1

POLYNOMIAL PRECONDITIONED BICGSTAB AND IDR

JENNIFER A. LOE[†] AND RONALD B. MORGAN[‡]

Abstract. Polynomial preconditioning is applied to the nonsymmetric Lanczos methods BiCGStab and IDR for solving large nonsymmetric systems of linear equations. A polynomial related to the minimum residual polynomial is used for the preconditioner. It is implemented in a simple way that makes polynomial preconditioning of nonsymmetric matrices more practical. This preconditioning can reduce costs, especially for difficult problems. It also can improve stability for highly non-normal or indefinite matrices. Examples are given to demonstrate this.

Key words. linear equations, polynomial preconditioning, nonsymmetric Lanczos, BiCGStab, IDR

AMS subject classifications. 65F15, 15A18

1. Introduction.

The nonsymmetric Lanczos algorithm [8, 10] can be used to derive iterative methods for solving large nonsymmetric systems of linear equations. Two such methods are BiCGStab [26] and IDR [22]. We look at applying polynomial preconditioning to these methods. The preconditioning is done in a simple way involving the minimum residual polynomial. It will be shown that polynomial preconditioning can often reduce costs and improve stability.

Polynomial preconditioning involves multiplying a matrix by a polynomial of the matrix and thus changing or preconditioning the spectrum. With right preconditioning, the system of linear equations $Ax = b$ is transformed to $Ap(A)y = b$, with $x = p(A)y$, where p is a polynomial of degree $deg - 1$. Define the polynomial $s(\alpha) = \alpha * p(\alpha)$. Then the polynomial preconditioned problem is $s(A)y = b$, with s of degree deg.

Polynomial preconditioning has been investigated extensively for solving linear equations. A few references are [3, 9, 23, 16, 17, 20, 14, 2, 4, 7] and see [18] for a summary. In spite of much work being done, polynomial preconditioning does not seem to have caught on in practice. It is shown in [11] that for difficult problems, polynomial preconditioning can greatly improve the restarted GMRES method [19]. This is partly because with more matrix-vector products per iteration, there are less iterations and thus less orthogonalization expense. Also, the polynomial preconditioning allows the method to use higher degree polynomials, and thus for difficult problems, the convergence in terms of matrix-vector products can be substantially reduced. Polynomial preconditioning can be combined with regular preconditioning but may not be needed if the regular preconditioning is extremely effective.

Krylov iterative methods for solving large systems of linear equations may either use full orthogonalization, as in GMRES, or be based on the nonsymmetric Lanczos recurrence, as with CGS [21], TFQMR [6], BiCGStab and IDR. The Lanczos methods do not need to be restarted, so they can develop larger Krylov subspaces. For difficult problems, this sometimes leads to much faster convergence than restarted GMRES.

*The second author was supported by NSF grant DMS-1418677.

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

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

We will apply polynomial preconditioning to non-restarted methods. It may seem that there is no reason for this. Since they do not restart, they naturally use high degree polynomials. However, the goal of this paper is to show that this polynomial preconditioning is worth considering. We give four reasons for this:

Reason 1: Sometimes it is more efficient to apply together the *deg* matrix-vector products used by the polynomial preconditioned operator. This can be either because of the form of the matrix or to take advantage of parallel processing. This is problem and computer dependent and will not be considered further.

Reason 2: Polynomial preconditioning decreases the number of vector operations per matrix-vector product. Therefore, the expense can be significantly reduced for fairly sparse matrices. Also, reducing dot products helps cut communication costs.

Reason 3: We can view polynomial preconditioning as creating an easier problem with a better distribution of eigenvalues. The spectrum of $s(A)$ is hopefully better than that of A, with the tradeoff that more matrix-vector products are needed per iteration. Non-restarted Krylov solvers are not optimal; they are not guaranteed to find the best approximate solution from the subspace. Roundoff error can add to this problem. The approximation is particularly likely to not be nearly optimal for difficult problems. The easier polynomial preconditioned problem can thus potentially give faster convergence in terms of matrix-vector products.

Reason 4: Non-restarted linear equations solvers can have stability problems for certain cases. The easier spectrum created by the polynomial preconditioning (as discussed in Reason 3) may help with stability.

This paper is structured as follows: Section 2 gives the algorithm for finding the polynomial. Section 3 considers Reasons 2 and 3 (above) and shows that polynomial preconditioning can reduce costs. Improved stability, as mentioned in Reason 4, is in Section 4.

2. The Polynomial. Here we discuss our implementation of the polynomial preconditioner. We use polynomials related to the minimum residual or GMRES polynomial. This polynomial has harmonic Ritz values [12, 15] as roots [5, 13]. As detailed in the next paragraph, if this minimum residual polynomial is q, then $s = 1-q$, so s can be implemented with harmonic Ritz values. However, then it is difficult to accurately implement p [11]. Instead, we develop p and use it for s.

