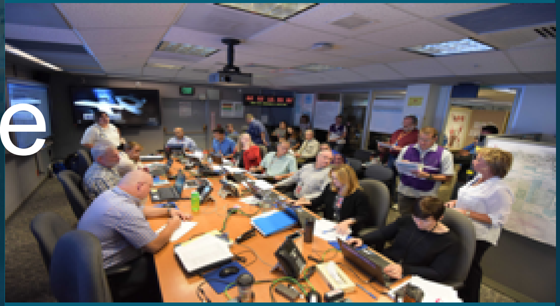
SAND2022-7460C

Polynomial Preconditioning with the GMRES Polynomial



Jennifer Loe, Sandia National Laboratories

With collaborations from:
Ron Morgan (Baylor Univ.), Mark Embree (Virginia Tech),
Erik Boman (Sandia Labs), Heidi Thornquist (Sandia Labs)

SAND2022-2154 PE

Background / Introduction:

- What is the GMRES polynomial?
- How do we implement the polynomial?
- Why does the polynomial work for preconditioning?
- Introductory Examples

Generalized Minimum RESidual Method: GMRES

Algorithm GMRES (Modified Gram-Schmidt) [Saad, Schultz ’86]

$\gamma = \|b\|_2$ and $v_1 = b/\gamma$
for $j = 1 : m$ **do**
    $w_j = Av_j$
    for $i = 1 : j$ **do**
        $h_{ij} = v_i^T w_j$
        $w_j = w_j - h_{ij} v_i$
    end for
    $h_{j+1,j} = \|w_j\|_2.$
    $v_{j+1} = w_j / h_{j+1,j}$
end for
Define the $(m+1) \times m$ matrix $\overline{H} = \{h_{ij}\}$
Solve least-squares problem $\overline{H}d = \gamma e_1$ for d.
$\hat{x} = V_m d$

Orthonormalize basis vectors (lines 4–9)

Build Krylov Subspace (lines 2–10)

Project to find minimum residual solution (lines 11–13)

Add restarting when needed.

Previous Works on Polynomial Preconditioning….

Lanczos 1952; Stieffel 1958; Rutishauser 1959; Saad 1984, 1987; Ashby 1987; Smolarski, Saylor 1988; Fischer, Reichel 1988; O'Leary 1991; Ashby, Manteuffel, Otto 1992; van Gijzen 1995;

Liang, Weston, Szularz 2002; Liang 2005; Thornquist 2006;

Liang, Szularz, Yang 2013; Zhang, Zhang 2013; Liu, Morgan, Wilcox 2015; Li, Xi, Vecharynski, Yang, Saad 2016; Zhang, Huang, Sun 2017; Bergamaschi, Calomardo 2020; Loe, Thornquist, Boman 2020; Ye, Xi, Saad 2021; Embree, Loe, Morgan 2021; Loe, Morgan 2021;

And many, many more!!

The GMRES Polynomial

$$\hat{x} \in \mathcal{K}(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\}$$

So $\hat{x} = p(A)b$.

Then $r = b - A\hat{x} = b - Ap(A)b = (I - Ap(A))b$.

So as $\|b - A\hat{x}\|_2 \to 0$, we get that $\|I - Ap(A)\|_2 \to 0$.

$\hat{x}$ becomes a better solution as $deg(p(A)$ increases

So $Ap(A) \to I$ and $p(A) \to A^{-1}$ as $deg(p(A))$ increases.

Note: Others have used the GMRES Polynomial, just not for preconditioning linear systems! (See Nachtigal, Reichel, Trefethen, "Hybrid GMRES…" 1992 and Thornquist Thesis 2006, Rice Univ.)

Implementing the Polynomial Preconditioner

$$Ap(A)y = b,$$
$$x = p(A)y.$$

$$Ap(A) = I - \prod_{i=1}^{d}\left(I - \frac{1}{\theta_i}A\right)$$

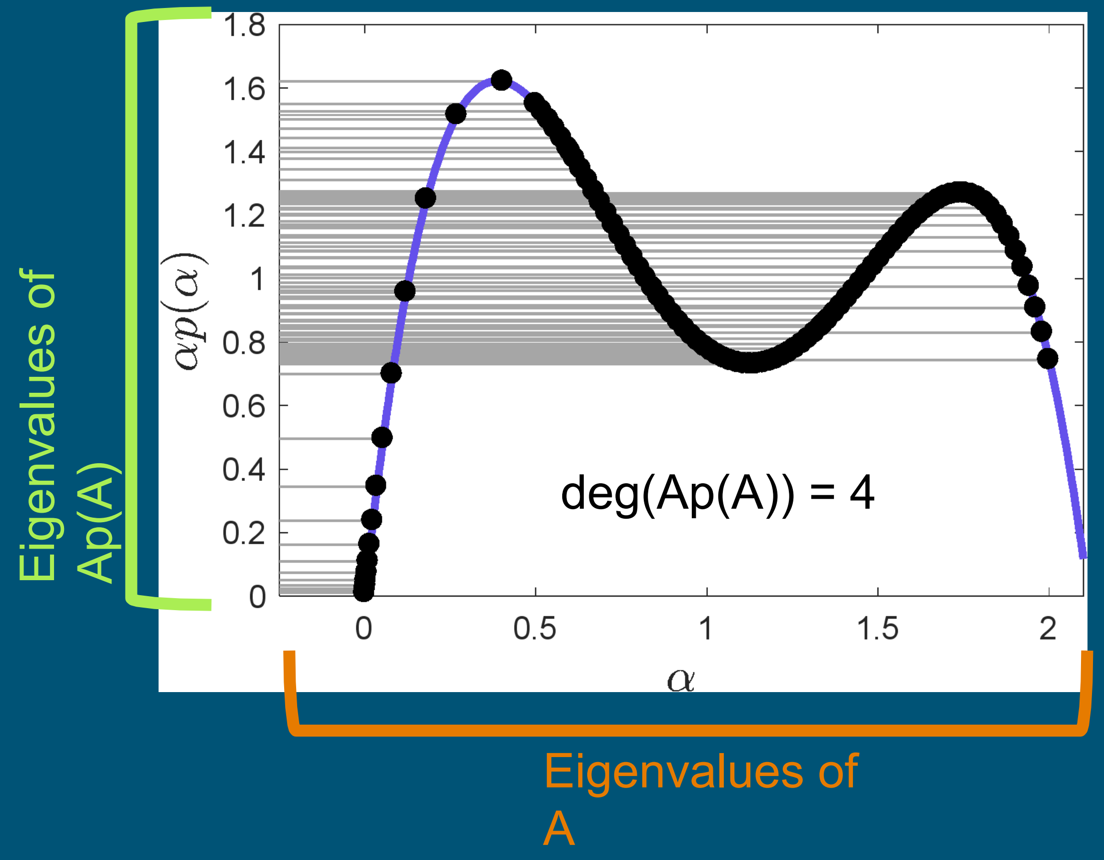
$$p(A) = \sum_{k=1}^{d}\frac{1}{\theta_k}\left(I - \frac{1}{\theta_1}A\right)\left(I - \frac{1}{\theta_2}A\right)\cdots\left(I - \frac{1}{\theta_{k-1}}A\right)$$

