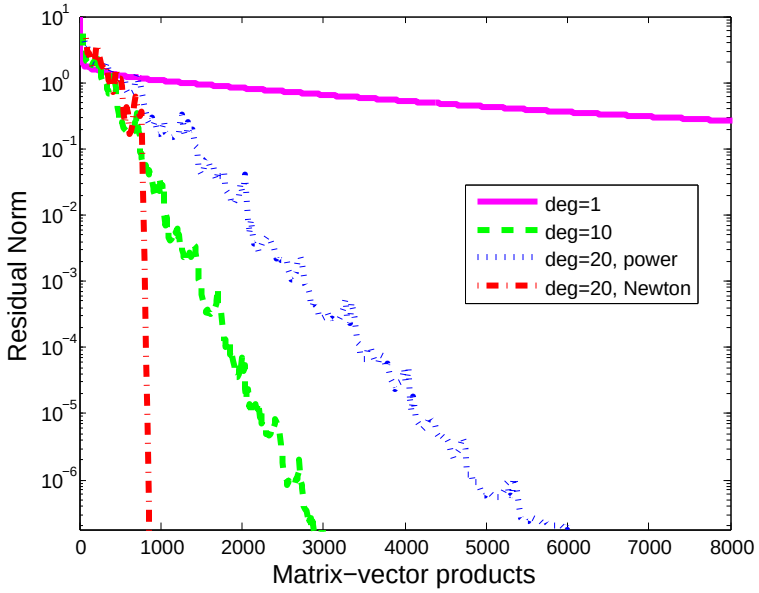

FIG. 3.1. *Convergence for Helmholtz Matrix with wave number of 10 and $n = 500$. Comparison of power polynomials to Newton.*

TABLE 3.2
Stability and expense for the Helmholtz Matrix with $c = 1000$

	deg	20	25	30	35	40
Power	rel accur	6.3e-6	3.0e-9	1.1e-8	5.0e-9	2.6e-8
	mvp's	1,812,850	26,320	111,133	36,840	164,161
	vector ops	6,980,000	86,600	323,000	97,400	399,000
Arnoldi	rel accur	2.1e-13	6.9e-14	3.4e-14	3.7e-14	3.8e-14
	mvp's	243,282	98,878	27,077	22,750	12,085
	vector ops	3,500,000	1,610,000	499,000	472,000	278,000
Newton	rel accur	9.3e-14	7.1e-14	3.6e-14	4.4e-14	4.7e-14
	mvp's	243,321	98,902	27,106	22,866	12,124
	vector ops	1,400,000	515,000	132,000	106,000	55,300

Unlike Example 7, there is no multigrid preconditioning. Table 3.2 has some of the results, and we discuss other degree polynomials. The results of using Newton instead of power start to differ at degree 14. The Newton results are very inconsistent for polynomials of degree 20 and less. For degree 10 through 20, the least number of matrix-vector products is 44 thousand and the most is 1.2 million. Results are similarly inconsistent for the power basis, with sometimes one being much better and sometimes the other. However, for higher degree polynomial, with degree from 30 to 40, the Newton basis gives consistently good results. The number of matrix-vector products is between 9700 and 27,106. With the power basis, results are inconsistent for degree 30 up to 40. The number of matrix-vector products ranges from 54 thousand to near a million. For this difficult problem, fairly high degree polynomials and the Newton basis are needed for the polynomial preconditioning to be consistently effective and low in cost.

The Newton basis had been very stable in the previous examples. However, we conclude this section with a cautionary example. Neither of the supposedly more stable methods with Arnoldi and Newton bases work as well as the power basis for the following matrix.