For a Krylov subspace $K = Span\{b, Ab, A^2b, \ldots, A^{deg-1}b\}$ of dimension deg, the approximate solution from K is $\hat{x} = p(A)b$, where p is a polynomial of degree $deg-1$. We let s be the degree deg polynomial defined by $s(\alpha) = \alpha p(\alpha)$. The residual vector is $r = b - A\hat{x} = q(A)b$, where $q(\alpha) = 1 - \alpha p(\alpha) = 1 - s(\alpha)$. We choose the p and s polynomials to correspond to the q that minimizes the norm of the residual vector. We will compute them in the simple way from [1, 11] using a power basis. Let

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

Solve

$$(AV)^T(AV)d = (AV)^Tb$$

for d. Then $\hat{x} = Vd$, and d has the coefficients of p. Coefficients of s are the same but shifted since $s(\alpha) = \alpha p(\alpha)$. This process can clearly be unstable because the columns of V become nearly linearly dependent for large deg. However, in [11] it is shown that this can be useful for low degree polynomials. If higher degree polynomials are needed, more stable methods are given in [11], but they are not considered here.

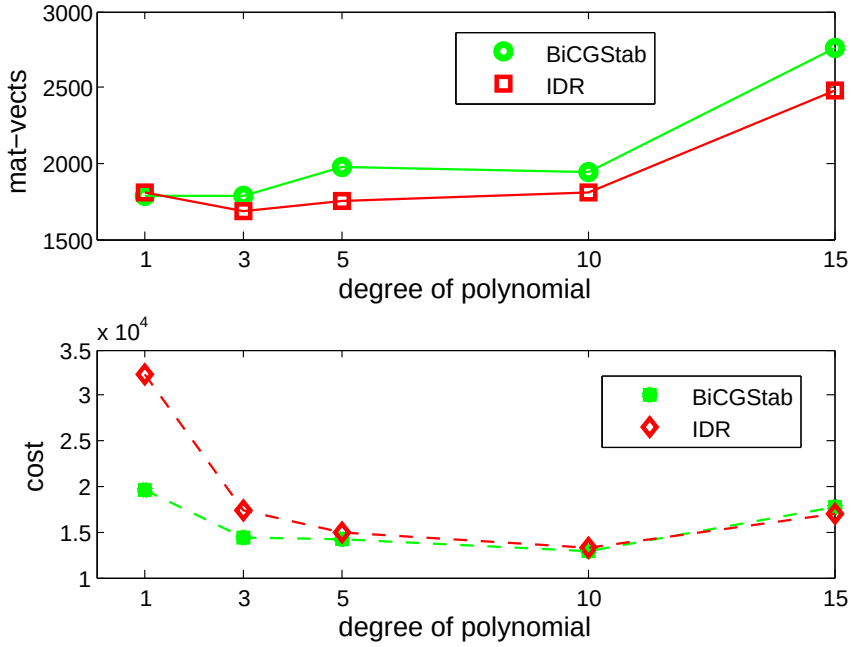

FIG. 3.1. *Convection-diffusion equation with $n = 160,000$. Polynomials of different degree are compared to no polynomial preconditioning for BiCGStab and IDR(4). Both matrix-vector products and an estimate of the overall cost are given.*

3. Cost. In this section, we give tests showing that polynomial preconditioning can reduce costs by cutting the number of length-n vector operations. Occasionally for difficult problems, the number of matrix-vector products can also be reduced.

In all examples, we apply right preconditioning to both BiCGStab and IDR(4) and compare to the non-preconditioned versions. The degree given for the polynomial is the degree of $s(\alpha)$. Degree 1 means no polynomial preconditioning. We programmed BiCGStab in MATLAB. For IDR, we use the program described in the paper [27] and available in MATLAB from the authors. Only the default method IDR(4) is considered. We use residual norm tolerance 10^{-10}. Right-hand sides are generated with random Normal(0,1) elements, then normed to 1.

Example 1. Let the matrix be from finite difference discretization of the partial differential operator $u_{xx} + u_{yy} + 20u_x$ on the unit square with Dirichlet boundary conditions. With $h = \frac{1}{401}$, the matrix is size $n = 160,000$. Polynomials $s(\alpha)$ of degree 3, 5, 10 and 15 are used. The top half of Figure 3.1 has the matrix-vector products. For BiCGStab, the number of matrix-vector products is higher with polynomial preconditioning except for being slightly lower with $deg = 3$. With IDR, polynomial preconditioning reduces matrix-vector products a little except for $deg = 15$. IDR needs less matrix-vector products than BiCGStab for all the tests with polynomial preconditioning.

The bottom half of the figure has an estimate of the overall cost in terms of vector operations, calculated as $cost = 5 * mvp + vops$. Here $vops$ is the number of length n vector operations such as dot products and daxpys, and mvp is the number of matrix-vector products. The actual execution time depends on many factors including the computer architecture, but hopefully this cost estimate gives some useful information.