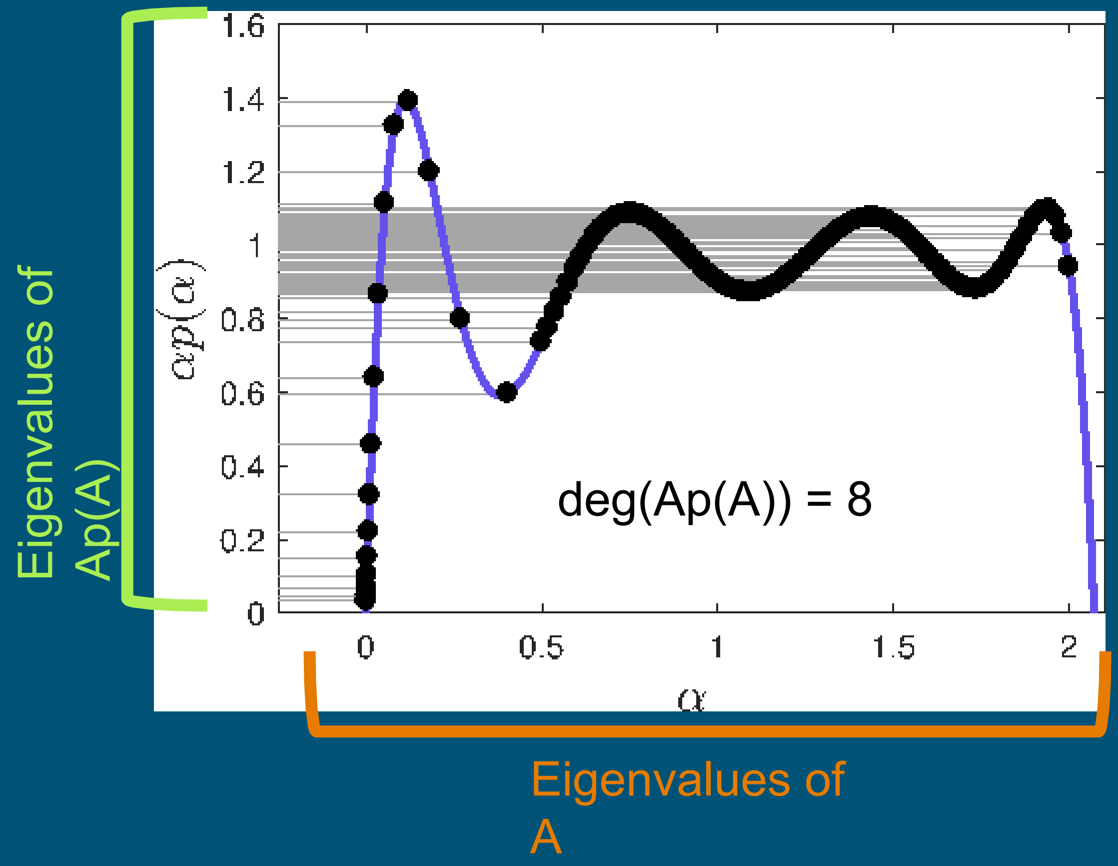
[See: Toward Efficient Polynomial Preconditioning for GMRES, J. Loe and R. Morgan, *Numerical Linear Algebra with Applications,* December 2021.]

Generating the polynomial preconditioner: p(A) has degree d-1; Ap(A) has degree d

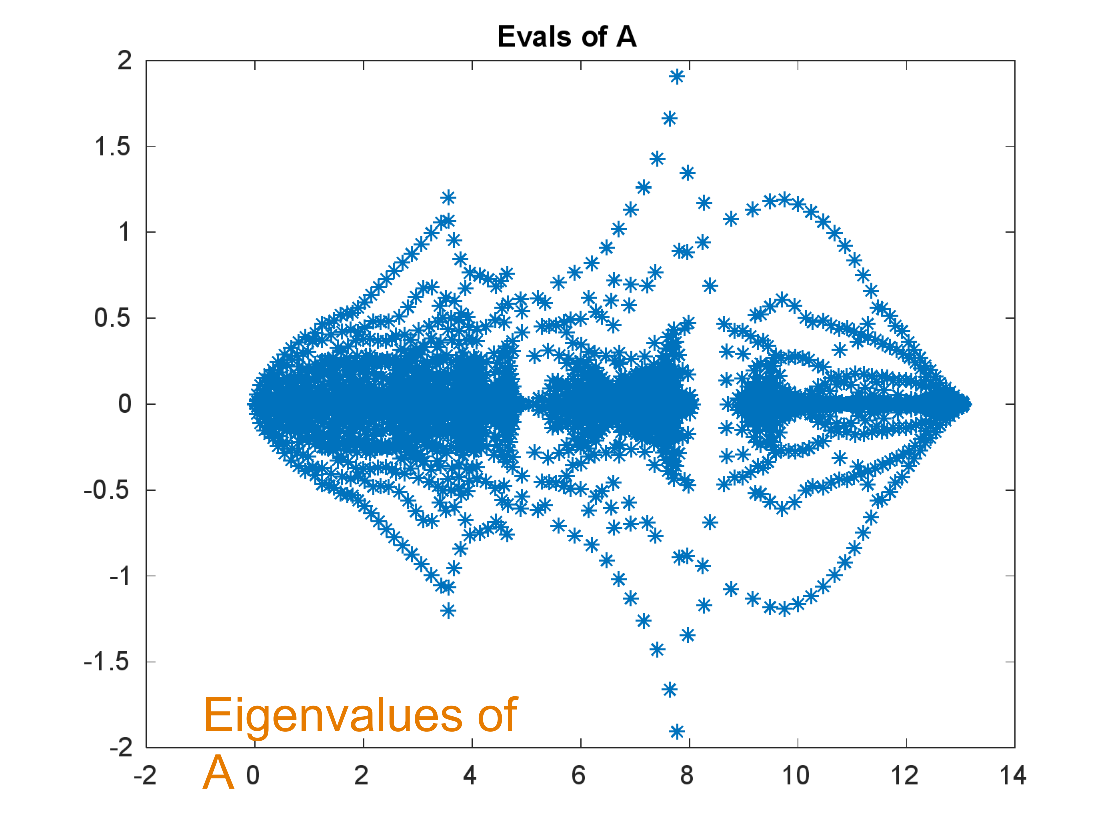
1. Run one cycle of GMRES(d) using a random starting vector.
2. Find the harmonic Ritz values $\theta_1, \ldots, \theta_d$, which are the roots of the GMRES polynomial:
   With Arnoldi decomposition $AV_d = V_{d+1}H_{d+1,d}$, find the eigenvalues of $H_{d,d} + h^2_{d+1,d} f e_d^T$, where $f = H_{d,d}^{-*} e_d$ with elementary coordinate vector $e_d = [0, \ldots, 0, 1]^T$.
3. Order the GMRES roots with *modified Leja ordering* [Bai, Hu, Reichel]
   (This ordering uses products of absolute values of differences of roots.)

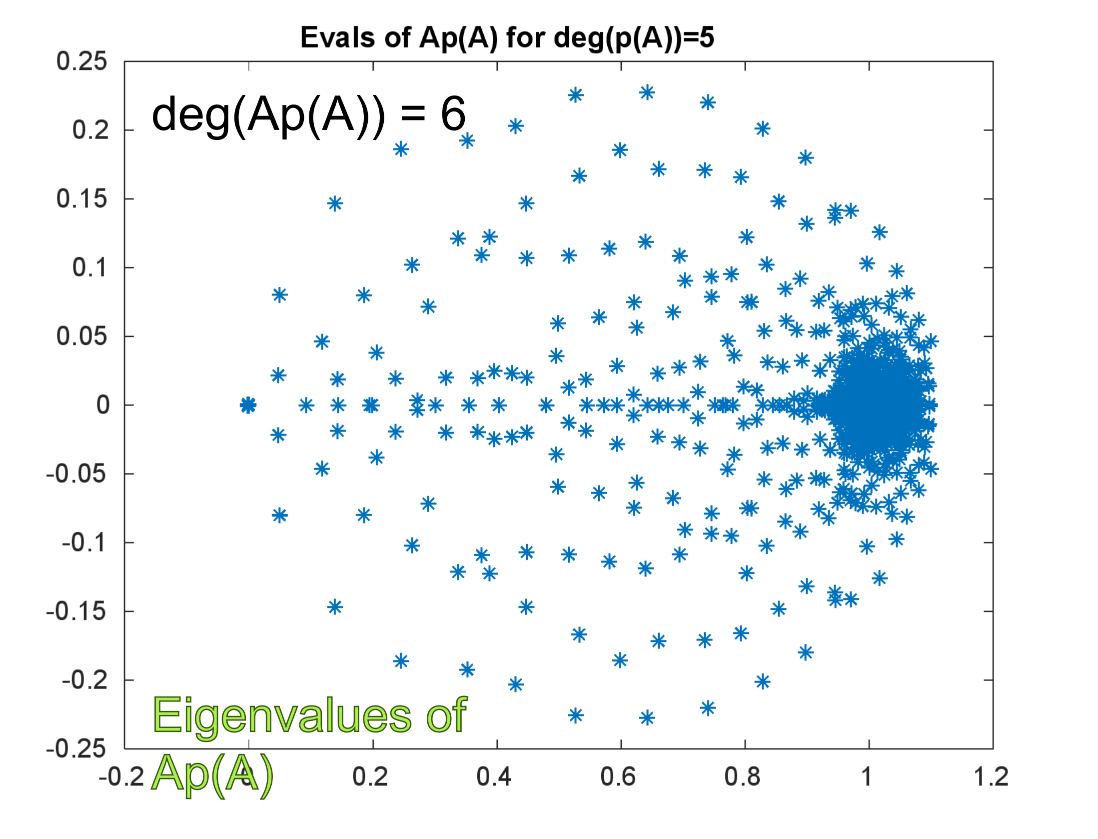
Remapping Eigenvalues (Symmetric Matrix)





