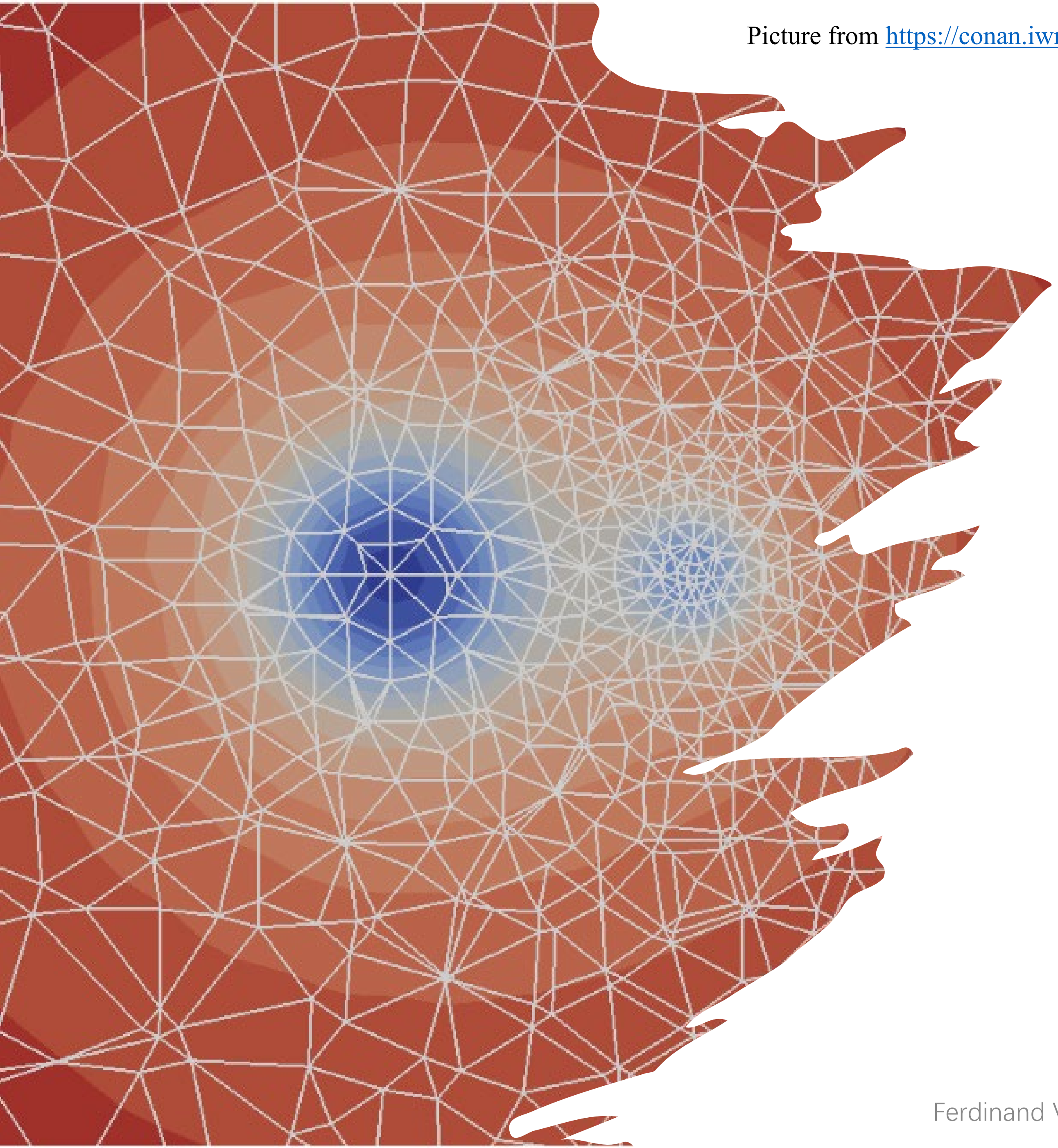
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Polynomial Preconditioners

Ferdinand Vanmaele



Picture from https://conan.iwr.uni-heidelberg.de/data/teaching/finiteelements_ws2017/num2.pdf

Introduction

- Discretization of **partial differential equations**
  - Solve $Ax = b$ with A matrix of high dimension n
- Common in science and industry
  - Fluid dynamics
  - Astrophysics
  - Biochemistry
  - Economics
  - etc...