Example 11. We use the Bidiagonal Matrix with superdiagonal of 0.2 except the last diagonal value is changed from 4991 to 10,000. This gives one very dominant eigenvalue, over twice as large as the next one. With both Arnoldi and Newton bases, there is instability for higher degree polynomials. With $deg = 20$, they both reach minimum relative residual norm of about 10^{-4}. Then they do not converge at all with a degree 30 polynomial. Meanwhile, the power basis is much better. It reaches relative residual norm of $7.0 * 10^{-9}$ for degree 20 and $7.5 * 10^{-8}$ for $deg = 30$. We look at a possible reason for the instability of Arnoldi and Newton. Both form $A * w_j - \delta w_j$, for some δ, although Arnoldi also subtracts away other terms. Ideally, the δ should be chosen to make the resulting vector orthogonal to w_j, but in implementing $p(A)z$, a value determined in an earlier phase is used. If there is a dominant eigenvalue and w_j has a very large component in that direction, then the multiplication by A will enlarge this component. With orthogonalization, this component would be significantly reduced. However, the predetermined δ likely will not significantly reduce this component. The resulting vector $A * w_j - \delta w_j$ will then be dominated by the $A * w_j$ portion and again have large component in the direction of the large eigenvalue. Thus it is nearly parallel to the previous vector w_j and this can cause numerical problems. On the other hand, if w_j has a very small component in that direction, then $A * w_j - \delta w_j$ may have the $-\delta w_j$ portion dominate. So the next vector is nearly a multiple of w_j. We have seen both of these happen for this example.

4. Polynomial Preconditioned GMRES-DR. The GMRES-DR method deflates eigenvalues by computing eigenvectors while it is solving linear equations. At the conclusion of a cycle, it uses its subspace to both update the linear equations solution and compute new approximate eigenvectors. Then for the next cycle it builds a subspace that is spanned by the approximate eigenvectors and new Krylov vectors. This subspace is

$$Span\{\tilde{y}_1, \tilde{y}_2, \ldots \tilde{y}_k, r_0, Ar_0, A^2r_0, A^3r_0, \ldots, A^{m-k-1}r_0\}, \tag{4.1}$$

where the $\tilde{y}_i$'s are harmonic Ritz vectors generated by the previous cycle and r_0 is the linear equations residual vector from the previous cycle. GMRES-DR(m,k) has maximum size of subspace m and saves k approximate eigenvectors at the restart. The presence of approximate eigenvectors in the subspace essentially removes or deflates the corresponding eigenvalues. This can vastly improve convergence compared to regular GMRES when there are small eigenvalues. However, if there are many small eigenvalues, polynomial preconditioning may assist GMRES-DR. In the following examples, only the power basis is used to generate the polynomial p.

Example 12. The matrix is the Bidiagonal Matrix with superdiagonal elements of 0.2. We apply polynomial preconditioning to GMRES-DR. Tests are done for $deg = 1$ (no polynomial preconditioning) and $deg = 3$ and 10. Table 4.1 first shows the results from GMRES(20) for comparison and then gives GMRES-DR(20,5) and (30,15). With $k = 5$, only some of the small eigenvalues are deflated, so polynomial preconditioning is fairly effective. The number of matrix-vector products is reduced from 4700 to 2032 with $deg = 10$ and the vectors operations go down from 192,000 to 8130. Meanwhile, with $k = 15$, the most significant small eigenvalues are deflated.

TABLE 4.1
Expenses for PP-GMRES-DR(m,k)

	deg	1	3	10
m=20, k=0	mvp's	18,140	9521	3637
	vector ops	493,000	84,900	9870
m=20, k=5	mvp's	4700	2872	2032
	vector ops	192,000	38,100	8090
m=30, k=15	mvp's	1320	1154	1226
	vector ops	115,000	32,100	9610

TABLE 4.2
Expenses for PP-GMRES-DR(m,k) - indefinite matrix with 3 negative eigenvalues

	deg	1	3	10
m=50, k=0	mvp's	404,000	49,533	3025
	vector ops	23,100,000	936,000	17,200
m=50, k=5	mvp's	25,115	1244	1422
	vector ops	1,720,000	27,700	9340
m=50, k=15	mvp's	680	792	871
	vector ops	64,200	23,800	7070

Polynomial preconditioning is not very effective at reducing the number of matrix-vector products. It does still significantly reduce the vector operations.

Example 13. Next we try the Helmholtz Matrix with $c = 10$ and $n = 500$. Regular GMRES-DR(50,5) struggles because it only computes two of the three negative eigenvalues. Polynomial preconditioning fixes that; see Table 4.2. When more eigenvalues are computed, GMRES-DR does not have trouble. In fact, GMRES-DR(50,15) uses less matrix-vector products than the polynomial preconditioned versions. However as usual, the vector operations are significantly reduced by the preconditioning.

