Iterative Solvers

Gustav Cedergrund, Kevin Stull

APPM 4600 – Final Project

Iterative Solvers

Gustav Cedergrund, Kevin Stull

APPM 4600 – Final Project

Project Overview

Our project covered applying Iterative Solvers to...

Introductory Material: Linear Systems ($\boldsymbol{Ax^* = b}$)

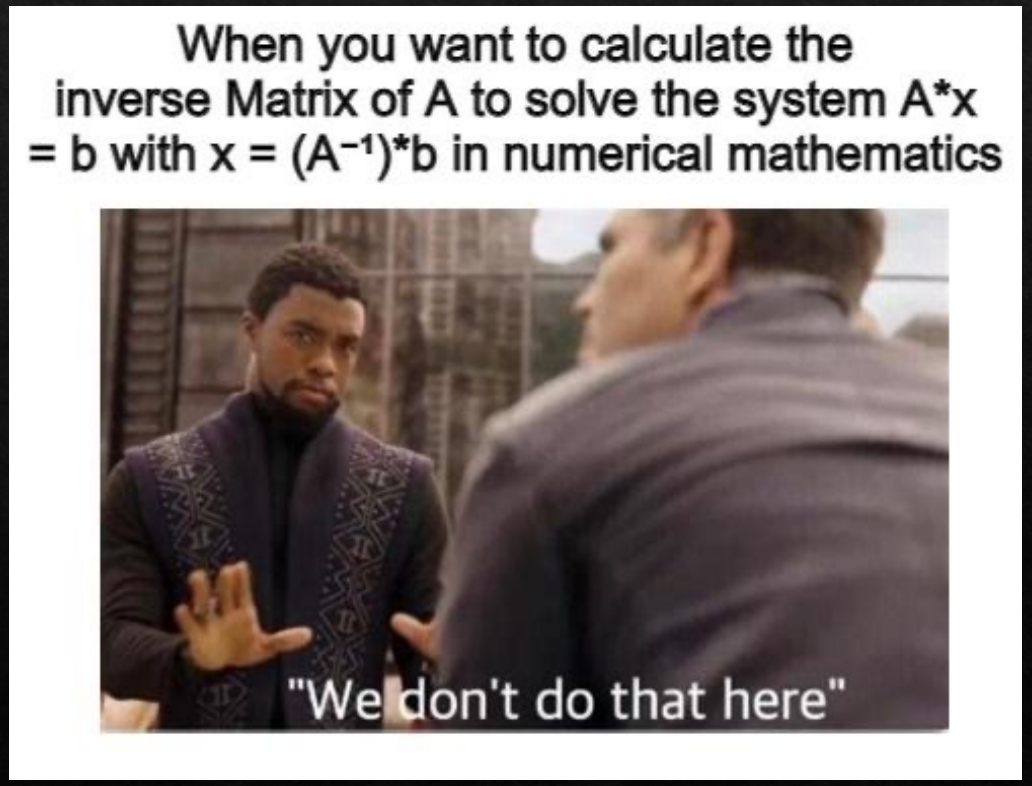
- Richardson's Iteration
- Generalized Minimum Residual (GMRES)

Independent Extension: Eigenvalue Problem ($\boldsymbol{Av = \lambda v}$)

- Power Method & QR Iteration
- Real World Application of Eigenvalue Iterations

Why Iterative Methods for Linear Systems

- Operate by refining an initial guess x_0 over a sequence of successive approximations
  - $\{x_n\} \to x^*$
- Direct Inversion has poor runtime of $O(n^3)$.
- Iterative solvers can be applied in specific cases at a fraction of the computational cost.



Richardson's Iteration

Linear System → Fixed Point

$$\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b} \Longrightarrow \boldsymbol{x}^* = (\boldsymbol{I} - \alpha\boldsymbol{A})\boldsymbol{x}^* + \alpha\boldsymbol{b}$$

$$\boldsymbol{x}_{k+1} = g(\boldsymbol{x}_k) = (\boldsymbol{I} - \alpha\boldsymbol{A})\boldsymbol{x}_k + \alpha\boldsymbol{b}$$

Thus, we have…

$$\frac{||\boldsymbol{x}^* - \boldsymbol{x}_{k+1}||}{||\boldsymbol{x}^* - \boldsymbol{x}_k||} \leq ||\boldsymbol{I} - \alpha\boldsymbol{A}||.$$

Need $\alpha \in \mathbb{R}$ such that $||\boldsymbol{I} - \alpha\boldsymbol{A}|| < 1$

Require that $\boldsymbol{A}$ is positive definite.

Richardson's Iteration

Require that $\boldsymbol{A}$ is positive definite.

- Matrix is symmetric, so $\boldsymbol{I} - \alpha\boldsymbol{A}$ is symmetric as well
  - Symmetric matrices → norm = spectral radius.
- All eigenvalues of $\boldsymbol{A}$ are >0
  - define eigenvalues of $\boldsymbol{A}$ as $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$
  - Thus, eigenvalues of $\boldsymbol{I} - \alpha\boldsymbol{A}$ are $1 > 1 - \alpha\lambda_1 \geq \cdots \geq 1 - \alpha\lambda_n$

$$\left|\left|\boldsymbol{I} - \alpha\boldsymbol{A}\right|\right| = \max\{1 - \alpha\lambda_1, 1 - \alpha\lambda_n\}$$

$$\alpha = \frac{2}{\lambda_1 + \lambda_n} \Longrightarrow \left|\left|\boldsymbol{I} - \alpha\boldsymbol{A}\right|\right| < 1$$

Richardson's Iteration

$$\alpha = \frac{2}{\lambda_1 + \lambda_n} \Longrightarrow ||\boldsymbol{I} - \alpha \boldsymbol{A}|| < 1$$

Rate of Convergence is linear dependent entirely on condition number, that is, $\kappa(\boldsymbol{A}) = \frac{\lambda_n}{\lambda_1}$.