Introduction

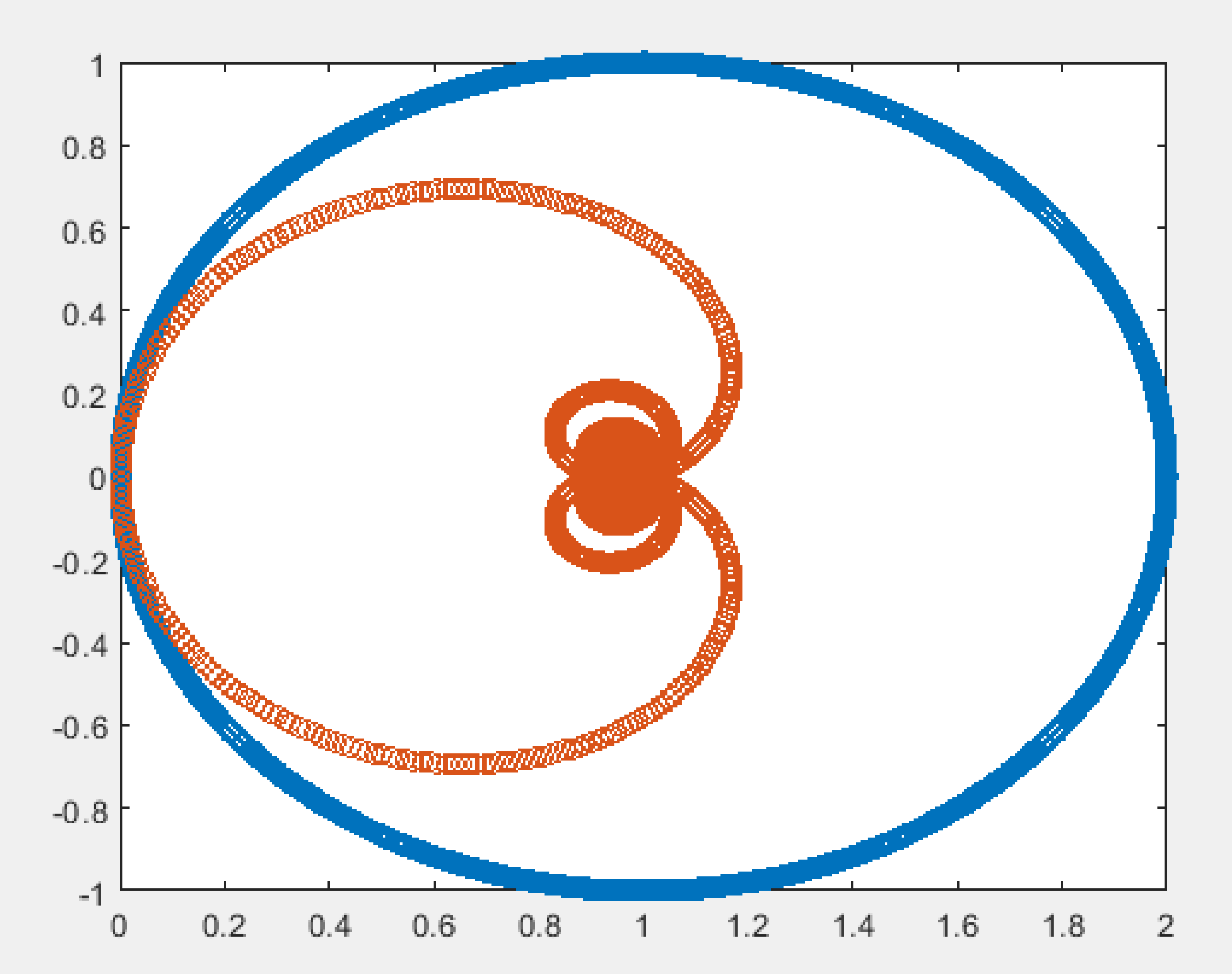
- Examples
  - Circle Eigenvalue Matrix
  - Model problem
- Iterative solvers
  - **Goal:** Reduce storage and complexity requirements
  - Series $(x_i) \to x$ of approximate solutions to $Ax = b$
- Preconditioned system $M^{-1}Ax = M^{-1}b$
  - **Goal:** Improve convergence rate of (x_i)
  - Same solution as $Ax = b$

Example: Circle Eigenvalue Matrix

- A block diagonal, size $n = 2000$
- 2×2 blocks $\begin{bmatrix} 1+\cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & 1+\cos(\alpha) \end{bmatrix}$, $\alpha \in \left\{0, \frac{2\pi}{n}, \frac{4\pi}{n}, \dots, \frac{1998\pi}{n}\right\}$
- Eigenvalues on unit circle in complex plane
- Condition number[1] $K(A) \approx 637$
- Difficult problem without preconditioning

1: $K(A) = \|A^{-1}\|\|A\|$ denotes how sensitive the solution x is to perturbations in the matrix A or the right-hand side b.