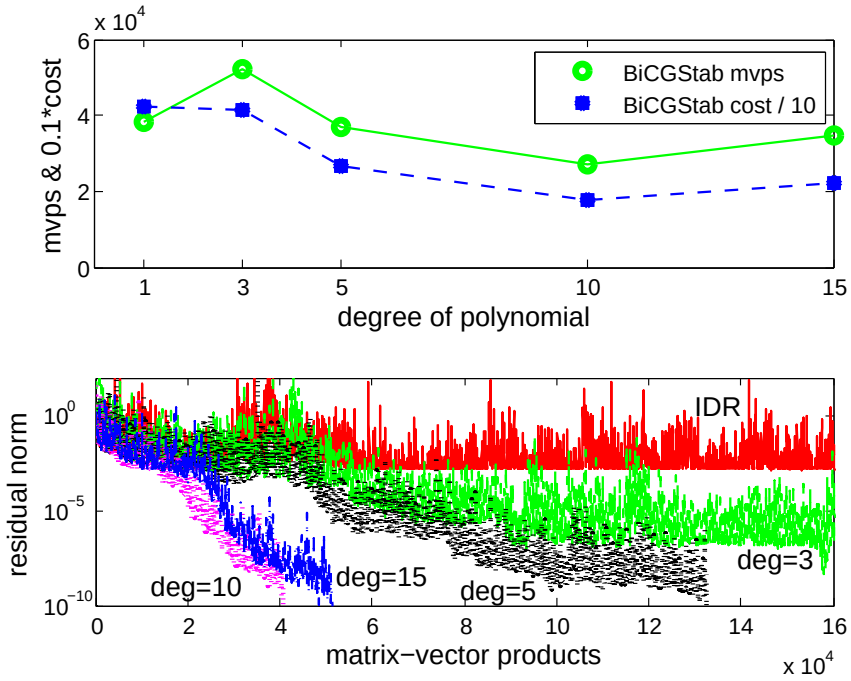

FIG. 3.2. *Convection-diffusion with top half of operator multiplied by 250. Again* $n = 160,000$. *The top half of the figure has matrix-vector products and the estimate of cost (scaled by one-tenth) for different degree polynomials and BiCGStab. The bottom half has the residual norm convergence of IDR(4) with different degree polynomials.*

BiCGStab uses 2 *mvp*s and 12 *vops* for every iteration. Polynomial preconditioned BiCGStab uses $2*deg$ matrix-vector products and $12+2*deg$ vector ops per iteration. The extra $2*deg$ vector ops are part of implementing $s(A)$ times a vector twice. The figure shows that polynomial preconditioning gives a significant improvement. For instance, for polynomial preconditioned BiCGStab with $deg = 10$, the cost is 12,842 versus 19,690 for regular BiCGStab.

IDR(4) uses more length n vector operations per iteration than BiCGStab with 25.6 per iteration to go along with the two matrix-vector products. Therefore polynomial preconditioning can potentially help IDR more by reducing the iterations and thus the vector ops. For example, the cost estimate is reduced from 32,218 down to 13,201 with $deg = 10$.

Example 2. We now modify the matrix from Example 1 to make it more ill-conditioned. The top half of the unit square has the differential operator multiplied by 250, so it is $250u_{xx} + 250u_{yy} + 5000u_x$. In the bottom half, it stays the same as in the previous example. The ratio of largest to smallest eigenvalue is $2 * 10^6$. The linear equations problem is now difficult enough that polynomial preconditioning can reduce matrix-vector products as mentioned by Reason 3 in the Introduction. The top half of Figure 3.2 has the number of matrix-vector products and the estimate of cost for BiCGStab. Polynomial preconditioning with $deg = 10$ is best. It reduces the *mvp*'s by 30% and the cost by 58%.

IDR is much less effective than BiCGStab for this matrix. Regular IDR(4) does not converge. Polynomial preconditioning helps; see the residual norm curves in the bottom of Figure 3.2. With $deg = 5$, the method converges slowly. With $deg = 10$ and 15, convergence is much better, although still a little slower than BiCGStab.

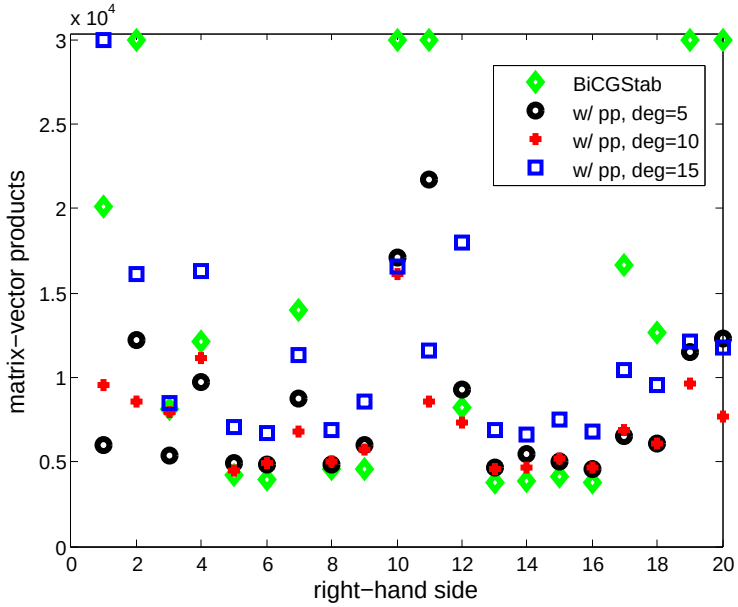

FIG. 4.1. *Regular BiCGStab compared to polynomial preconditioned BiCGStab for 20 right-hand sides. Polynomials are deg* $= 5, 10, 15$. *Matrix is bidiagonal with superdiagonal of 1.*

This example with polynomial preconditioning helping IDR to converge leads into the next section where we look at problems for which the solvers may be unstable.