Remapping Eigenvalues (Nonsymmetric- e20r0100)

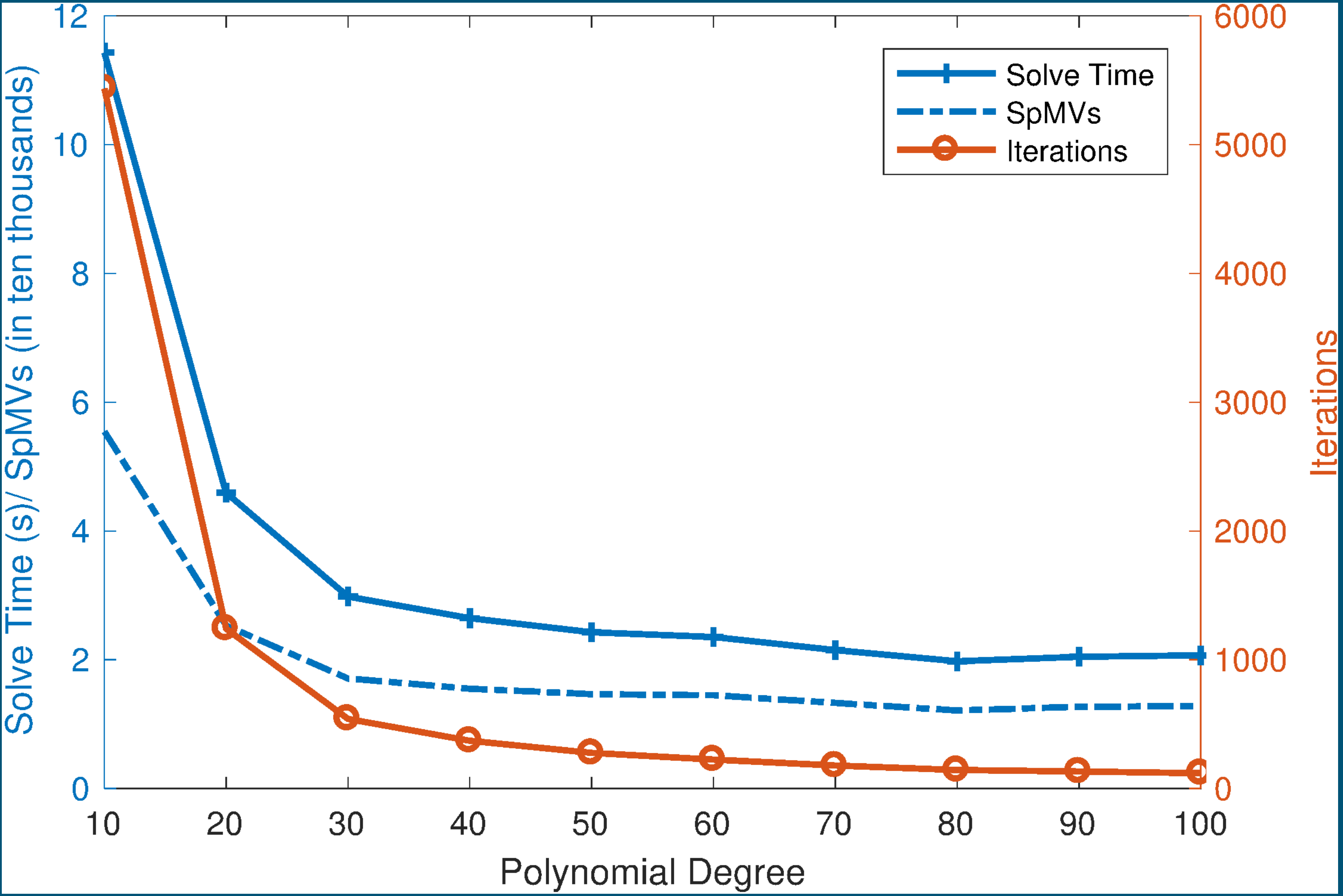




A first example:

Matrix: cfd2
n = 123,440
b = random
GMRES(50)
32 MPI ranks

No preconditioning:
113.8s, 171541 iters



Degree 80 polynomial gives 57x speedup over no preconditioning, with over 14x reduction in SpMVs.

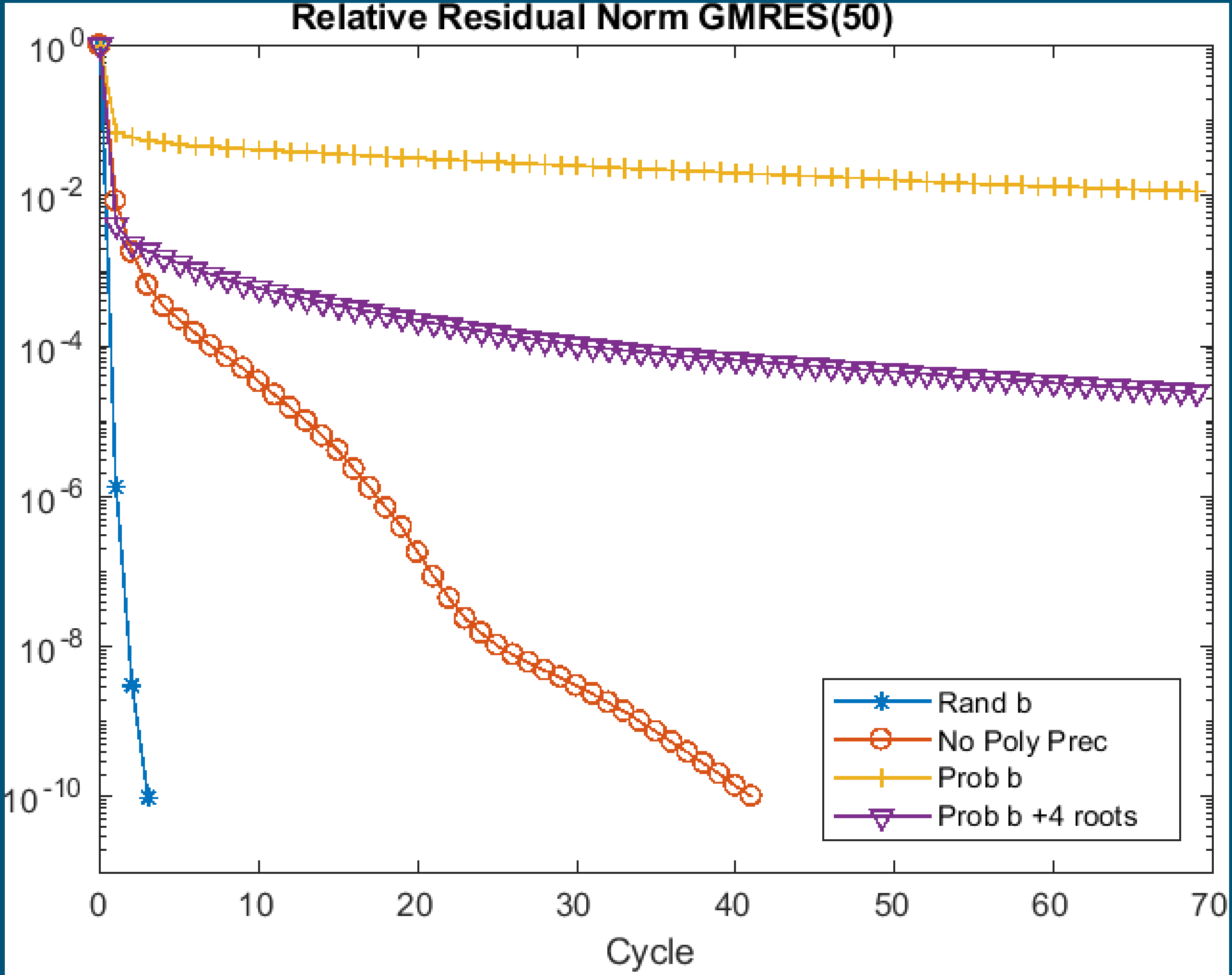
Polynomial Properties

- Choosing the starting vector for the polynomial
- Composing with other Preconditioners
- Double Preconditioning
- Added Roots for Stability
- Vs other forms of GMRES (FGMRES or GMRES non-restarted)

Why pick a random starting vector for the polynomial?