Example: Circle Eigenvalue Matrix

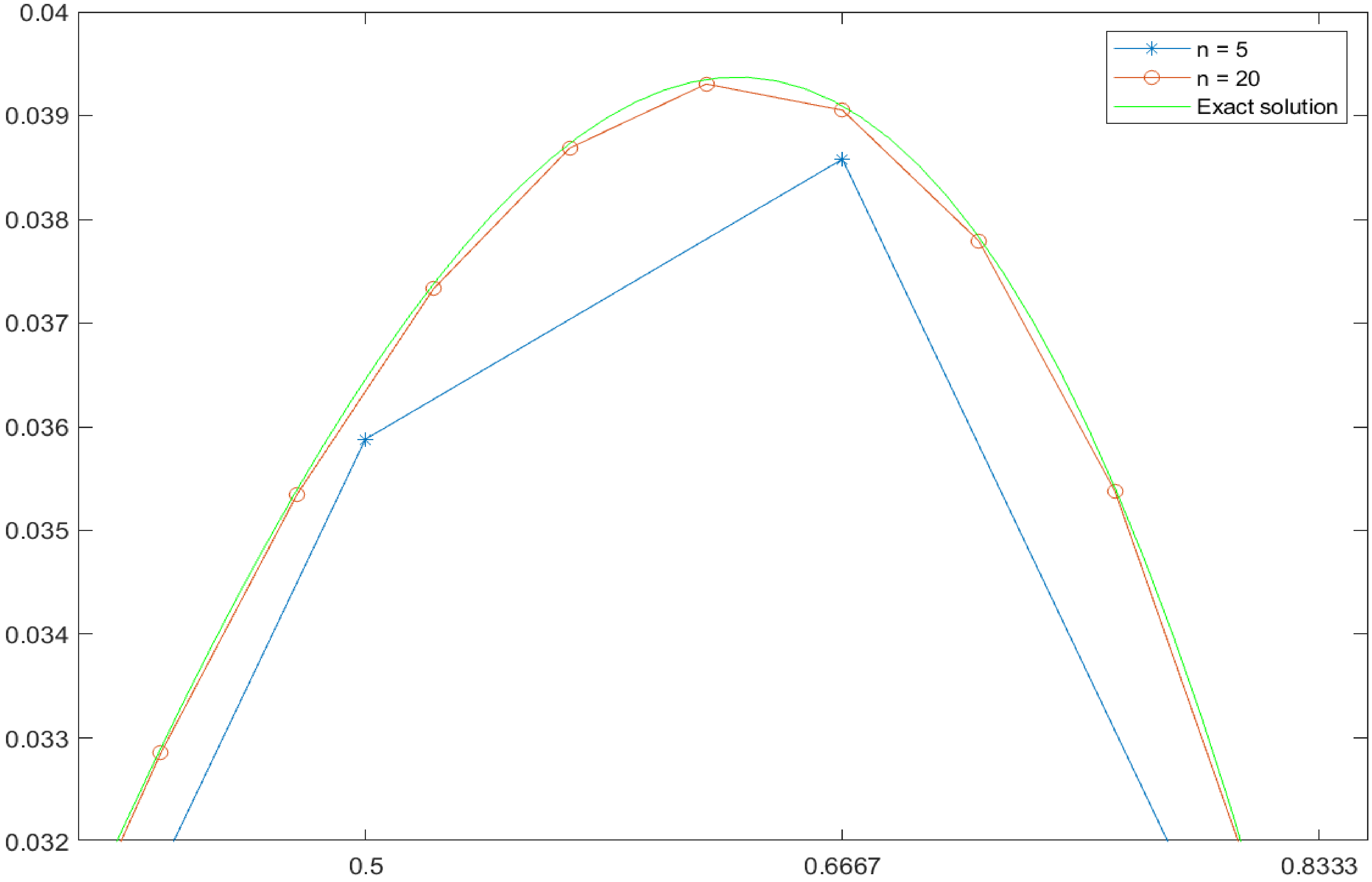


- Polynomial $M^{-1} = s(A)$
- $\deg(s) = 10$
- $K(M^{-1}A) \approx 69.9$
- Eigenvalues of M^{-1} in $\mathbb{C}$

Example: Model problem

- Problem
  - $-u''(x) = f(x)$ for $x \in (0,1)$
    $u(0) = u(1) = 0$
- Discretization
  - $x_i = i \times h,\ i = 0, \dots, n+1$, where $h = 1/(n+1)$
- Central difference approximation
  - $-u_{i-1} + 2u_i - u_{i+1} = h^2 f(x_i)$
- Linear system
  - $Ax = f$ with $A \in R^{n \times n}$

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$


0.04
0.039
0.038
0.037
0.036
0.035
0.034
0.033
0.032
0.5
0.6667
0.8333
n = 5
n = 20
Exact solution

Iterative solvers

- Most entries of A are zero
  - Store A as a *sparse matrix*
  - Multiple formats available, e.g. CSR
- Direct methods
  - Decomposition $A = LU$
  - L and U may become dense: *fill-in*
  - High storage and complexity requirements for large n
- Alternative: Iterative methods

Iterative solvers

- Starting approximation x_0
- Method $x_i \rightarrow x_{i+1}$
  - $x_{i+1} = x_i + \alpha N^{-1} r_i$ with $r_i = b - Ax_i$ (stationary)
  - $x_{i+1} = x_i + constant_i * search\ direction_i$ (non-stationary)
- Stopping criterion
  - Relative residual: $\frac{\|b - Ax_i\|}{\|b\|} \leq \epsilon$
    - ⚠ $\|b - Ax_i\| \leq \|A\| \|x_i - x\|$
  - Typically $\epsilon = 10^{-1} \dots 10^{-12}$
- Optimality condition?

Iterative solvers

- **Idea:** Let $N = I$, $\alpha = 1$, $x_0 = 0$, $r_0 = b$. Write out $x_{i+1} = x_i + r_i$:
  - $x_1 = x_0 + r_0 = r_0$
  - $x_2 = x_1 + r_1 = r_0 + (b - Ax_1) = r_0 + (r_0 - Ar_0)$
    $= 2r_0 - Ar_0$
  - $x_3 = x_2 + r_2 = 2r_0 - Ar_0 + (b - Ax_2)$
    $= 2r_0 - Ar_0 + (r_0 - A(2r_0 - Ar_0))$
    $= 3r_0 - 3Ar_0 + A^2 r_0$
  - …
- Implication:
  - $x_i \in x_0 + span\{r_0, Ar_0, \dots, A^{i-1}r_0\}$ $\longleftarrow$ Krylov subspace $\mathcal{K}^i(A, r_0)$

Iterative solvers

- **GMRES**
  - Minimize $\|b - Ax_i\|$ for $x_i \in x_0 + \mathcal{K}_i(A, r_0)$
  - Works for general matrices A
- Limitations
  - No short recurrence for general matrices 😥
    - Orthonormalization of $\mathcal{K}_i$
    - $x_i = x_0 + (\alpha_1 v_1 + \cdots + \alpha_i v_i)$
  - After m steps, set $x_m = x_0$ and restart algorithm
    - **GMRES(m)**
    - Convergence not guaranteed! (unless A s.p.d.)