4. Stability. Methods based on the nonsymmetric Lanczos algorithm always have potential for instability. This is particularly true for highly non-normal problems. Even though BiCGStab and IDR are designed to work well for such problems, they still can have trouble. In the next example, we see that instability increases as the matrix is made more non-normal and that polynomial preconditioning can help. Then in the final example, the indefiniteness of a matrix is increased.

Example 3. We let the matrix be bidiagonal of size $n = 10,000$. The diagonal elements are $0.1, 0.2, 0.3, \ldots 9.8, 9.9, 10, 11, 12, \ldots, 9909, 9910$, and the superdiagonal elements are all the same, say β. The value of β will vary. As β increases, the matrix becomes more non-normal and more difficult to solve. We first let $\beta = 1$ and solve systems of equations with 20 random right-hand sides. Regular BiCGStab is applied and then polynomial preconditioned BiCGStab with polynomials of degree 5, 10 and 15. The maximum number of matrix-vector products is set at 30,000. Figure 4.1 gives the number of matrix-vector products needed for convergence. The symbols at the top of the graph at 30,000 indicate that the method did not converge. For eight of the right-hand sides, regular BiCGStab is the best method in terms of matrix-vector products. For others it converges slowly, and for five it does not converge. With polynomial preconditioning there is again variability, but the results are improved. All systems converge except for one with $deg = 15$. The $\beta = 1$ column of Table 4.1 gives the number of systems that converge with each degree polynomial ($deg = 1$ indicates regular BiCGStab). The table also includes results for $\beta = 1.5$ and $\beta = 2$, displaying the number of systems that converge within 40,000 matrix-vector products for $\beta = 1.5$ and within 50,000 for $\beta = 2$. Polynomial preconditioning usually improves the method. For instance, with $\beta = 2$ and $deg = 10$, 15 of the 20 systems converge compared to only two for regular BiCGStab. Figure 4.2 has the number of matrix-

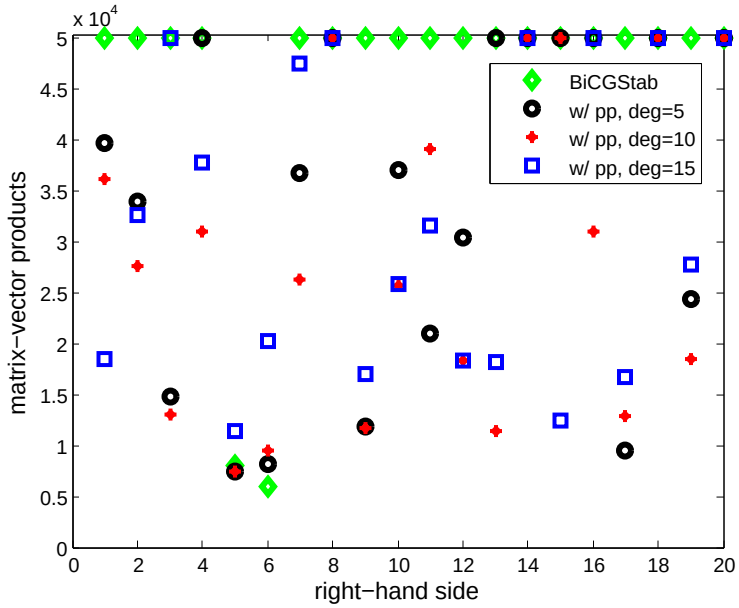

FIG. 4.2. *Regular BiCGStab compared to polynomial preconditioned BiCGStab for 20 right-hand sides. Polynomials are deg* $= 5, 10, 15$. *Matrix is bidiagonal with superdiagonal of 2.*

TABLE 4.1

BiCGStab with and without polynomial preconditioning and with increasing superdiagonal terms. The number of systems that converge is given out of 20 right-hand sides.

deg	$\beta = 1$ 30,000 max	$\beta = 1.5$ 40,000 max	$\beta = 2$ 50,000 max
1	15	5	2
5	20	17	12
10	20	19	15
15	19	19	14

vector products for each $\beta = 2$ system that converges. Again the symbols at the top are for the systems that do not converge.