Example 14. We use a new test matrix which will be called the Eigenvalue Circle Matrix. It is chosen to have eigenvalues on the circle in the complex plane with center at (1,0) and radius 1. This matrix is block diagonal with 2 by 2 blocks that have elements $[1 + cos \;\; sin; -sin \;\; 1 + cos]$, where cos and sin are the cosine and sine of the angles 0, $2 * \pi/n$, $4 * \pi/n$, $\ldots (n-2) * \pi/n$. The size is $n = 2000$. We again apply polynomial preconditioned GMRES-DR. Figure 4.1 shows a plot of the absolute value of the degree 10 polynomial s. We can see that this polynomial is near 1.0 over the spectrum except near the origin. Table 4.3 shows the results from GMRES(50) and then gives GMRES-DR(50,10), (50,20), (50,30) and (100,30). The improvement from adding polynomial preconditioning is best for the case of no deflation (5.2 times less matrix-vector products with degree 10 polynomial). However, polynomial preconditioning still can give significant improvement with GMRES-DR. The improvement factor for matrix-vector products from no preconditioning to degree 10 polynomial is 2.6 for GMRES-DR(50,10) and 1.7 for both GMRES-DR(50,20) and (50,30). Matrix-vector products increase with polynomial preconditioning for GMRES-DR(100,30). This example is interesting because while removing just a few small eigenvalues helps, the deflated spectrum retains the difficulty of having eigenvalues partially surround the origin (both above and below and to the right). So polynomial preconditioning is helpful in bringing higher degree polynomials to help with this difficult spectrum.

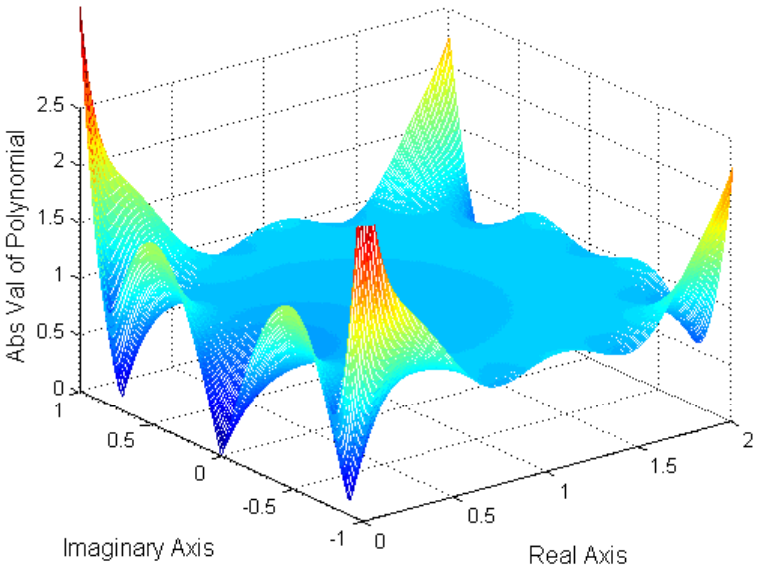

FIG. 4.1. *Plot of the absolute value of the polynomial s of degree 10.*

TABLE 4.3
Expenses for PP-GMRES-DR(m,k) - matrix with eigenvalues in circle.

	deg	1	3	10
m=50, k=0	mvp's	224,050	113,255	43,105
	vector ops	12,800,000	2,140,000	246,000
m=50, k=10	mvp's	12,610	7658	4931
	vector ops	1,020,000	205,000	39,200
m=50, k=20	mvp's	6950	4888	4132
	vector ops	794,000	182,000	45,200
m=50, k=30	mvp's	6250	4365	3736
	vector ops	1,070,000	241,000	59,300
m=100, k=30	mvp's	3110	3240	3824
	vector ops	563,000	193,000	65,000

GMRES-DR(100,30) has high enough polynomials on its own that it does not need the preconditioning. We next look at the vector operations. Polynomial preconditioning substantially reduces them even for the case where matrix-vector products go up. In tests with m and k multiples of 10, the minimum number of vector operations for GMRES-DR occurs for $(m, k) = (100, 30)$. Meanwhile with polynomial preconditioner of degree 10, the minimum is at (50,10). The polynomial preconditioning allows for smaller values of m and k to do well.