Iterative solvers

- **Building blocks**
  - Vector updates (SAXPY)
    - Easy to parallelize ✔
  - Inner products
    - Parallelization: synchronization between all processes 🙄
  - Sparse matrix-vector products (SpMVs)
    - Parallelization: synchronization between neighbours 😋
  - Matrix-matrix product
    - Determine preconditioner (before iteration start, later)

Preconditioning

- Solve $M^{-1}Ax = M^{-1}b$
  - Same solution as original system $Ax = b$
  - Between $M = I$ and $M = A$
- Desired properties
  - Matrix norm $||I - M^{-1}A||$ small
  - Eigenvalues of $M^{-1}A$ close to 1
- Polynomials
  - $A^{-1} = \frac{1}{a_0}(-a_n A^{n-1} - a_{n-1}A^{n-2} - \cdots - a_1 I)$ (Cayley-Hamilton)
  - $M^{-1} = y_{d+1}A^d + y_d A^{d-1} + \cdots + y_2 A + y_1 I$ with $d \ll n$

Preconditioning

- Determine coefficients of M^{-1}
  - **Ansatz:** Minimize $\|(I - M^{-1}A)v_0\|$ for some $v_0 \neq 0$
- Algorithm
  1. Choose v_0
     - A random vector, e.g. uniform distributed, gives good results in practice
  2. Construct power basis $\mathrm{Y} = \{v_0, Av_0, \dots, A^d v_0\}$
     - ⚠ Columns of Y lose linear independence!
  3. Solve least squares problem $\min\|v_0 - s(A)Av_0\|$
     - Solve normal equations $(AY)^T AYy = (AY)v_0$
     - $y = (y_1 \cdots y_{d+1})$ coefficients of M^{-1}

Example: Model problem $(n = 5)$

$$A = \begin{pmatrix} 72 & -36 & 0 & 0 & 0 \\ -36 & 72 & -36 & 0 & 0 \\ 0 & -36 & 72 & -36 & 0 \\ 0 & 0 & -36 & 72 & -36 \\ 0 & 0 & 0 & -36 & 72 \end{pmatrix} \qquad v0 = \begin{pmatrix} \frac{1}{3} \\ -1 \\ 0 \\ -1 \\ -1 \end{pmatrix} \qquad AY = \begin{pmatrix} 60 & 7344 & 917568 \\ -84 & -10800 & -1384128 \\ 72 & 9504 & 1213056 \\ -36 & -3888 & -575424 \\ -36 & -1296 & 46656 \end{pmatrix}$$

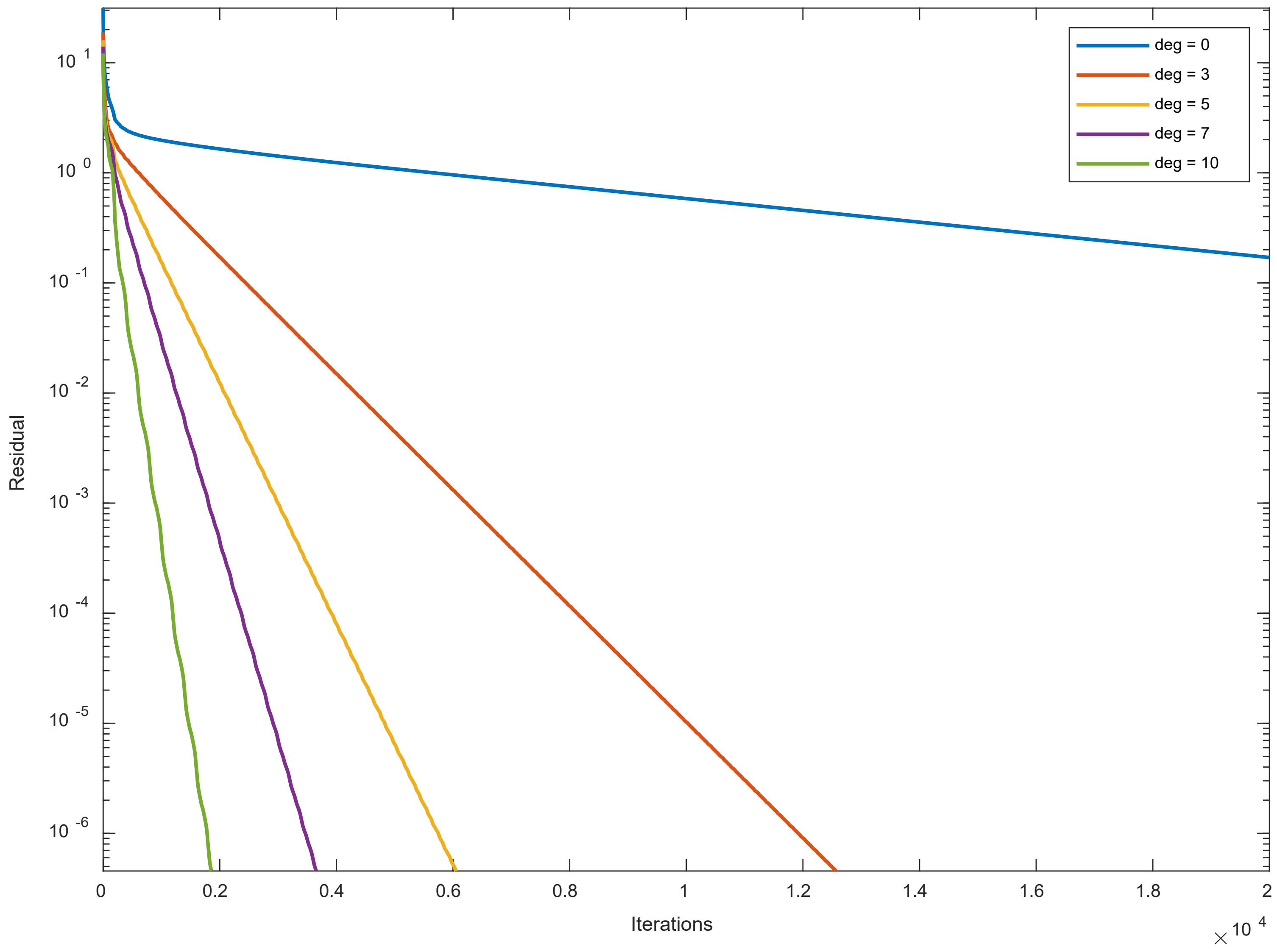
$$(AY)^T AY = \begin{pmatrix} 18432 & 2218752 & 277696512 \\ 2218752 & 277696512 & 35392868352 \\ 277696512 & 35392868352 & 4562535776256 \end{pmatrix} \qquad (AY)^T v_0 = \begin{pmatrix} 176 \\ 18432 \\ 2218752 \end{pmatrix}$$