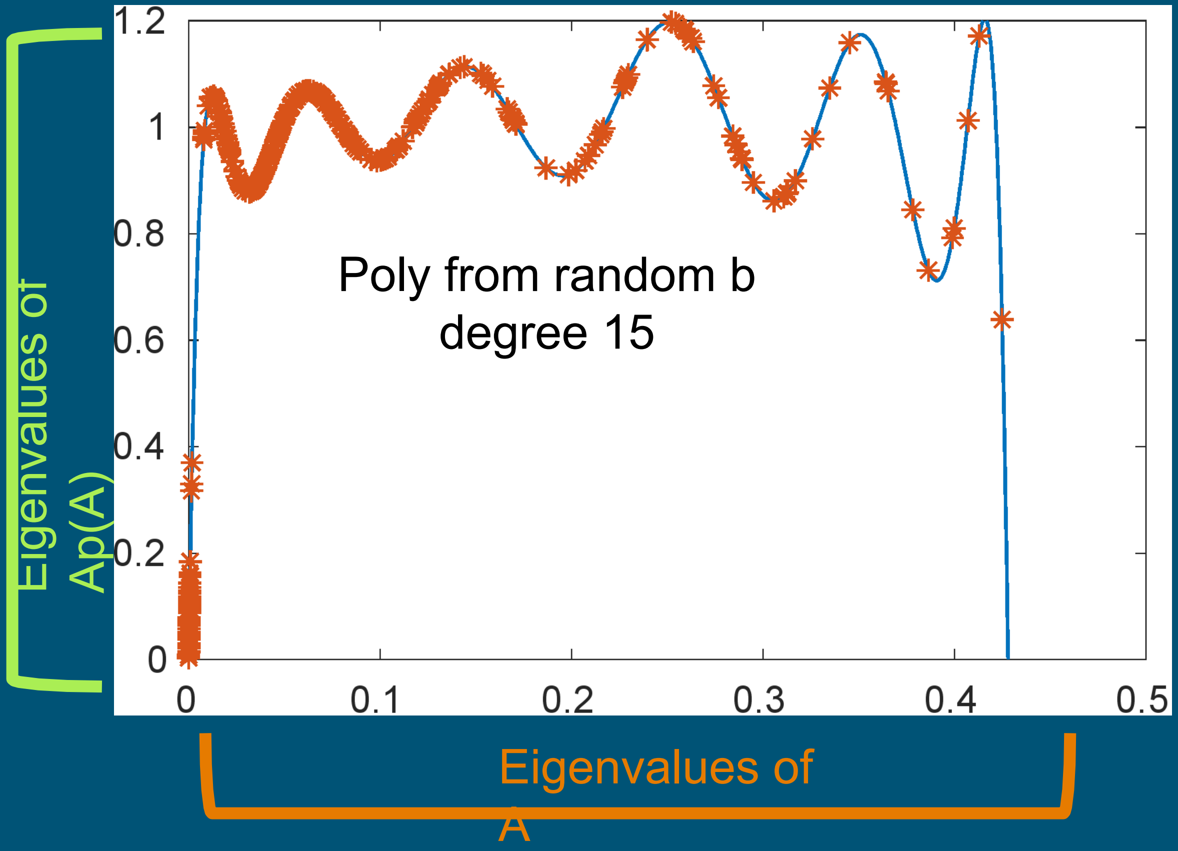
Prob b = right-hand side given with problem
Rand b = randomly generated rhs

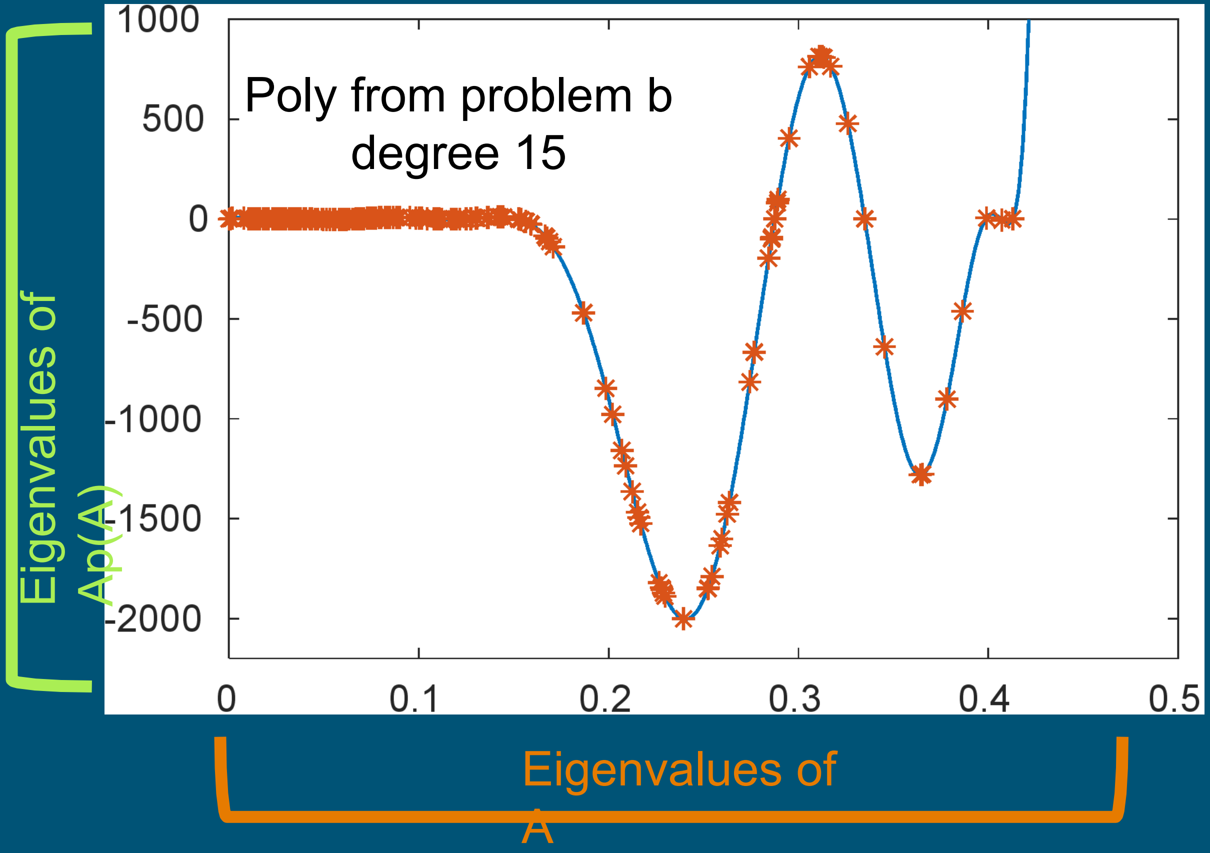
Polynomials have degree 15 (before added roots.)