Example 15. We experiment with a lattice quantum chromodynamics (QCD) [14, 1, 28, 2] matrix of size $n = 2,654,208$ from a 24^3 x 32 lattice. It is developed using the quenched Wilson gauge at $\beta = 6.0$. The κ value is 0.1571, which is close to $\kappa_{critical}$. This matrix has a spectrum that superficially resembles that of the matrix from Example 4, with eigenvalues approximately inside a circle in the complex plane centered at (25,0) with radius 25. The eigenvalues are not all along the boundary of the circle,

but the difficulty is from small eigenvalues above and below the origin. We apply GMRES-DR(80,60) both without polynomial preconditioning and with s of degree 6. Tests are run on the Baylor Cluster using 432 processors. Regular GMRES-DR converges in less matrix-vector products, 1820 versus 2880 with polynomial preconditioning. Deflating out 60 eigenvalues makes the problem easy enough that polynomial preconditioning is not needed to assist with small eigenvalues. However, the computational time is significantly reduced. Regular GMRES-DR uses 119 CPU seconds, while the polynomial preconditioned version takes 49.2 seconds. Based on the results for the matrix with eigenvalues in a circle, it seems possible for the reduction to be even more for larger, more difficult QCD problems.

5. Multiple Right-hand Sides. We wish to apply polynomial preconditioning to the solution of systems of linear equations with multiple right-hand sides for a fixed matrix. We use a technique called GMRES-Proj from [28] that efficiently deflates eigenvalues to improve the convergence. The first right-hand side is solved with GMRES-DR and the approximate eigenvectors that are computed can then be used for the other right-hand sides. For the second and subsequent right-hand sides, we alternate between projection over the eigenvectors and cycles of regular GMRES. GMRES(m)-Proj(k) projects over k eigenvectors and has cycles of GMRES(m). See [28] for the details. We apply the same polynomial preconditioner to both the first system and the other right-hand sides.

Example 16. This example has the Eigenvalue Circle Matrix. We use the method GMRES-DR(50,10)/GMRES(40)-Proj(10). So the first right-hand side is solved with GMRES-DR and 10 eigenvectors are generated. The second and subsequent right-hand sides use alternating projection over these eigenvectors and cycles of GMRES(40). We solve a total of five systems with right-hand sides generated randomly. Figure 5.1 shows the convergence of all five both without polynomial preconditioning and with a polynomial of degree 10. Both methods have faster convergence for the second and subsequent right-hand sides, because the eigenvectors corresponding to smallest eigenvalues have already been computed. Polynomial preconditioning improves convergence for all right-hand sides. This is because there are more than 10 significant small eigenvalues, and the polynomial preconditioning assists the method in dealing with the remaining ones.

6. Conclusion. We apply polynomial preconditioning with a polynomial related to the minimum residual (GMRES) polynomial. This approach is very effective for difficult problems and can be used along with standard preconditioning. It can be implemented in a very simple way. For this, low degree polynomials should be used for stability. We also give more stable methods for implementing higher degree polynomials.

It should be noted that it is possible to have polynomial preconditioning make the problem worse. This is seen with the highly indefinite matrix in Example 5 where some polynomials help and some hurt. But it is interesting that later we do get consistently good results with higher degree polynomials and the more stable Newton polynomial implementation.

Polynomial preconditioning is also applied to the eigenvalue deflation method GMRES-DR. The polynomial is particularly helpful for difficult problems with many small eigenvalues and for sparse matrices. The benefit extends to problems with multiple right-hand sides.

Based on the results in this paper, polynomial preconditioning should generally be used when restarted GMRES is applied to difficult problems. With the current push