Solution: $y_3 = 5.1599e - 06$, $y_2 = -0.0012148$, $y_1 = 0.078039$

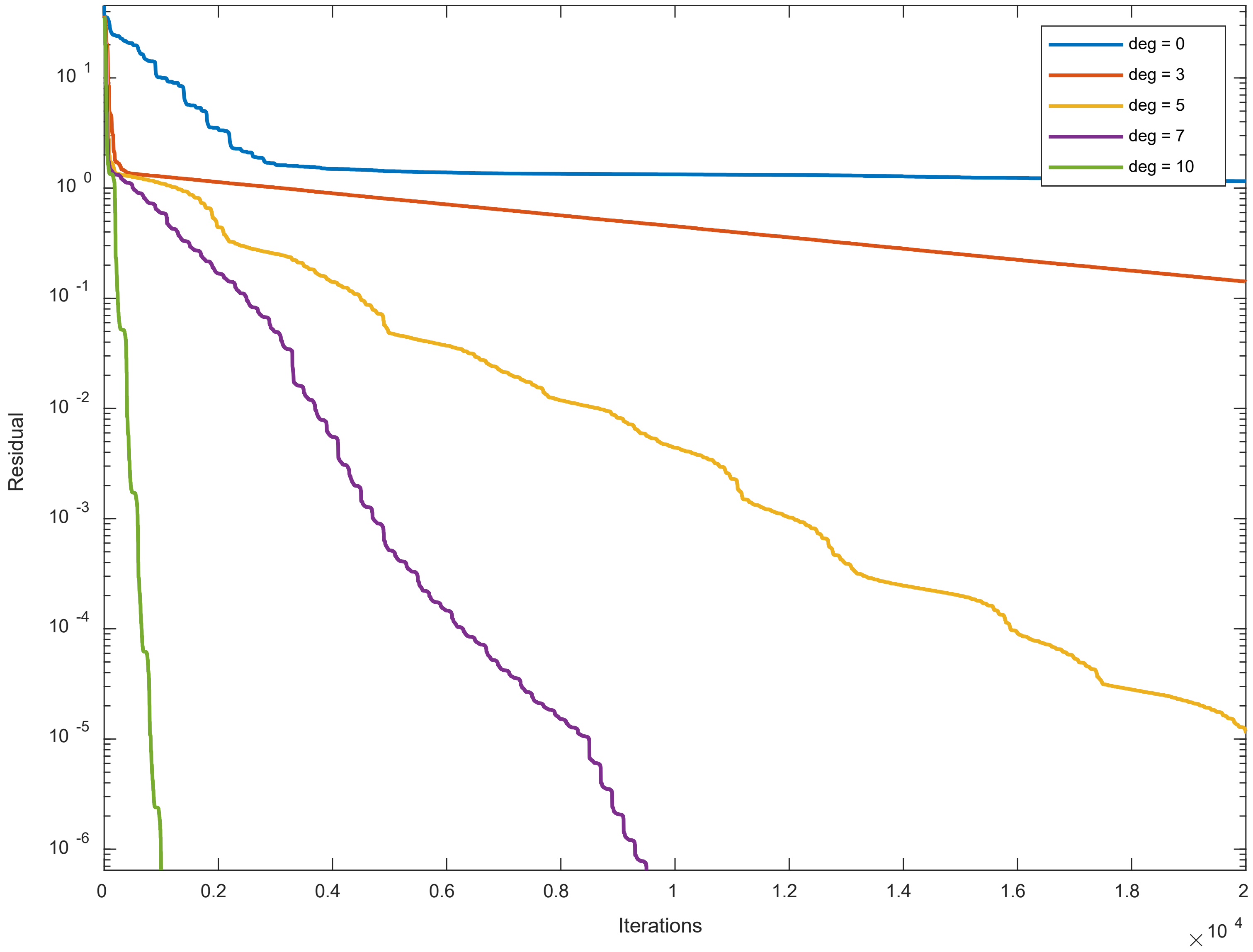
Preconditioning

- Practical considerations
  - Do not store $\mathrm{M}^{-1} = s(A)$ explicitly (vector products)
    - $z^{(1)} = y_{d+1}Av + y_d v$
    - $z^{(2)} = A * z^{(1)} + y_{d-1}v$
    - $z^{(3)} = A * z^{(2)} + y_{d-2}v$
    - $z^{(d)} = A * z^{(d-1)} + y_1 v$