Next IDR is tested. For this problem it is more effective than BiCGStab. Figure 4.3 shows results of tests with $\beta = 2$. Here we consider IDR to have converged even if it returns FLAG=2. This indicates that it may not have reached the full desired accuracy but has made progress. FLAG=1 indicates little or no convergence. Under this assumption, regular IDR converges in under 30,000 matrix-vector products for all 20 systems, and the average relative residual outputted by the routine is RELRES$=5.7 * 10^{-7}$. Figure 4.3 shows that polynomial preconditioning significantly reduces the number of matrix-vector products, except for two systems that do not converge within 30,000 for $deg = 15$. The solution accuracy is also better with $deg = 5$ and 10. The average RELRES is $6.3 * 10^{-8}$ for $deg = 5$, $1.6 * 10^{-8}$ for $deg = 10$ and $7.2 * 10^{-7}$ for the 18 systems that converge with $deg = 15$.

Finally, Figure 4.4 has results with $\beta = 3$. Regular IDR does not converge for any of the 20 right-hand sides. All of the polynomial preconditioned runs do converge, at least in the sense of getting FLAG=2. Degree 10 usually has the fewest matrix-vector products. RELRES ranges from $1.4 * 10^{-1}$ up to $6.5 * 10^{7}$ for regular IDR. The average

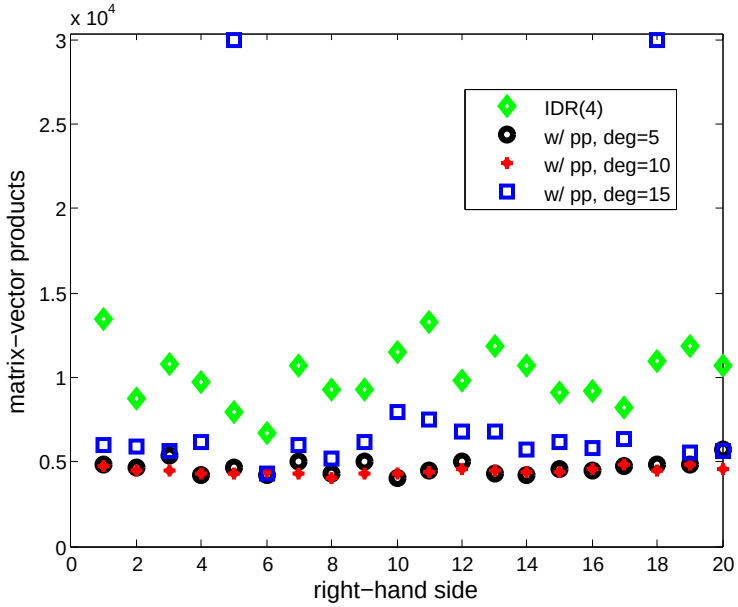

FIG. 4.3. *IDR for 20 right-hand sides compared to IDR with polynomial preconditioning. The polynomials have deg* $= 5, 10, 15$. *Matrix is bidiagonal with superdiagonal of 2.*

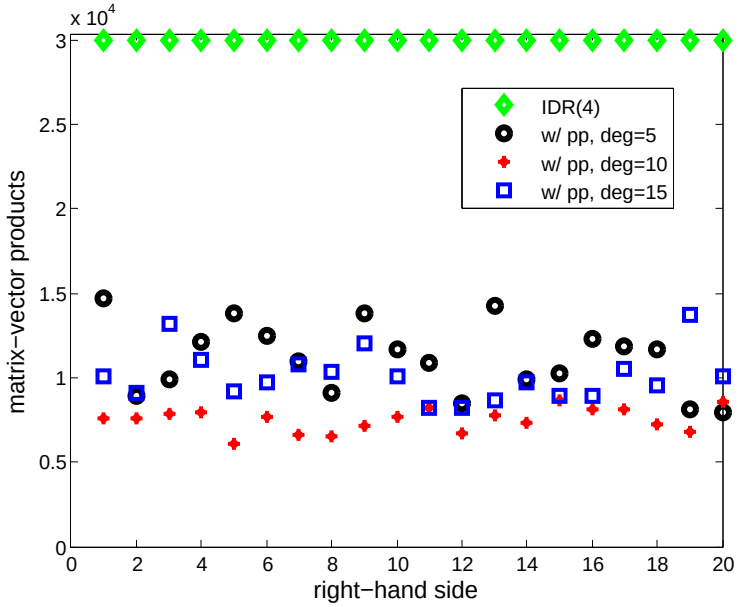

FIG. 4.4. *IDR for 20 right-hand sides compared to IDR with polynomial preconditioning using polynomials of deg* $= 5, 10$ *and* 15. *Matrix is bidiagonal with superdiagonal of 3.*

RELRES is $5.0 * 10^{-4}$ with $deg = 5$, then $8.0 * 10^{-5}$ for $deg = 10$, and $1.7 * 10^{-5}$ for $deg = 15$. The combination of IDR and polynomial preconditioning allows us to find approximate solutions with a difficult non-normal matrix.