Matrix: Memplus
GMRES(50)

Choosing Polynomial Starting Vector





Other flavors of Polynomial Preconditioning:

- Polynomial Preconditioning for Eigenvalue Problems

[Polynomial Preconditioned Arnoldi with Stability Control, J. Loe, M. Embree and R. Morgan, *SIAM Journal on Scientific Computing*, Vol. 43, No. 1, pp. A1-A25, 2021.]

- Compose with other preconditioners!

$$AMp(AM)y = b,$$

$$x = Mp(AM)y.$$

[More examples in: Polynomial Preconditioned GMRES in Trilinos: Practical Considerations for High-Performance Computing, J. Loe, H. Thornquist and E. Boman, *Proceedings of the 2020 SIAM Conference on Parallel Processing for Scientific Computing*, pp. 35-45, 2020.]

Polynomial Preconditioning to Accelerate ILU:

- Matrix III Stokes.
- ILU(0.001) is computed from the shifted matrix $A + 0.001I$

degree d	cycles	*mvps*	*vops*	dot products	time
No Standard Preconditioning					
1	485,042	$2.43 * 10^7$	$1.36 * 10^9$	$6.43 * 10^8$	21.6 hours
50 + 5	1072	$2.95 * 10^6$	$5.90 * 10^6$	$1.42 * 10^6$	29.9 min's
100 + 20	277	$1.66 * 10^6$	$2.43 * 10^6$	$3.71 * 10^5$	15.2 min's
With ILU Preconditioning					
1	958	47,902	$2.69 * 10^6$	$1.27 * 10^6$	211 sec's
50	3	7051	16,978	4799	13.6 sec's
100 + 10	2	7691	21,273	6668	14.8 sec's

Improvements for Biharmonic

2D Biharmonic matrix with n = 40,000

degree d	cycles	*mvps* (thousands)	*vops* (thousands)	dot products (thousands)	time
1	229,740	11,487	643,729	304,404	14.6 hours
5	11,248	2,812	33,766	14,903	1.14 hours
10	4,159	2,079	13,522	5,509	34.6 minutes
25	742	927	2,969	983	10.4 minutes
50	196	489	1,029	260	4.89 minutes
100	47	235	374	137	2.29 minutes
200	11	105	174	33.9	54.9 seconds
400 + 3	4	73.7	245	85.1	47.9 seconds
800 + 19	1	41.7	688	322	1.33 minutes

Double polynomial preconditioning!

$$\phi_2(\phi_1(A))z = b,$$
$$x = p_1(A)(p_2(\phi_1(A))z.$$

With No Preconditioning:
14.6 hours!!
(Matlab on CPU)

Table 5.1: Biharmonic matrix with $n = 40,000$.

degree deg = d_1 x d_2	cycles	*mvps* (thousands)	*vops* (thou's)	dot prod's (thou's)	time
100 = 10 x 10	117	583	903	154	5.29 minutes
225 = 15 x 15	31	348	433	41.0	3.00 minutes
400 = 20 x 20	8	153	174	10.2	1.26 minutes
900 = 30 x 30	3	99.1	107	3.66	49.5 seconds
1600 = 40 x 40	1	75.3	81.0	2.76	35.8 seconds
2500 = 50 x 50	1	77.6	83.9	3.07	36.6 seconds
3600 = 60 x 60	1	79.3	87.4	3.95	38.3 seconds

Allows high-degree polynomials with less basis vector storage and fewer dot products.

Polynomial Preconditioning vs FGMRES vs Adaptive Polynomials

- Why not just use FGMRES with GMRES as the preconditioner?
- What happens if we update the polynomial at each GMRES restart?

1. Fixed polynomial (deg(Ap(A) = d) throughout GMRES. [PP-G]
2. FGMRES with GMRES(d) as the preconditioner at each iteration. [FG]
3. Polynomial preconditioned GMRES, but get a new polynomial (deg(Ap(A) = d) at each restart based upon the most recent residual. [ChPoly]

Polynomial Preconditioning vs FGMRES vs Adaptive Polynomials

Table 5.1: Comparison with different degree polynomials between PP-GMRES (PP-G), FGMRES (FG) and PP-GMRES with polynomial changing for each cycle (Ch-Poly). GMRES(50) is used for all tests. The matrix is diagonal with entries $\frac{i^2}{n}$ and with $n = 10{,}000$.

degree d	PP-G *mvps* (thousands)	FG *mvps* (thou's)	ChPoly *mvps* (thou's)	PP-G time (seconds)	FG time (sec's)	ChPoly time (sec's)
5	5765	2468	580	3006	1166	191
10	2798	1064	335	907	472	64.0
25	1213	693	293	168	454	32.7
50	593	383	365	39.6	386	36.7
100	322	209	550	18.4	367	61.5
150	223	155	636	12.0	389	88.9
200	202	76.4	747	11.3	253	125
250	95.4	82.5	626	5.7	338	128

Polynomial Preconditioning vs FGMRES vs Adaptive Polynomials

Table 5.1: Comparison with different degree polynomials between PP-GMRES (PP-G), FGMRES (FG) and PP-GMRES with polynomial changing for each cycle (Ch-Poly). GMRES(50) is used for all tests. The matrix is diagonal with entries $\frac{i^2}{n}$ and with $n = 10{,}000$.

degree d	PP-G *mvps* (thousands)	FG *mvps* (thou's)	ChPoly *mvps* (thou's)	PP-G time (seconds)	FG time (sec's)	ChPoly time (sec's)
5	5765	2468	580	3006	1166	191
10	2798	1064	335	907	472	64.0
25	1213	693	293	168	454	32.7
50	593	383	365	39.6	386	36.7
100	322	209	550	18.4	367	61.5
150	223	155	636	12.0	389	88.9
200	202	76.4	747	11.3	253	125
250	95.4	82.5	626	5.7	338	128

Unstable Polynomials?!?

$$Ap(A) = I - \prod_{i=1}^{d} \left(I - \frac{1}{\theta_i} A\right)$$

$$p(A) = \sum_{k=1}^{d} \frac{1}{\theta_k} \left(I - \frac{1}{\theta_1} A\right) \left(I - \frac{1}{\theta_2} A\right) \cdots \left(I - \frac{1}{\theta_{k-1}} A\right)$$

What happens when one of the theta_i is much bigger than the others??

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

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

Derivative of the polynomial gets too big!!