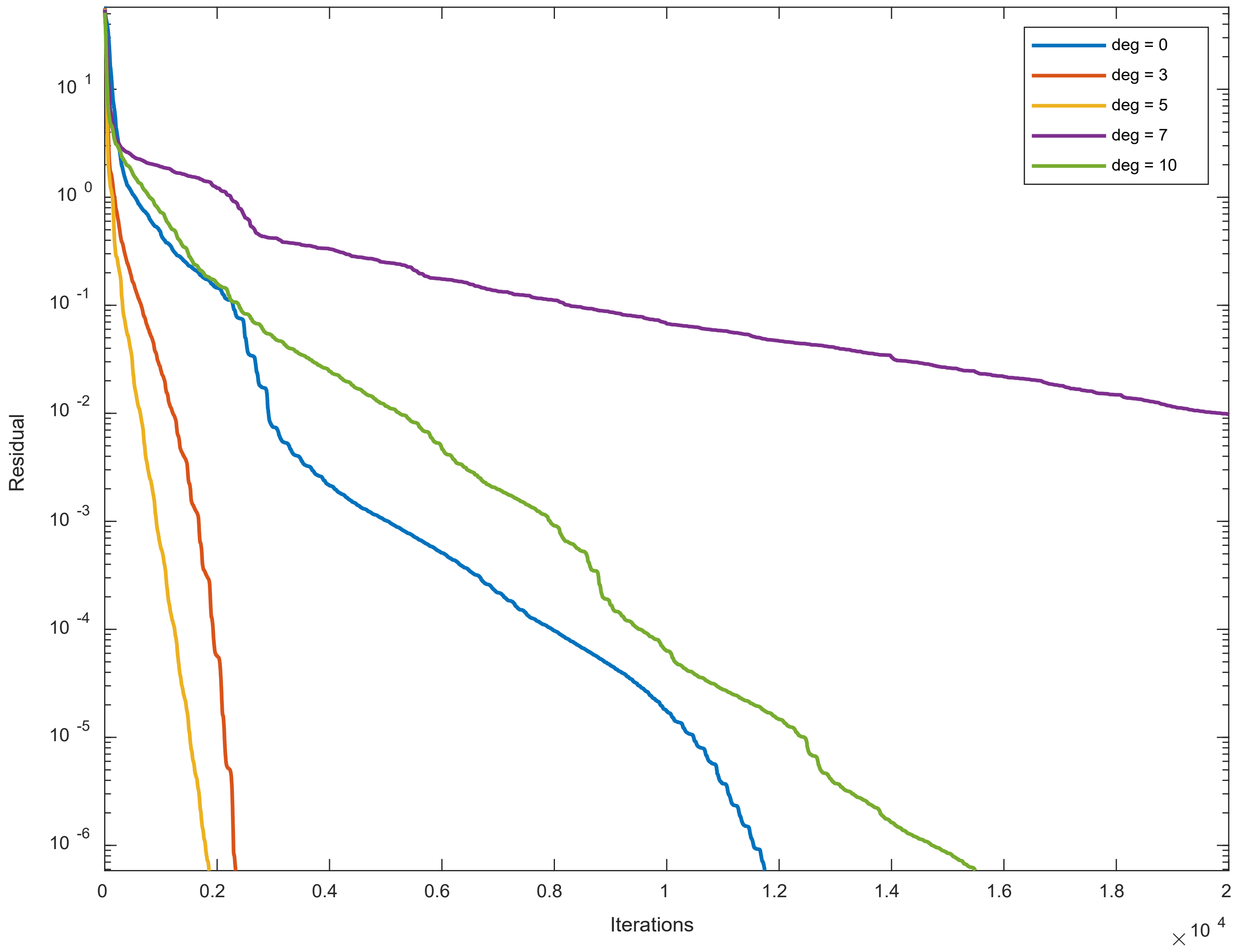
    $\mathrm{M}^{-1}\mathrm{v} = A(A \cdots (y_{d+1}Av + y_d v) + \cdots y_2 v) + y_1 v$
  - Set degree of the polynomial
    - Cost of multiplication $(AY)^T AY$ → lower degree
    - Difficulty of problem $Ax = b$ → higher degree
  - ⚠ Residuals relative to preconditioned system
    - Use *right preconditioning*: solve $AM^{-1}u = b$, $Mx = u$