Figure 4.5 has plots of the pseudospectrum [24, 25] for a smaller version of this matrix with $n = 150$. It has the same 100 small eigenvalues, but the largest eigenvalue is 60. The β is again 3. The top of the figure has the pseudospectrum for A and the

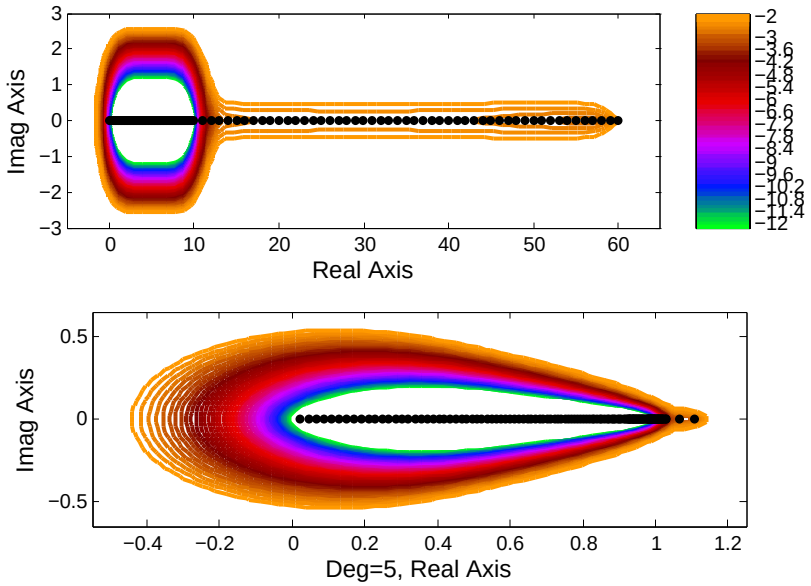

FIG. 4.5. *Pseudospectra for A and for $s(A)$ with deg $= 5$. Matrix is bidiagonal with superdiagonal of 3 and $n = 150$.*

bottom has the pseudospectrum for $s(A)$ with s of degree 5. It is not clear that the non-normality is improved by the polynomial preconditioning. The overall spectrum is improved, and perhaps this helps the iterative method to deal with the non-normality.

Example 4. We generate a matrix B that is 500 by 500 with all of the elements random Normal(0,1). The eigenvalues are roughly inside of a circle centered at the origin with radius between 21 and 22. We shift the matrix by multiples of the identity to make it more reasonable for solving the linear equations. So $A = B - \sigma I$.

We first compare regular BiCGStab to polynomial preconditioned BiCGStab and give results with one random right-hand side. Other right-hand sides seem to give similar results. The first matrix has shift $\sigma = 22$. The leftmost eigenvalues in the complex plane have real part 1.043, so the matrix is somewhat away from being indefinite. A matrix is considered to be indefinite if it has eigenvalues with both positive and negative real parts. In terms of matrix-vector products, regular BiCGStab outperforms polynomial preconditioned, 404 compared to 429 with $deg = 5$ and 459 with $deg = 10$. Next, the shift is changed to 21.5. Polynomial preconditioning now needs less matrix-vector products with 519 for $deg = 10$ compared to 672 for regular BiCGStab; see Figure 4.6 for the residual norm curves. Though not shown in the figure, $deg = 5$ also uses 519 *mvp*'s. Finally with $\sigma = 21$, all eigenvalues still have positive real parts, but four are very near the edge. Also importantly, the spectrum is difficult for solving linear equations due to having eigenvalues both above and below the origin. The top left portion of Figure 4.7 has these eigenvalues plotted. Regular BiCGStab does not converge (it produces NAN's after 348 *mvp*'s). Meanwhile with polynomial preconditioning, BiCGStab converges with 659 matrix-vector products for $deg = 5$ and 699 for $deg = 10$. These runs are also on Figure 4.6. The top right of Figure 4.7 has a plot of the absolute value of the polynomial $s(\alpha)$ of $deg = 5$. The eigenvalues of the resulting matrix $s(A)$ are in the bottom left of the figure. While there are still eigenvalues on the edge of the positive real part of the complex plane, many eigenvalues are moved together near one making this an easier spectrum. The