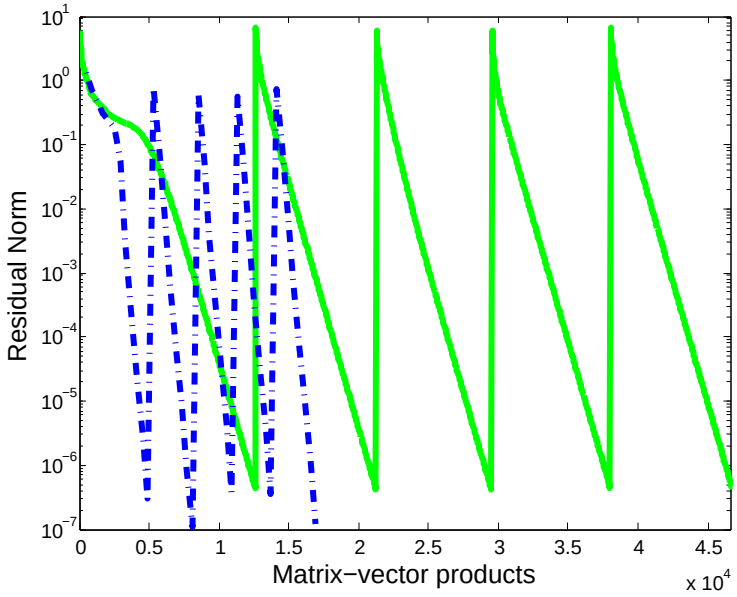

FIG. 5.1. *Five right-hand sides for the matrix with eigenvalues in circle in complex plane. Solid line is GMRES-DR(50,10)/GMRES(40)-Proj(10) without polynomial preconditioning. Dot-dashed line has polynomial preconditioning with s of degree 10.*

towards creating algorithms that are effective on modern architectures, polynomial preconditioning is one possible tool.

A subject for further investigation is using right polynomial preconditioning instead left. In preliminary experiments, right preconditioning is a little more stable for the power basis.

Another future research topic could involve testing polynomial preconditioning on non-restarted methods such as BiCGStab [43] and IDR [41]. Polynomial preconditioning will not be as useful for such methods, because they develop large subspaces. However, polynomial preconditioning could be used to reduce the ratio of vector operations to matrix-vector products. Also, there are occasional problems so difficult that they cause stability problems for non-restarted methods, and polynomial preconditioning may make such problems easier and thus improve stability. Another possible topic is polynomial preconditioning for the Arnoldi method for eigenvalues [42].

Acknowledgments. The second author appreciates the discussions with Mark Embree. Thanks to the referees for many corrections and helpful suggestions. Included in these was the suggestion to try right preconditioning, which was mentioned for future research in the conclusion.

REFERENCES

[1] A. M. Abdel-Rehim, R. B. Morgan, and W. Wilcox. Deflated BiCGStab for linear equations in QCD problems. *Proceedings of Science, LAT2007*, pages 026/1–026/7, 2007.

[2] A. M. Abdel-Rehim, R. B. Morgan, and W. Wilcox. Improved seed methods for symmetric positive definite linear equations with multiple right-hand sides. *Numer. Linear Algebra Appl.*, 21:453–471, 2014.