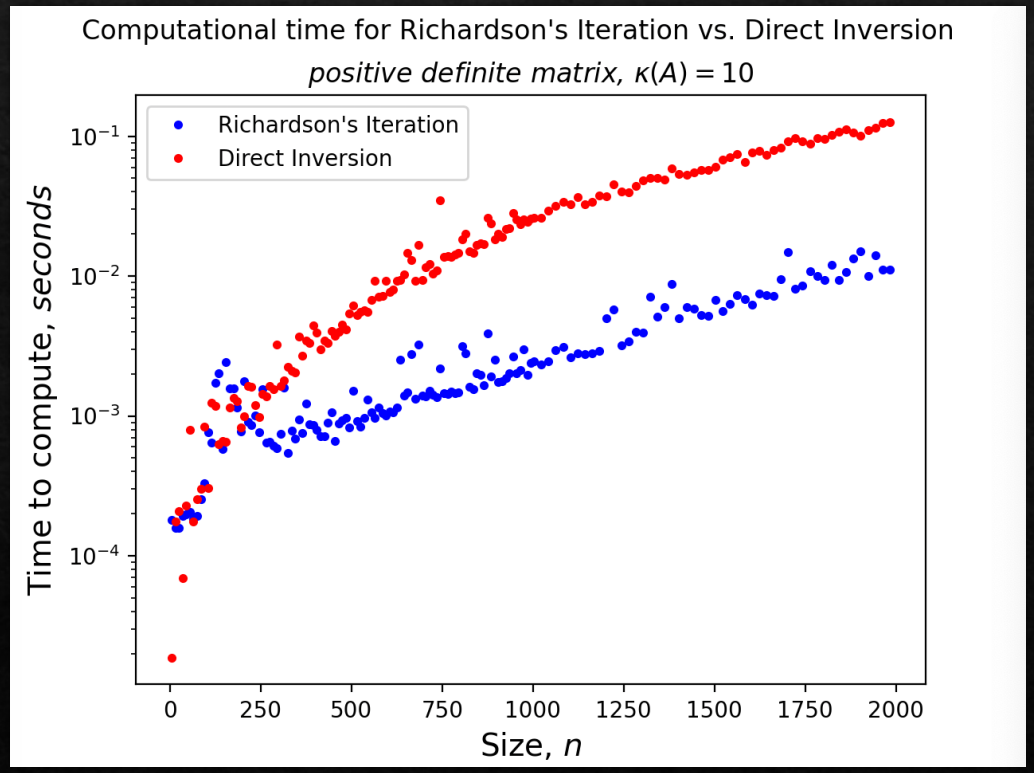
If $\lambda_n = \lambda_1$, then quadratic convergence

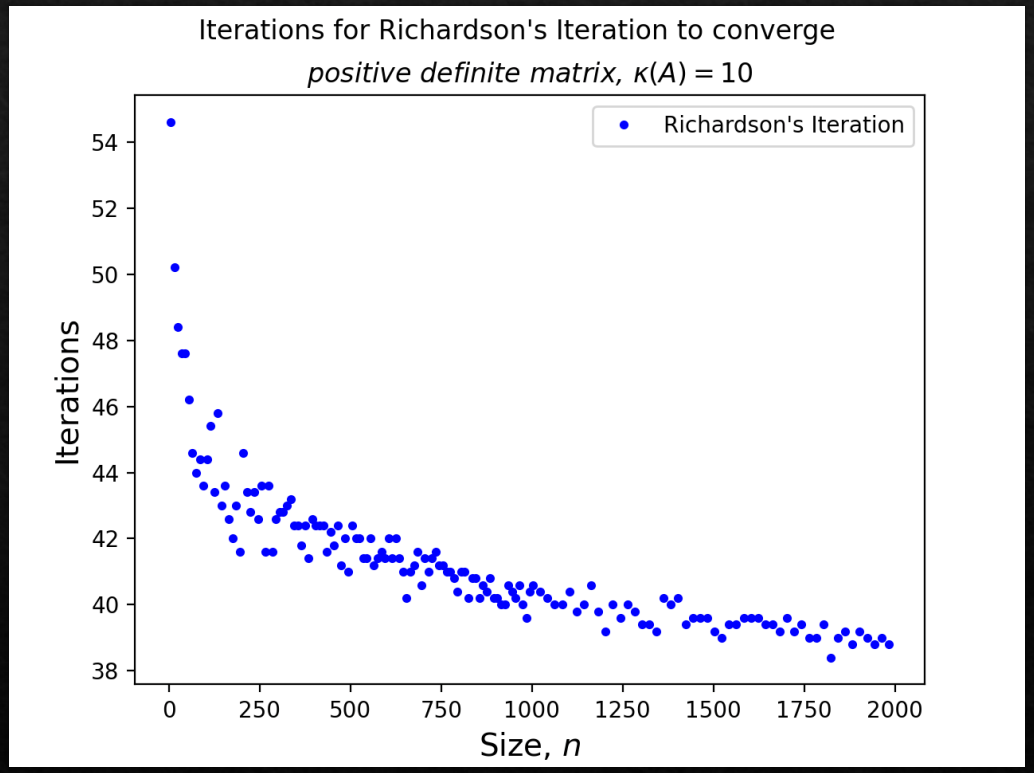
Pseudocode:

Algorithm 1 - Richardson's Iteration

$\mathbf{x_0} \leftarrow \mathbf{0}$ // arbitrary initial guess
for $k = 0, 1, 2, \ldots$ **do**
  $\mathbf{x_{k+1}} \leftarrow (\mathbf{I} - \alpha\mathbf{A})\mathbf{x_k} + \alpha\mathbf{b}$
end for

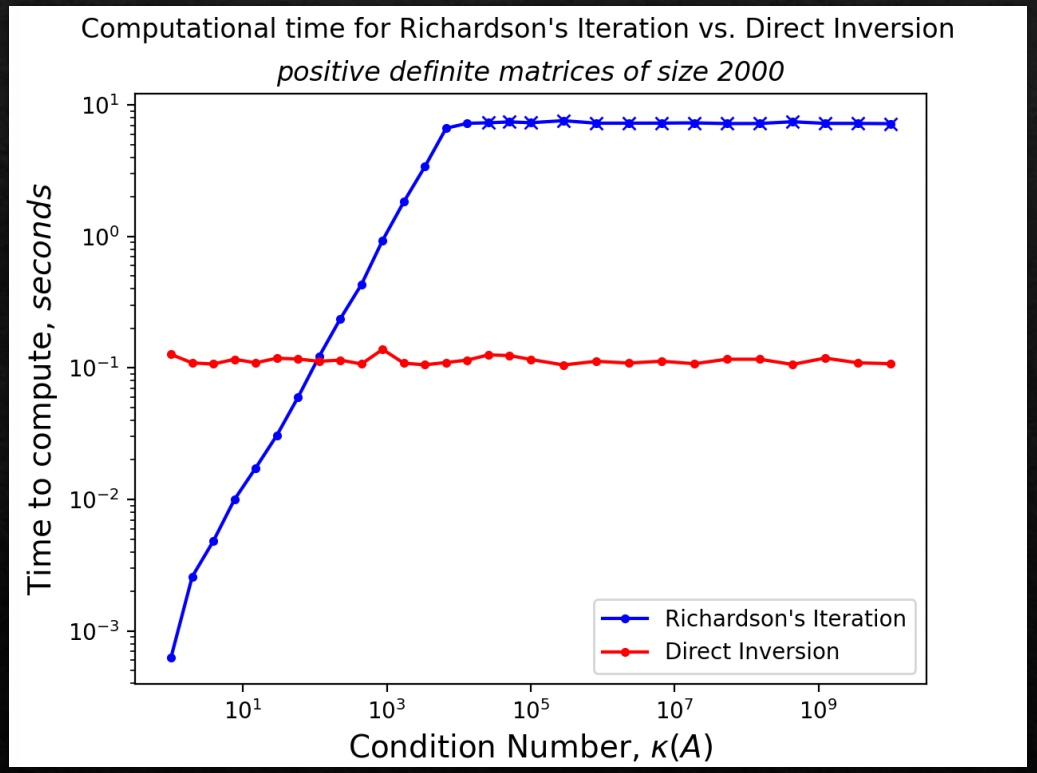
Richardson's Iteration – Size Test

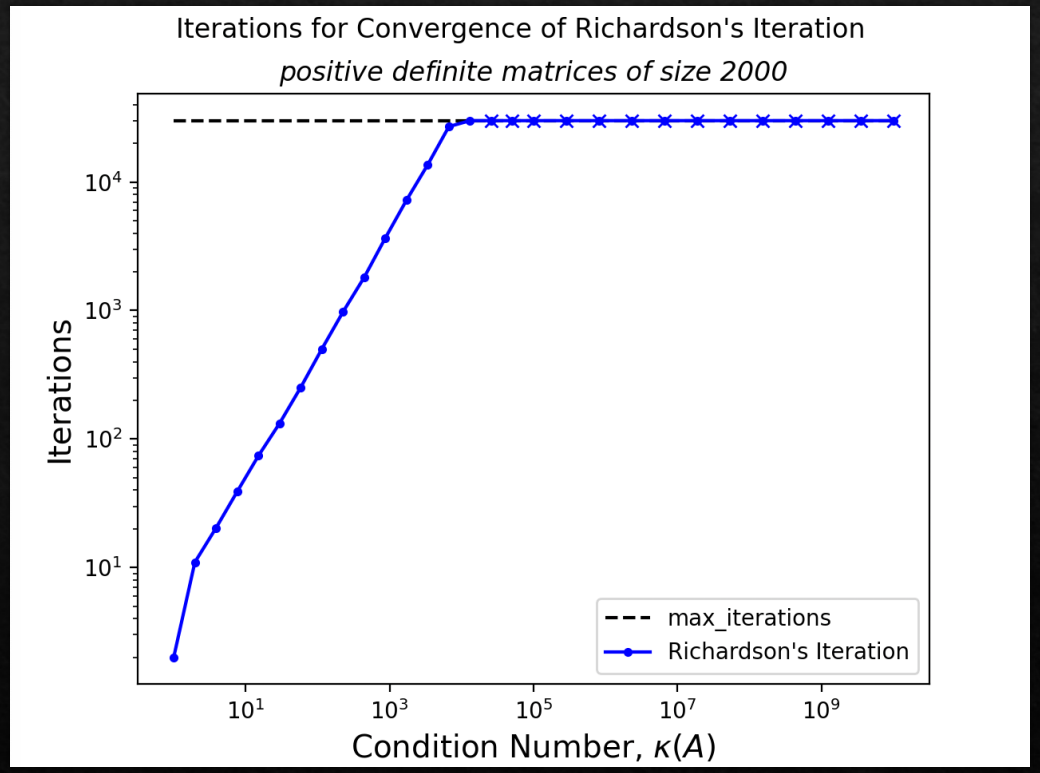




The number of iterations is not directly correlated with n. Thus, Richardson's performance is only bounded by matrix-vector multiplication - a $O(n^2)$ operation – compared to Direct Inversion at $O(n^3)$.
Thus, the larger the matrix, the more the iterative method outperforms the direct solving method.

Richardson's Iteration – Condition Test





Two extremes of performance at $\kappa(A) = 1$ and at $\kappa(A) \sim 10^4$. $\kappa(A) = 1$, 1 iteration for converge of size 2000 with perfect accuracy – reflecting quadratic convergence. $\kappa(A) \sim 10^4$ max iterations of 30000 reached.

GMRES Algorithm

Summary of GMRES-

1) Build Residual using an initial guess
2) Arnoldi iteration on Krylov subspace
3) See if least squares solution satisfies tolerance
4) if not satisfied, repeat with an updated initial guess

Algorithm 2 - GMRES

$\mathbf{x_0} \leftarrow \mathbf{0}$
$\mathbf{r} \leftarrow \mathbf{A x_0} - \mathbf{b}$
$\mathbf{v_1} \leftarrow \frac{\mathbf{r}}{\|\mathbf{r}\|}$
for $k = 0, 1, 2, \ldots$ **do**
  $h_{i,j} = \langle \mathbf{A v}_j, \mathbf{v}_j \rangle$ for i = 1,2,. . . ,j
  $\mathbf{v}_{j+1} = \mathbf{A v}_j - \sum_{i=1}^{j} \mathbf{h}_{i,j} \mathbf{v}_i$
  $h_{j+1,j} = \|\mathbf{v}_{j+1}\|$
  $\mathbf{v}_{j+1} = \frac{\mathbf{v}_{j+1}}{h_{j+1,j}}$
  $\mathbf{x_k} \leftarrow \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k$
  if $\|\mathbf{r}\| <$ tol **return** $\mathbf{x}_k$
end for

GMRES Results

Example Test:

TABLE 2. Comparison of Algorithms: Matrix Scenario 2

Algorithm	GMRES_ours	GMRES_scipy	numpy.linalg.inv
Converged	True	True	-
Time (sec)	0.037	0.001	0.29
$\mathbf{Ax - b}$	7.82×10^{-10}	1.10×10^{-12}	3.19×10^{-14}

Tests Conducted:

1) A matrix with $2{,}000$ distinct well-separated eigenvalues.