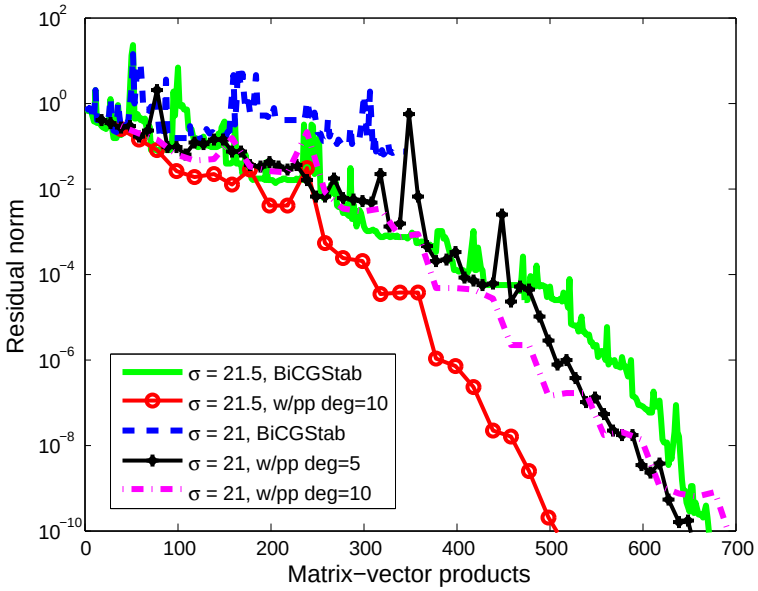

FIG. 4.6. *Shifted random matrix with two shifts. Compare regular BiCGStab to polynomial preconditioned BiCGStab.*

$deg = 10$ polynomial preconditioned version converges for shifts as far down as $\sigma = 18$ (with 5079 matrix-vector products needed). There are 29 eigenvalues with negative real parts for this matrix, so the polynomial preconditioning helps BiCGStab to be much more effective for these indefinite problems.

IDR is much better than BiCGStab at handling this indefiniteness and so we can look at smaller shifts that give more indefinite matrices. The top of Figure 4.8 has matrix-vector product results for IDR without polynomial preconditioning and with polynomials of degree 3 and 5. Results for $deg = 10$ are not quite as good. Regular IDR converges with FLAG=2 for shifts of 19 down to 9, although with increasing number of matrix-vector products and decreasing final residual norm. By $\sigma = 9$, there are 127 eigenvalues with negative real parts, and so the origin is deep into the spectrum. For the larger shifts (easier problems), polynomial preconditioning uses more matrix-vector products, but both degree polynomials give a reduction for $\sigma = 11$ and smaller. However, it is the final residual norms that are substantially improved by polynomial preconditioning; see the bottom of Figure 4.8. They are several orders of magnitude better for the smaller shifts. At $\sigma = 8$, regular IDR does not converge (FLAG=1). The polynomial preconditioned versions do converge to some degree with final residual norms of $2.3 * 10^{-4}$ with $deg = 3$ and $3.9 * 10^{-5}$ for $deg = 5$.

We look at the polynomial preconditioned spectrum with $\sigma = 12$ and $deg = 5$. The bottom right of Figure 4.7 has the eigenvalues of $s(A)$. There are many eigenvalues with negative real parts, specifically 55. This compares to 93 for A (the eigenvalues are the same as in the top left part of the figure except shifted left by 9). While the spectrum is still very indefinite, it is better because of less negative real part eigenvalues and especially due to the clumping together of many of the eigenvalues.

5. Conclusion. Polynomial preconditioning for BiCGStab and IDR can increase the number of matrix-vector products, but may still reduce the cost by cutting the number of vector operations. Communication costs may thus also be reduced. For

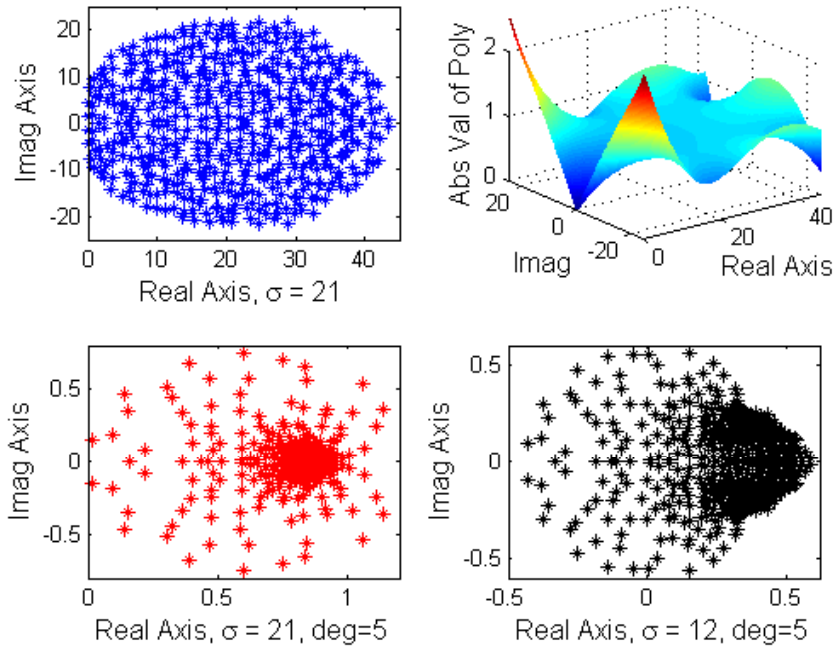

FIG. 4.7. *Top left has the eigenvalues of the random matrix with shift $\sigma = 21$. Top right has the absolute value of the polynomial s of degree 5 for $\sigma = 21$. Bottom left has the eigenvalues of $s(A)$ for deg $= 5$. Bottom right has the eigenvalues of $s(A)$ with $\sigma = 12$ and deg $= 5$.*