Root Adding Example

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

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

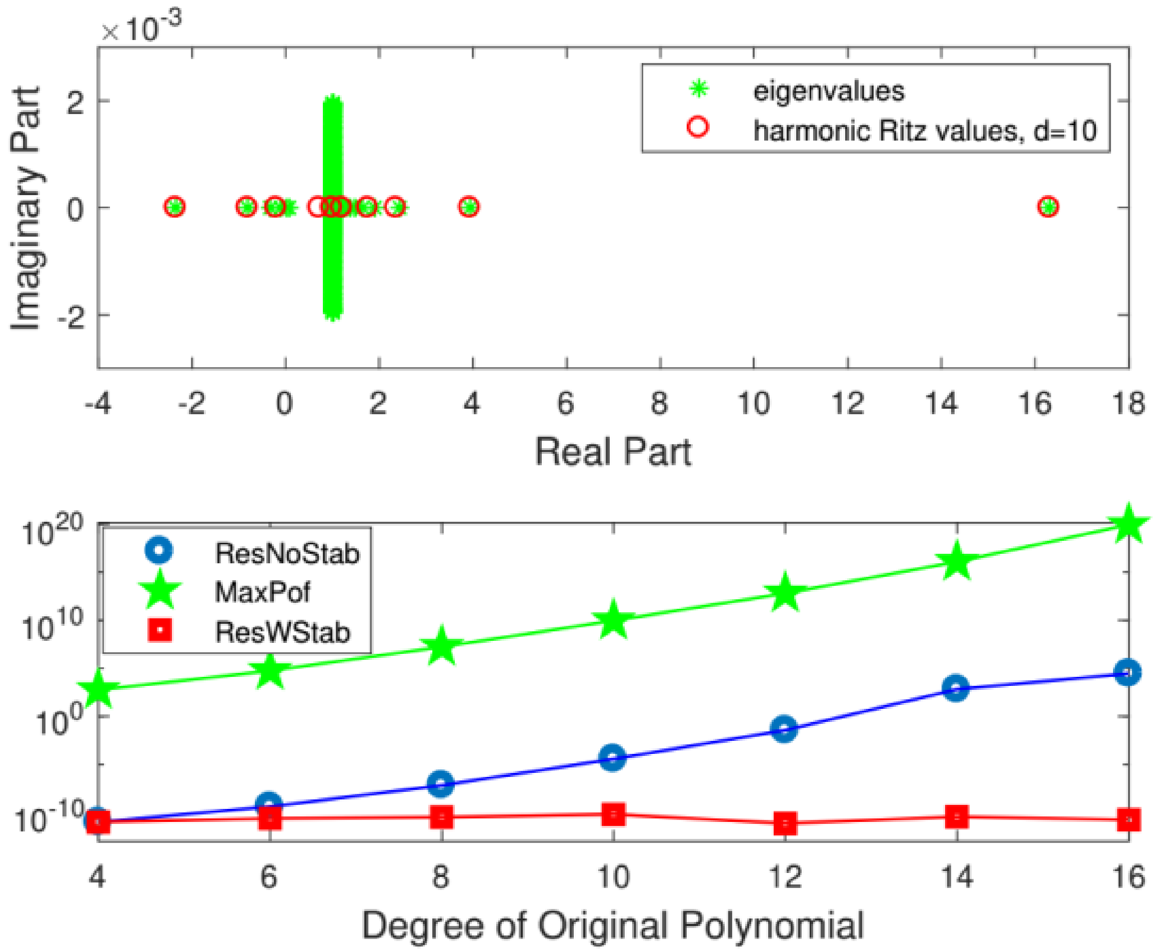


Fig. 3.2: The top half has the spectrum of OLM1000 after ILU(0) preconditioning and has the roots of the GMRES polynomial of degree 10. The bottom half gives the residual norm at the end of solving the linear equations with PP-GMRES first without stability control (circles) then with control (squares). The maximum pof values are also given (stars).

Root Adding Example

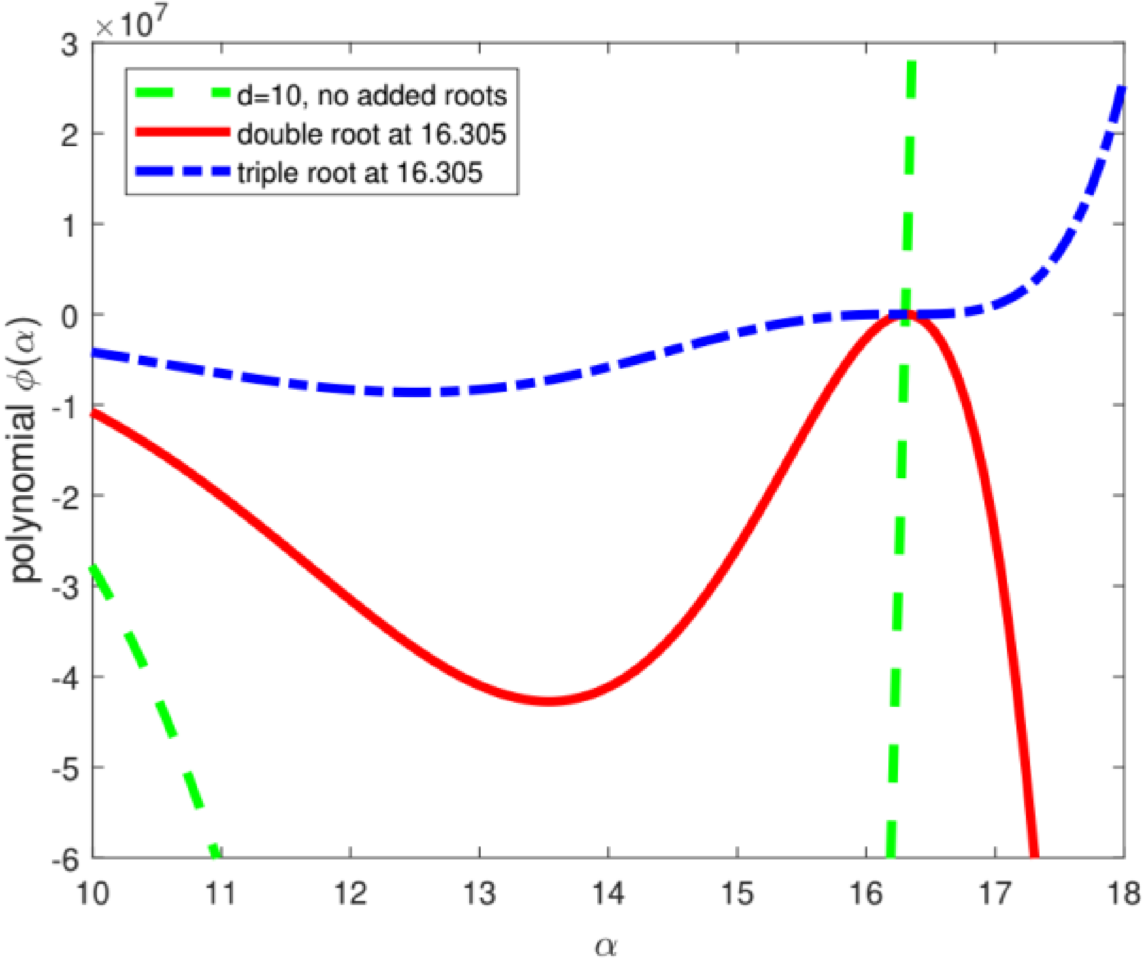


Fig. 3.3: Graph of the degree 10 GMRES polynomial for the OLM1000 matrix with ILU(0) preconditioning (dashed line), then the polynomial with one root added at 16.305 (solid line) and finally with two added roots at 16.305 (dash-dot line).