2) A full rank matrix with 1 distinct eigenvalue

3) A full rank matrix with 3 distinct eigenvalues

4) A full rank matrix with all eigenvalues in a ball of radius $1 \cdot 10^{-5}$ centered at 1.

5) A matrix whose condition number larger than 10^{20}

6) A matrix with one zero eigenvalue & $\boldsymbol{b} \in range(\boldsymbol{A})$.

TABLE 2. Comparison of Algorithms: Matrix Scenario 2

Algorithm	GMRES_ours	GMRES_scipy	numpy.linalg.inv
Converged	True	True	-
Time (sec)	0.037	0.001	0.29
$\mathbf{Ax - b}$	7.82×10^{-10}	1.10×10^{-12}	3.19×10^{-14}

TABLE 3. Comparison of Algorithms: Matrix Scenario 3

Algorithm	GMRES_ours	GMRES_scipy	numpy.linalg.inv
Converged	True	True	-
Time (sec)	0.01	0.002	0.30
$\mathbf{Ax - b}$	1.255×10^{-13}	3.03×10^{-13}	5.5×10^{-14}

TABLE 4. Comparison of Algorithms: Matrix Scenario 4

Algorithm	GMRES_ours	GMRES_scipy	numpy.linalg.inv
Converged	True	True	-
Time (sec)	2394	0.001	0.26
$\mathbf{Ax - b}$	3.77×10^{-12}	9.43×10^{-5}	7.82×10^{-14}

GMRES Results: Ideal Cases

2) A full rank matrix with 1 distinct eigenvalue

3) A full rank matrix with 3 distinct eigenvalues

4) A full rank matrix with all eigenvalues in a ball of radius $1 \cdot 10^{-5}$ centered at 1.

TABLE 1. Comparison of Algorithms: Matrix Scenario 1

Algorithm	GMRES_ours	GMRES_scipy	numpy.linalg.inv
Converged	False	False	-
Time (sec)	7339	132	0.28
$\mathbf{Ax - b}$	4.8×10^{-1}	6.71×10^{-2}	9.93×10^{-5}

TABLE 5. Comparison of Algorithms: Matrix Scenario 5

Algorithm	GMRES_ours	GMRES_scipy	numpy.linalg.inv
Converged	False	False	-
Time (sec)	6863	132	0.30
$\mathbf{Ax - b}$	1.47×10^{1}	1.90×10^{1}	1.44×10^{3}

GMRES Results: Difficult Cases

1) A matrix with $2{,}000$ distinct well-separated eigenvalues. (large spectrum)

5) A matrix whose condition number larger than 10^{20} (ill-conditioned)


It's eigenvalue time

Why Iterative Solvers for Eigenvalues

- Finding the eigenvalues is one of the most important problems in numerical analysis.
- No scalable direct-solving algorithm for eigenvalue computation exists.
- Thus, iterative solvers for finding eigenvalues are indispensable for larger matrices.