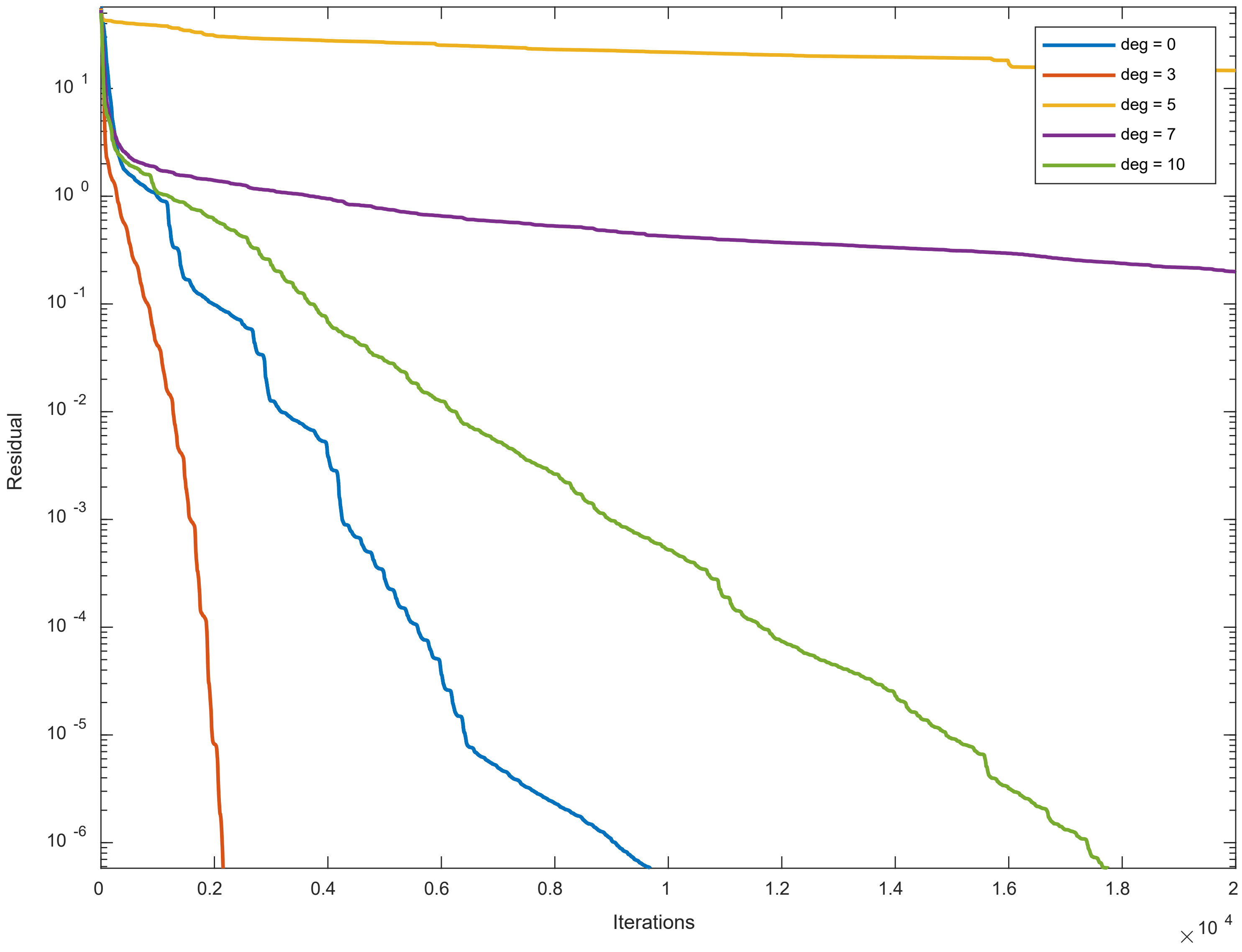
- **Circle Eigenvalue Matrix**
- $b \sim N(0,1)$
- GMRES(100)
- 20,000 Iterations
- $\epsilon = 10^{-8}$
  - Gradual improvement with raised degree 👍