Root Adding Example

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

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

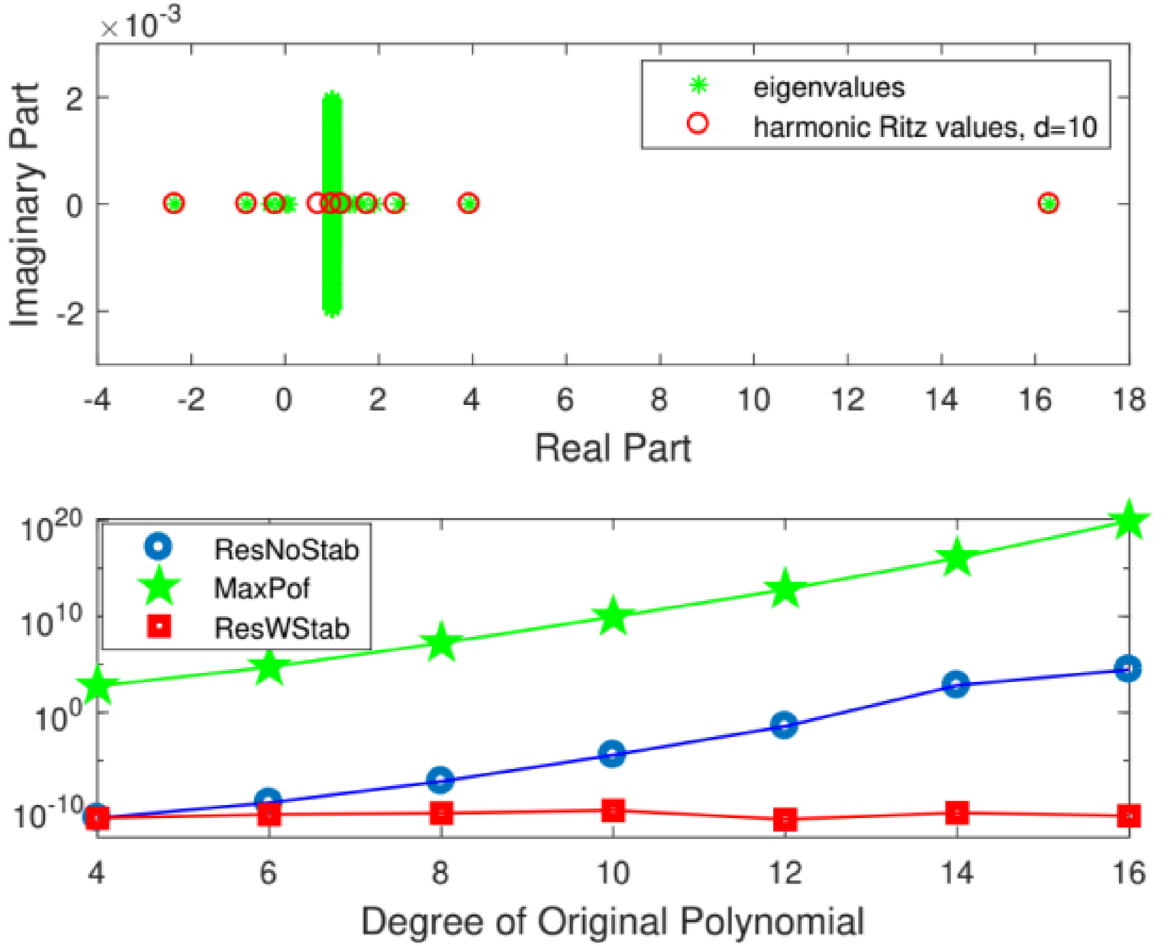
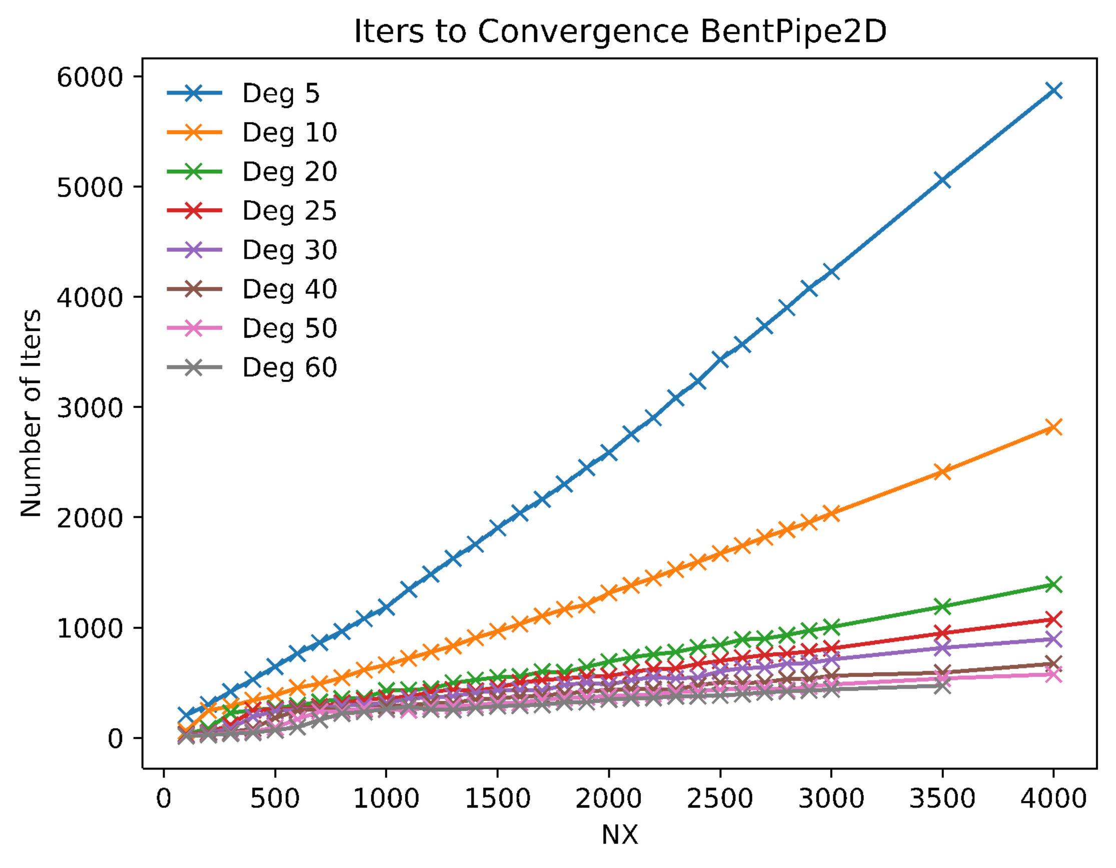


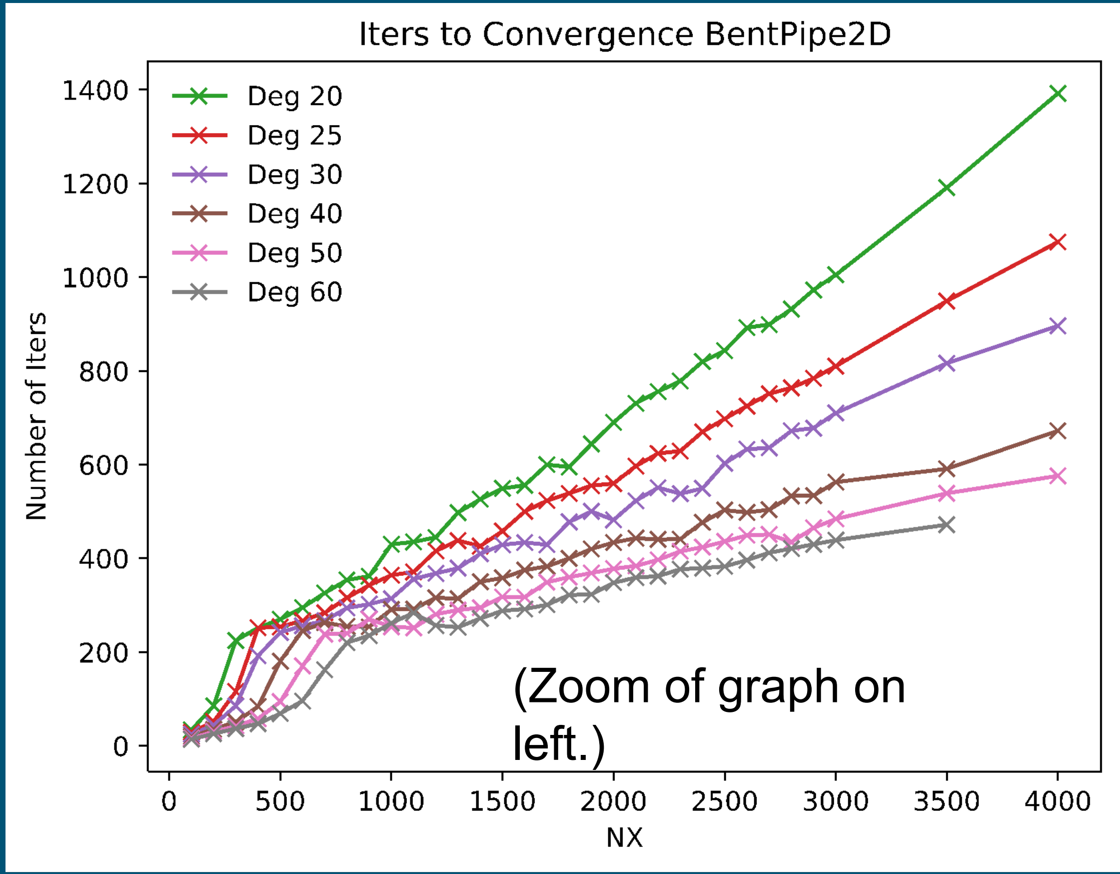
Fig. 3.2: The top half has the spectrum of OLM1000 after ILU(0) preconditioning and has the roots of the GMRES polynomial of degree 10. The bottom half gives the residual norm at the end of solving the linear equations with PP-GMRES first without stability control (circles) then with control (squares). The maximum pof values are also given (stars).