Power Method

Recall for $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\boldsymbol{b} \in \mathbb{R}^n$:

$$\boldsymbol{b} = c_1 \boldsymbol{v}_1 + c_2 \boldsymbol{v}_2 + \cdots + c_n \boldsymbol{v}_n$$

Thus, given λ_n is distinct,

$$\boldsymbol{A}^k \boldsymbol{b} = \lambda_n^k \left(c_1 \left(\frac{\lambda_1}{\lambda_n} \right)^k \boldsymbol{v}_1 + c_2 \left(\frac{\lambda_2}{\lambda_n} \right)^k \boldsymbol{v}_2 + \cdots + c_{n-1} \left(\frac{\lambda_{n-1}}{\lambda_n} \right)^k \boldsymbol{v}_{n-1} + c_n \boldsymbol{v}_n \right) \sim c_n \lambda_n^k \boldsymbol{v}_n$$

Define $\boldsymbol{x}_k = \frac{\boldsymbol{A}^k \boldsymbol{b}}{\|\boldsymbol{A}^k \boldsymbol{b}\|}$. As k grows, $\boldsymbol{x}_k \to \boldsymbol{v}_n$.

To get λ_n, we use Rayleigh Ritz Quotient $\to \lambda_n = \frac{\boldsymbol{v}_n^T \boldsymbol{A} \boldsymbol{v}_n}{\boldsymbol{v}_n^T \boldsymbol{v}_n}$

Algorithm 3 - Power Method

$\mathbf{x_0} \leftarrow$ arbitrary vector in $\mathbb{R}^n$
for $k = 1, 2, \ldots, n$ **do**
  $\mathbf{b} \leftarrow \mathbf{A}\mathbf{x}_{k-1}$
  $\mathbf{x_k} \leftarrow \frac{1}{\|\mathbf{b}\|} \cdot \mathbf{b}$
end for
$\lambda \leftarrow (\mathbf{x}_n)^\mathsf{T} \mathbf{A} (\mathbf{x}_n)$

Power Method - Extensions

Inverse Power Method - Power Method on A^{-1}. Converge to $\frac{1}{\lambda_1}$

Notes:

- If λ is an eigenvalue of A, then $\lambda - c$ is an eigenvalue of $A - cI$.
- Minimum eigenvalue of $A - cI$ is $\min_{i=1,\ldots,n} |\lambda_i - c|$

Algorithm 4 - Rayleigh-Ritz Method

$\mathbf{x_0} \leftarrow$ arbitrary normal vector in $\mathbb{R}^n$
$\lambda_0 \leftarrow (\mathbf{x}_0)^{\mathsf{T}}\mathbf{A}(\mathbf{x}_0)$
for $k = 1, 2, \ldots, n$ **do**
  $\mathbf{b} \leftarrow (\mathbf{A} - \lambda_{k-1}\mathbf{I})^{-1}\mathbf{x}_{k-1}$
  $\mathbf{x_k} \leftarrow \frac{1}{\|\mathbf{b}\|} \cdot \mathbf{b}$
  $\lambda_k \leftarrow (\mathbf{x}_k)^{\mathsf{T}}\mathbf{A}(\mathbf{x}_k)$
end for

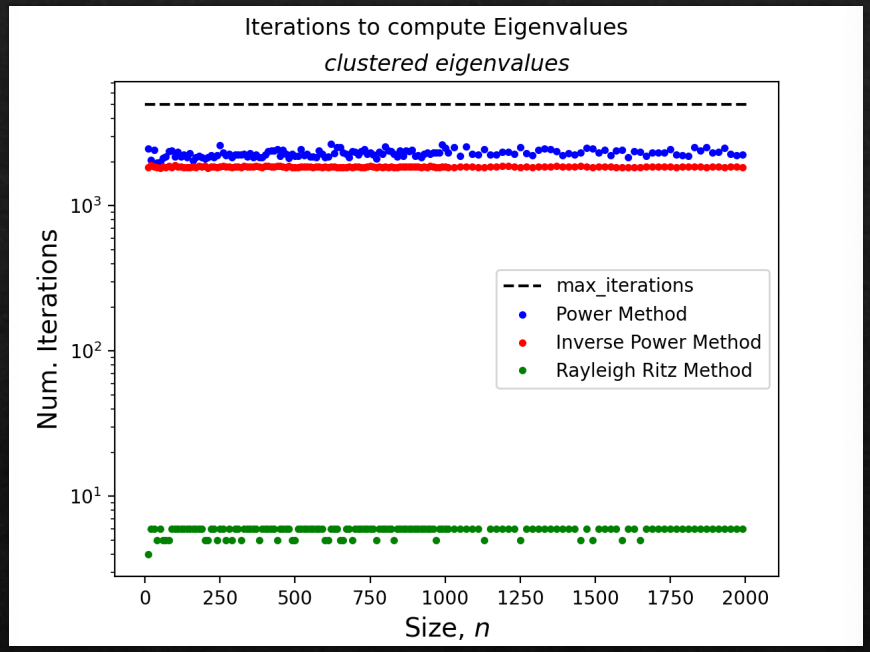
Rayleigh-Ritz Iteration – Inverse Power Method on $A - c_k I$ with a shift c_k at each iteration