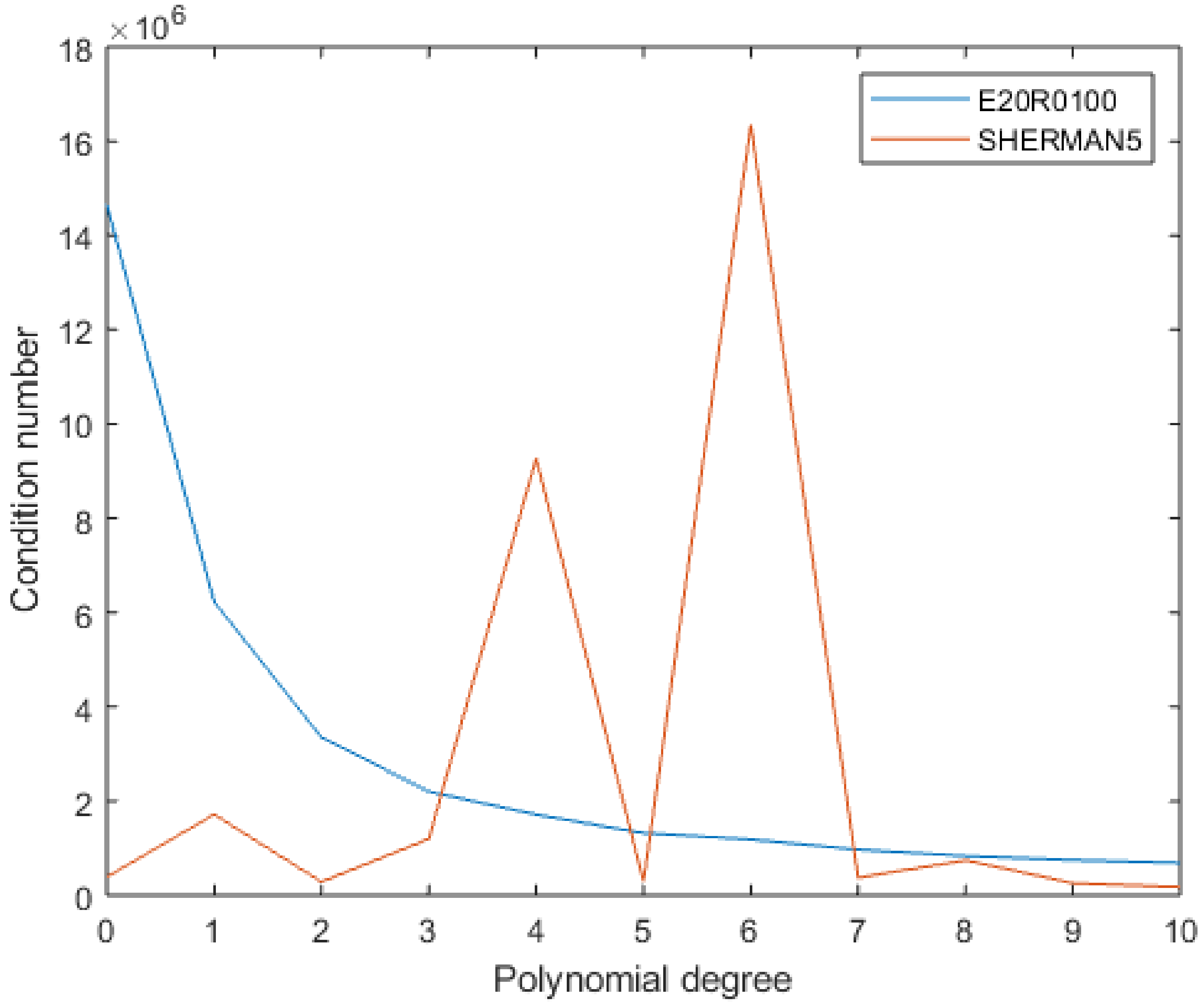
- **E20R0100**
- $b \sim N(0,1)$
- GMRES(100)
- 20,000 Iterations
- $\epsilon = 10^{-8}$
  - Achieved by deg 10 polynomial 🎉



- **SHERMAN5** *(Trial 1)*
- $b \sim N(0,1)$
- GMRES(100)
- 20,000 Iterations
- $\epsilon = 10^{-8}$
  - Results vary strongly with b and v_0 😔



- **SHERMAN5** *(Trial 2)*
- $b \sim N(0,1)$
- GMRES(100)
- 20,000 Iterations
- $\epsilon = 10^{-8}$
  - Results vary strongly with b and v_0 😟



- $K(s(A)A)$ (estimated)
- 1 order of magnitude improvement for most problems
- Erratic for SHERMAN5 🙈

Parallelization

Inner p. \ deg	**deg = 0**	**deg = 3**	**deg = 5**	**deg = 7**	**deg = 10**
CEM	1,010,000	633,330	304,176	183,285	92,385
BiDiag1	190,981	18,711	9,261	5,526	2,541
BiDiag2	2,691	630	423	288	153
S1RMQ4M1	510,000	510,000	303,451	171,553	84,051
E20R0100	1,010,000	1,010,000	1,010,000	479,750	50,203

TABLE I
INNER PRODUCTS FOR POLYNOMIALS OF DIFFERENT DEGREE

Parallelization

SpMVs \ deg	deg = 0	deg = 3	deg = 5	deg = 7	deg = 10
CEM	20,401	50,886	36,715	29,565	20,622
BiDiag1	20,013	7,594	5,631	4,491	2,862
BiDiag2	284	255	273	274	209
S1RMQ4M1	20,801	82,001	73,079	55,111	37,070
E20R0100	20,401	81,001	121,401	76,855	11,087

TABLE II
SPMVS FOR POLYNOMIALS OF DIFFERENT DEGREE

References

- Iterative Methods for Sparse Linear Systems
  https://www-users.cs.umn.edu/~saad/IterMethBook_2ndEd.pdf
- Templates for the Solutions of Linear Systems: Building Blocks for Iterative Methods
  https://www.netlib.org/templates/templates.pdf
- Polynomial Preconditioned GMRES to Reduce Communication in Parallel Computing
  https://arxiv.org/abs/1907.00072
- Seminar report 😊
  https://github.com/fvanmaele/polynomial-preconditioning

Thank you for your attention! 🤗