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

[4] S. F. Ashby. Polynomial preconditioning for conjugate gradient methods. PhD Thesis, University of Illinois at Urbana-Champaign, 1987.
[5] S. F. Ashby, T. A. Manteuffel, and J. S. Otto. A comparison of adaptive Chebyshev and least squares polynomial preconditioning for conjugate gradient methods. *SIAM J. Sci. Statist. Comput.*, 13:1–29, 1992.
[6] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel. Adaptively preconditioned GMRES algorithms. *SIAM J. Sci. Comput.*, 20:243–269, 1998.
[7] Z. Bai, D. Hu, and L. Reichel. A Newton basis GMRES implementation. *IMA J. Numer. Anal.*, 14:563–581, 1994.
[8] A. Bayliss, C. I. Goldstein, and E. Turkel. An iterative method for Helmholtz equation. *J. Comput. Phys.*, 49:443457, 1983.
[9] P.F. Dubois, A. Greenbaum, and G.H. Rodrigue. Approximating the inverse of a matrix for use in iterative algorithms on vector processors. *Computing*, 22:257–268, 1979.
[10] J. Erhel, K. Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *J. Comput. Appl. Math.*, 69:303–318, 1996.
[11] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM J. Sci. Comput.*, 27:1471–1492, 2006.
[12] B. Fischer and L. Reichel. A stable Richardson iteration method for complex linear systems. *Numer. Math.*, 54:225–241, 1988.
[13] R. W. Freund. On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices. *Numer. Math.*, 57:285–312, 1990.
[14] A. Frommer. Linear systems solvers - recent developments and implications for lattice computations. *Nucl. Phys. B (Proc. Suppl.)*, 53:120–126, 1997.
[15] M. H. Gutknecht. Spectral deflation in Krylov solvers: A theory of coordinate space based methods. *Electron. Trans. Numer. Anal.*, 39:156–185, 2012.
[16] W. Joubert. A robust GMRES-based adaptive polynomial preconditioning algorithm for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 15:427–439, 1994.
[17] W. D. Joubert and G. F. Carey. Parallelizable restarted iterative methods for nonsymmetric linear systems. II: parallel implementation. *Inter. J. Comput. Math.*, 44:269–290, 1992.
[18] W. D. Joubert and G. F. Carey. Parallelizable restarted iterative methods for nonsymmetric linear systems. part I: Theory. *Inter. J. Comput. Math.*, 44:243–267, 1992.
[19] I. M. Khabaza. An iterative least-square method suitable for solving large sparse matrices. *Comput. J.*, 6:202–206, 1963.
[20] S. A. Kharchenko and A. Y. Yeremin. Eigenvalue translation based preconditioners for the GMRES(k) method. *Num. Lin. Alg. with Appl.*, 2:51–77, 1995.
[21] A. L. Laird and M. B. Giles. Preconditioned iterative solution of the 2D Helmholtz equation. Technical Report 02/12, Oxford Computer Laboratory, Oxford, UK, 2002.
[22] C. Lanczos. Chebyshev polynomials in the solution large-scale linear systems. *Proceedings of the ACM*, pages 124–133, 1952.
[23] Y. Liang, M. Szularz, and L. T. Yang. Stability of polynomial preconditioning. *Proceedings of ALGORITMY 2000, 15th Conference on Scientific Computing*, pages 264–272, 2000.
[24] Y. Liang, J. Weston, and M. Szularz. Finite-element-wise domain decomposition iterative solvers with polynomial preconditioning. *Math. Comput. Model.*, pages 421–437, 2013.
[25] R. B. Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra Appl.*, 154-156:289–309, 1991.
[26] R. B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.*, 16:1154–1171, 1995.
[27] R. B. Morgan. GMRES with deflated restarting. *SIAM J. Sci. Comput.*, 24:20–37, 2002.
[28] R. B. Morgan and W. Wilcox. Deflated iterative methods for linear equations with multiple right-hand sides. arXiv:math-ph/0405053v2, 2004.
[29] N. M. Nachtigal, L. Reichel, and L. N. Trefethen. A hybrid GMRES algorithm for nonsymmetric linear systems. *SIAM J. Matrix Anal. Appl.*, 13:796–825, 1992.
[30] D. P. O'Leary. Yet another polynomial preconditioner for the conjugate gradient algorithm. *Linear Algebra Appl.*, 154-156:377–388, 1991.
[31] C. C. Paige, B. N. Parlett, and H. A. van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Num. Lin. Alg. with Appl.*, 2:115–133, 1995.
[32] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28:1651–1674, 2006.
[33] B. Philippe and L. Reichel. On the generation of Krylov subspace bases. *Appl. Numer. Math.*, 62:1171–1186, 2012.
[34] H. Rutishauser. Theory of gradient methods. In M. Engeli, Th. Ginsburg, H. Rutishauser, and E. Stiefel, editors, *Refined Iterative Methods for Computation of the Solution and the*

Eigenvalues of Self-Adjoint Boundary Value Problems, pages 24–49. Birkhauser, Basel, 1959.

[35] Y. Saad. Practical use of polynomial preconditionings for the conjugate gradient method. *SIAM J. Sci. Statist. Comput.*, 6:865881, 1985.

[36] Y. Saad. Least squares polynomials in the complex plane and their use for solving sparse nonsymmetric linear systems. *SIAM J. Numer. Anal.*, 24:155–169, 1987.

[37] Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd Edition.* SIAM, Philadelphia, PA, 2003.

[38] Y. Saad. *Numerical Methods for Large Eigenvalue Problems, 2nd Edition.* SIAM, Philadelphia, PA, 2011.

[39] Y. Saad and M. H. Schultz. GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[40] D. C. Smolarski and P. E. Saylor. An optimal iterative method for solving any linear system with a square matrix. *BIT*, 28:163–178, 1988.

[41] P. Sonneveld and M. B. van Gijzen. IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 31:1035–1062, 2008.

[42] H. K. Thornquist. Fixed-polynomial approximate spectral transformations for preconditioning the eigenvalue problem. PhD Thesis, Rice University, TR06-05, 2006.

[43] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 12:631–644, 1992.