- Use Rayleigh-Ritz Quotient to choose shift $c_k = \lambda_k = \frac{x_k^T A x_k}{x_k^T x_k}$
- Improves rate of convergence drastically: linear → cubic

Performance – Power Methods

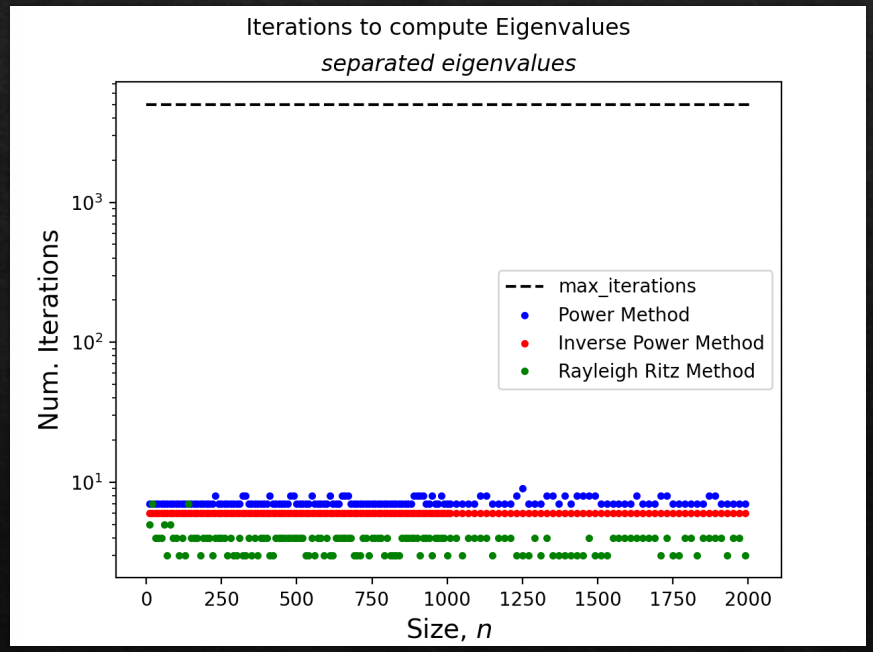
Clustered Eigenvalues - Iterations

$|\lambda| = (99, 100, \ldots, 100, 101)$



Separated Eigenvalues - Iterations

$|\lambda| = (1, 50, \ldots, 50, 2500)$

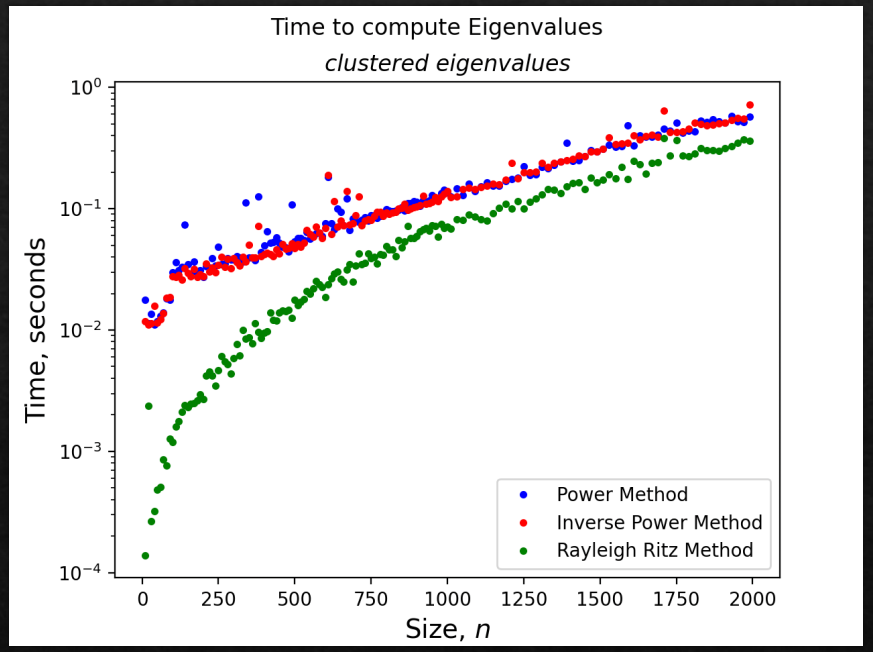


Rayleigh Ritz shift speeds up converge tremendously.