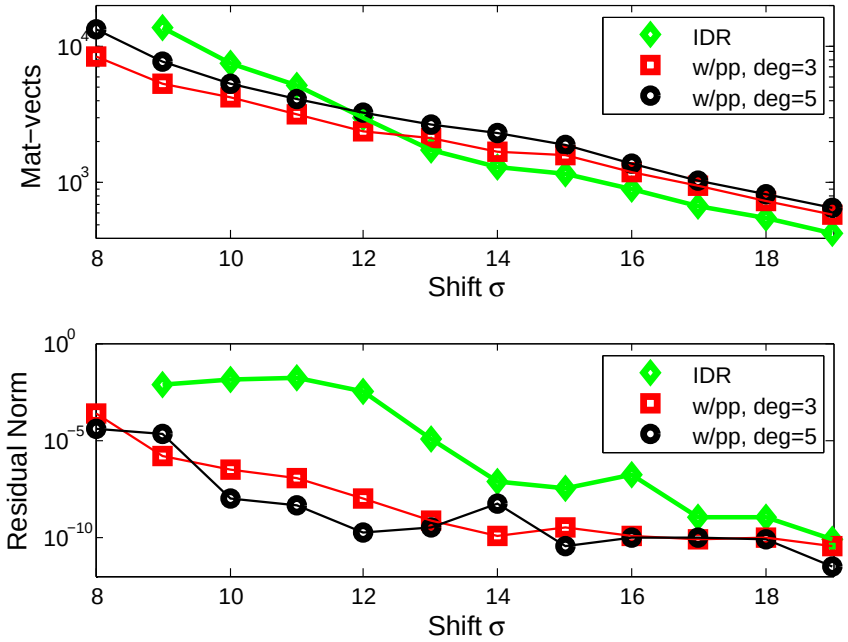

FIG. 4.8. *Different shifts of the random matrix. Compare regular IDR to IDR with polynomial preconditioning with deg $= 3$ and 5. Top half has the number of matrix-vector products until the program terminates. Bottom half has the residual norm at the termination.*

very difficult problems, the number of matrix-vector products may be decreased. Also, examples show that stability is improved for indefinite and highly-normal matrices.

In this work, the polynomial is generated in a simple way with a power basis. It would be interesting to compare with higher degree polynomials found by more stable methods in [11]. Comparisons with other choices for the polynomial preconditioner would also be worthwhile.

A comparison of BiCGStab and IDR was not the purpose of this paper. However, it is interesting that IDR is significantly better for the examples that challenge stability with non-normality or indefiniteness. But in Example 2 with a problem that is ill-conditioned due to small and large eigenvalues, BiCGStab is best.

REFERENCES

[1] 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.

[2] 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.

[3] L. Cesari. Sulla risoluzione dei sistemi di equazioni lineari per approssimazioni successive. *Atti Accad. Naz. Lincei Rend. Cl. Sci. Fis. Mat. Nat., Ser. 6a*, 25:422428, 1937.

[4] R. W. Freund. Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Statist. Comput.*, 13:425–448, 1992.

[5] R. W. Freund. Quasi-kernel polynomials and their use in non-Hermitian matrix iterations. *J. Comput. Appl. Math.*, 43:135–158, 1992.

[6] R. W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14:470–482, 1993.

[7] W. Joubert. A robust GMRES-based adaptive polynomial preconditioning algorithm for non-symmetric linear systems. *SIAM J. Sci. Comput.*, 15:427–439, 1994.

[8] C. Lanczos. An iterative method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards*, 45:255–282, 1950.

[9] C. Lanczos. Chebyshev polynomials in the solution large-scale linear systems. *Proceedings of the ACM*, pages 124–133, 1952.

[10] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards*, 49:33–53, 1952.

[11] Q. Liu, R. B. Morgan, and W. Wilcox. Polynomial preconditioned GMRES and GMRES-DR. *SIAM J. Sci. Comput.*, 37:S407–S428, 2015.

[12] R. B. Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra Appl.*, 154-156:289–309, 1991.

[13] R. B. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM J. Matrix Anal. Appl.*, 21:1112–1135, 2000.

[14] D. P. O'Leary. Yet another polynomial preconditioner for the conjugate gradient algorithm. *Linear Algebra Appl.*, 154-156:377–388, 1991.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

[21] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10:36–52, 1989.

[22] 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.
[23] E. L. Stieffel. Kernel polynomials in linear algebra and their numerical applications. *U. S. Nat. Bur. Standards, Appl. Math. Ser.*, 49:1–22, 1958.
[24] L. N. Trefethen. Approximation theory and numerical linear algebra. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation II*. Chapman and Hall, London, 1990.
[25] L. N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton Univ. Press, Princeton, NJ, 2005.
[26] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.
[27] M. B. van Gijzen and P. Sonneveld. Algorithm 913: An elegant IDR(s) variant that efficiently exploits bi-orthogonality properties. *ACM Trans. Math. Soft.*, 38:5:1–5:19, 2011.