Polynomial Preconditioning Extensions for Large-Scale Computing

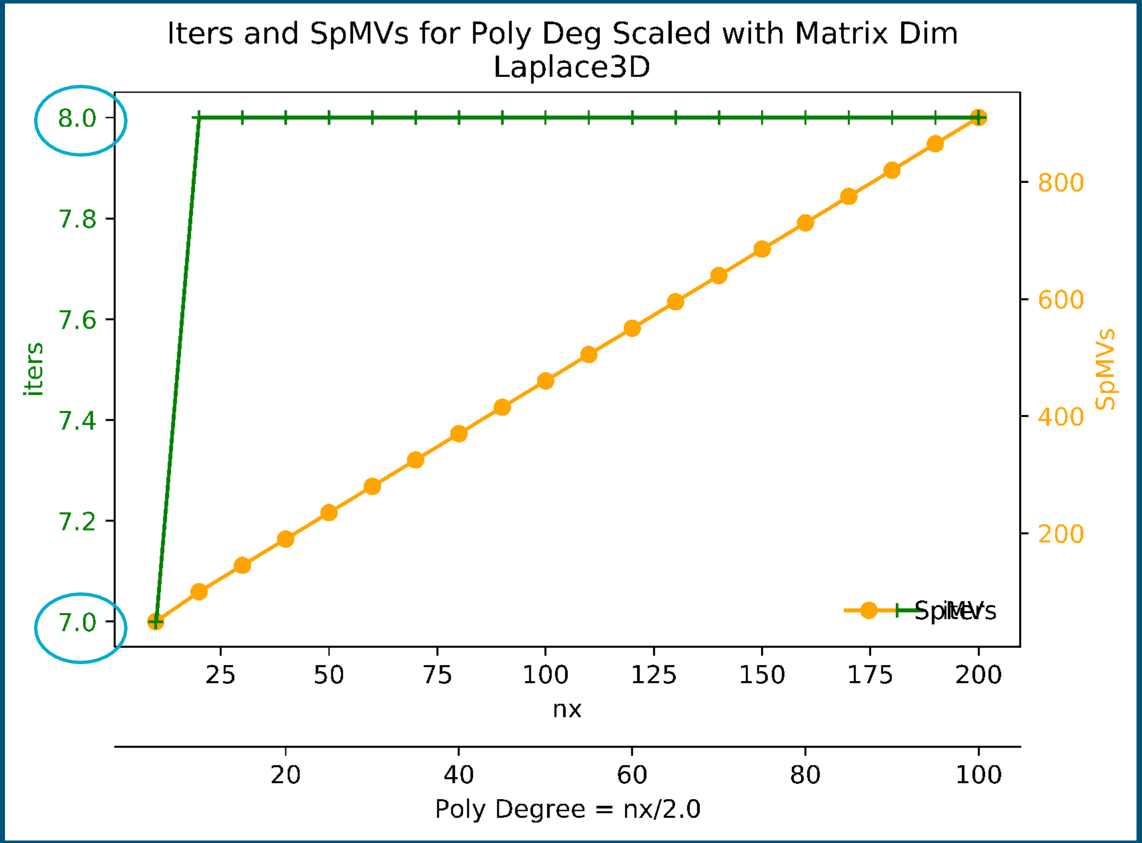
- Mixed Precision
- “Weak” Scaling
- Communication – Avoiding for MPI
- Usefulness for GPUs

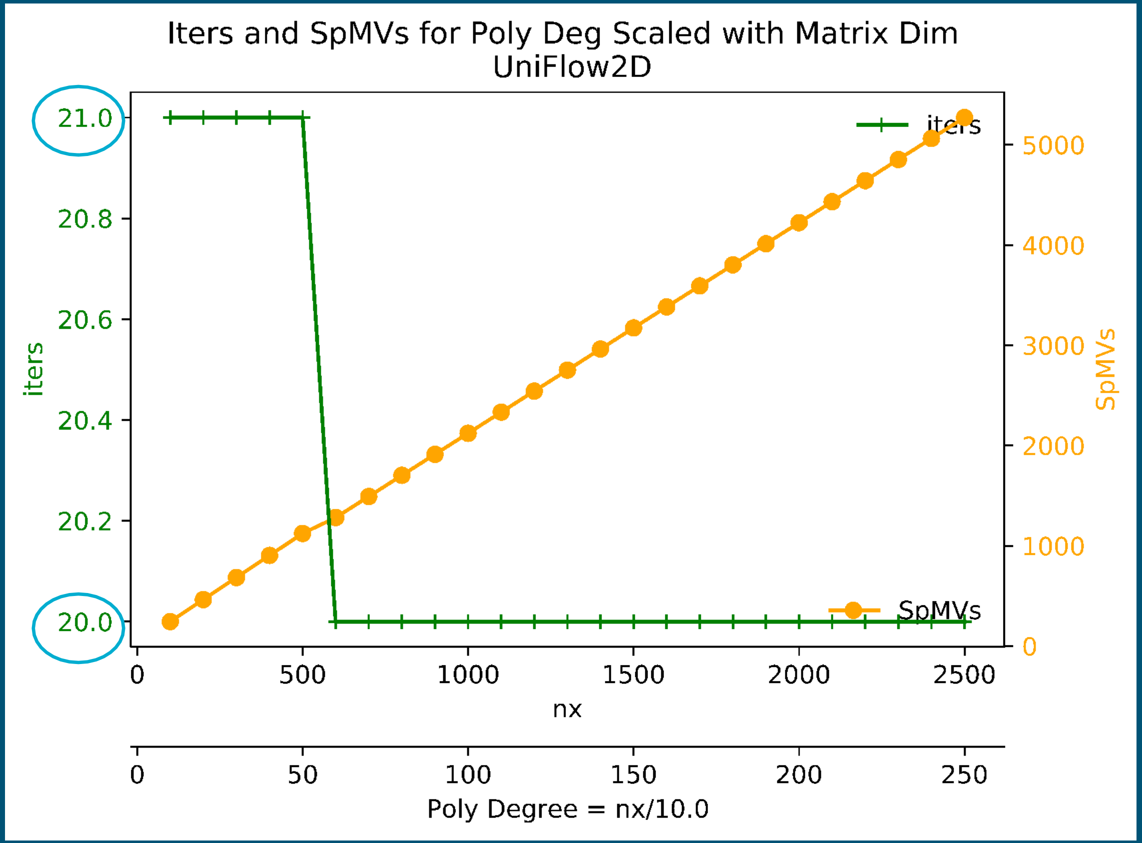
Scaling: Fix Polynomial Degree and Refine the Mesh?





Scaling: Increase Polynomial Degree wrt Matrix Size





Potential for Polynomial Preconditioning on GPUs:

- Very Parallelizable
  - Unlike, say, ILU. (Triangular solves need level sets; might not be able to extract parallelism.)
- Straightforward to code for GPUs
  - Need axpy and SpMV.
  - Easier to port than, say, multigrid.
- Great for Matrix-Free problems!

Bridging the gap between Polynomial Preconditioning and Applications:

- Simplify software interface in Trilinos (Belos linear solvers package).
- One-size-fits-all or automated degree selection.
  - Possible application- as currently run with ILUk(1):
  - Ex mini application: 10-20 timesteps, each with nonlinear solve x 2-10 linear solves = 20 to 200 linear solves!
    (10 to 300+ iterations each; can't use same degree polynomial.)
  - Full size application = 1000x larger!
- Better guarantees for problems with indefinite spectrum (Work in progress.)
- Build communication with application teams!

Takeaway points:

- Polynomial Preconditioning with the GMRES polynomial is power for linear systems and eigenvalue problems!
  - Can accelerate existing preconditioners.
  - Automated root-adding for stability.
  - Reliably generate high-degree polynomials.
- Competitive with FGMRES and with non-restarted Krylov methods
  - Reduced orthogonalization costs and basis storage costs!
  - May be useful for MPI communication avoiding.
- Paving a Path toward Parallel Computing:
  - Easily ported to GPU code.
  - Straightforward to parallelize on GPU.
  - Mixed-precision advantages
- Applying to Real-Life Problems:
  - Make reliable for indefinite problems.
  - Simplify software and parameters.
  - Communicate with applications teams!

Thank you!