Performance – Power Methods

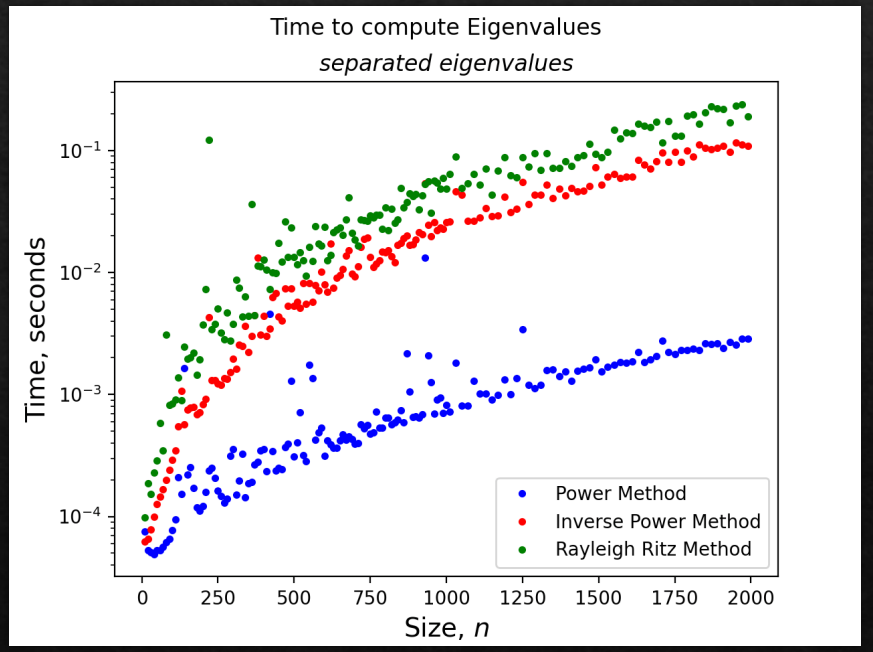
Clustered Eigenvalues - Time

$|\lambda| = (99, 100, \ldots, 100, 101)$



Separated Eigenvalues - Time

$|\lambda| = (50, 100, \ldots, 100, 2500)$



Simplicity at each iteration of Power Method means it runs much faster for similar # of iterations

QR Iteration

"Pure QR" – converts A into an upper-triangular matrix with QR decomposition at each iteration. Diagonal of this matrix is the eigenvalues. Requires all eigenvalues are distinct.

$$\mathbf{Q}_k\mathbf{R}_k = \mathbf{X}_{k-1}$$
$$\mathbf{X}_k = \mathbf{R}_k\mathbf{Q}_k$$

This iteration can be shifted in the same fashion as the Power Method

$$\mathbf{Q}_k\mathbf{R}_k = \mathbf{X}_{k-1} - \mu\mathbf{I}$$
$$\mathbf{X}_k = \mathbf{R}_k\mathbf{Q}_k + \mu\mathbf{I}$$

Before chosen shift was current guess for eigenvalue – for QR: lowest right corner of the matrix.

QR Iteration - Methods

Offer 3 methods for tests

- Pure QR – as described previous
- Lazy Shift (LS) –
  - choose $\mu = a_{n,n}$, the bottom right element of $\boldsymbol{A}$
- Complex Shift (CS) –
  - choose $\mu_k = x_{n,n}^k$, the bottom right element of $\boldsymbol{X}_k$
  - When this eigenvalue is found, "deflate" the matrix

Algorithm 6 - Complex Shift QR Algorithm

$\mathbf{X}_0 \leftarrow \mathbf{A}$
$k = 0$
for $m = n, n-1, \ldots, 1$ **do**
  while $x_{m,m}^{(k)}$ not converged **do** // while eigenvalue not found
    $k = k + 1$
    $\mu_k = x_{m,m}^{(k)}$ // entry of the last row and column of $\mathbf{X}_k$
    $\mathbf{Q}_k\mathbf{R}_k \leftarrow \mathbf{X}_{k-1} - \mu\mathbf{I}$ // perform QR factorization with shift
    $\mathbf{X}_k \leftarrow \mathbf{R}_k\mathbf{Q}_k + \mu\mathbf{I}$
  end while
  $\Lambda_m = x_{m,m}^{(k)}$ // set bottom right entry as m-th eigenvalue
  $\mathbf{X}_k \leftarrow \mathbf{X}_k[: m-1, : m-1]$ // deflate matrix by removing last column and row
end for

Will also compare with numpy.linalg.eig() method.

QR Iteration Test - Symmetric Matrices

Size	PQ Iter.	PQ Time (s)	LS Iter.	LS Time (s)	CS Iter.	CS Time (s)	eig() time (s)
10	415.65	8.446e-3	120.85	2.441e-3	17.6	3.61e-4	1.55e-04
25	3549.7	0.11252	1116.35	0.034519	45.4	1.174e-3	1.73e-4
50	9805.4	0.827592	4815.2	0.333424	88.85	3.993e-3	5.87e-4
100	*17667.2*	*11.558*	*10969.8*	*3.789*	174.2	0.0365	3.51e-3
250	-	-	*18701.4*	*56.866*	395.2	0.531	0.0316
500	-	-	-	-	778.4	4.575	0.130
1000	-	-	-	-	1510.667	51.19	0.843

Note: italics indicate only some simulations converged. '–' indicates none converged

Each method performs slightly better. Still, all are outperformed drastically by numpy.linalg.eig()

Eigenvalue Iterations in Practice

- PageRank
- Aerospace

Power Method In Practice: Google's PageRank

Summary of Application to PageRank

- Provides an approximation of a webpage's importance by ranking how many times it is referenced by other pages
- Calculated by determining the principal eigenvector of a normalized link matrix
- Does not require actually crawling each web-page or require having text to search because it counts references to a sight
- Since only the principal (first) eigenvector is required, iterative methods are ideal

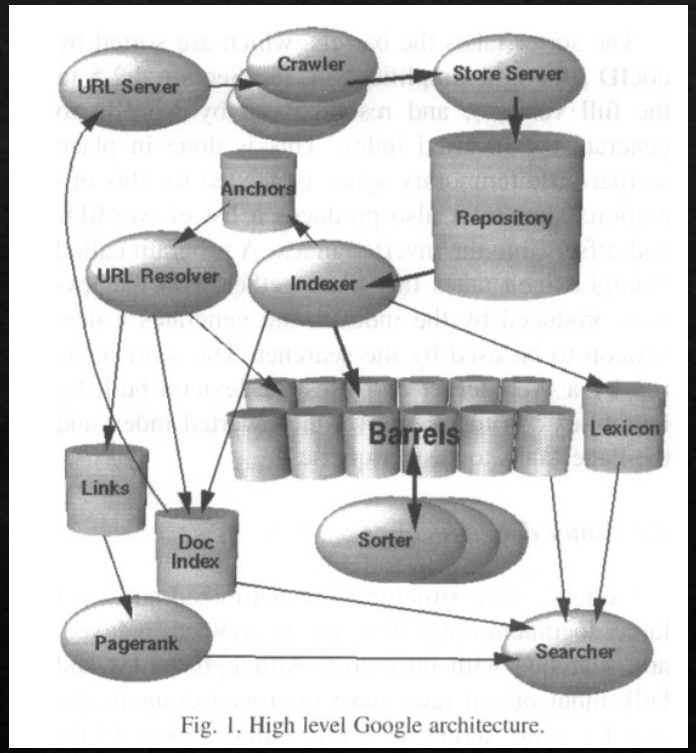


Fig. 1. High level Google architecture.

Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." Computer networks and ISDN systems 30.1-7 (1998): 107-117.

QR Iteration In Practice: Aerospace

Summary of Application to Aerospace

- Principal eigenvectors determine stability of a system
- Iterative methods can be used to quickly determine the leading eigenvector to show long term system behavior
- A more negative (larger magnitude < 0) number corresponds to a more stable configuration of the airframe
- Can use tune parameters based on eigen structure to improve or modify performance of aircraft

TABLE 3: Lateral eigenstructure assignment design[a].

	Feedback gain matrix				Achievable eigenvectors		Achievable eigenvalues
	β	p_s	$(r_s)_{wo}$		Dutch roll	Roll subsidence	
Unweighted B $\zeta_{dr} = 0.4$	−1.7256	0.2945	0.2292	elevonL	$1 \pm j0$	0	$(\zeta_{dr}, \omega_{dr}) = (0.4, 3.0)$
	1.7256	−0.2945	−0.2292	elevonR	$0 \pm j0$	−0.2531	$\lambda_{roll} = -4$
	0.7870	0.1188	−0.2387	amtL	$0 \pm j0.0001$	1	$\lambda_{spiral} = 0.0077$
	−0.7870	−0.1188	0.2387	amtR	$1.1007 \pm j2.8348$	−0.0106	$\lambda_{filter} = -1.33$
	7.7741	−0.4185	−1.5142	yawTV	$-1.0545 \pm j0.3236$	0.0035	
Weighted B $W = (1, 1, 1, 1, 0.25)$ $\zeta_{dr} = 0.4$	−8.0478	0.6333	1.4730	elevonL	$1 \pm j0$	0	$(\zeta_{dr}, \omega_{dr}) = (0.4, 3.0)$
	8.0478	−0.6333	−1.4730	elevonR	$0 \pm j0$	−0.2531	$\lambda_{roll} = -4$
	8.6966	−0.3052	−1.7949	amtL	$0 \pm j0.0001$	1	$\lambda_{spiral} = 0.0077$
	−8.6966	0.3052	1.7949	amtR	$1.1166 \pm j2.8222$	−0.0120	$\lambda_{filter} = -1.34$
	3.3934	−0.1820	−0.6667	yawTV	$-1.0504 \pm j0.3297$	0.0040	
Weighted B $W = (1, 1, 1, 1, 0.25)$ $\zeta_{dr} = 0.707$	−10.5499	0.6535	2.7356	elevonL	$1 \pm j0$	0	$(\zeta_{dr}, \omega_{dr}) = (0.707, 3.0)$
	10.5499	−0.6535	−2.7356	elevonR	$0 \pm j0$	−0.2531	$\lambda_{roll} = -4$
	11.8373	−0.3307	−3.3928	amtL	$0 \pm j0$	1	$\lambda_{spiral} = 0.0077$
	−11.8373	0.3307	3.3928	amtR	$2.1068 \pm j2.2247$	−0.0120	$\lambda_{filter} = -1.84$
	4.5464	−0.1913	−1.2514	yawTV	$-1.2299 \pm j0.3432$	0.004	

[a]Design model includes washout filter but no actuator dynamics.

Nieto-Wire, Clara, and Kenneth Sobel. "Flight control design for a tailless aircraft using eigenstructure assignment." International Journal of Aerospace Engineering 2011 (2011).

Conclusion

In this presentation, we have considered potentials and limitations of various iterative solvers

Linear Systems – Richardson's Iteration, GMRES

Eigenvalue Problem – Power Method & QR Iteration, Real-world Application

"Iterative solvers not only enhance the efficiency of numerical solutions but provide a foundational framework for breakthroughs in applications of numerical computation."

-Gus and Kevin

Conclusion

In this presentation, we have considered potentials and limitations of various iterative solvers

Linear Systems – Richardson's Iteration, GMRES

Eigenvalue Problem – Power Method & QR Iteration, Real-world Application

"Iterative solvers not only enhance the efficiency of numerical solutions but provide a foundational framework for breakthroughs in applications of numerical computation."

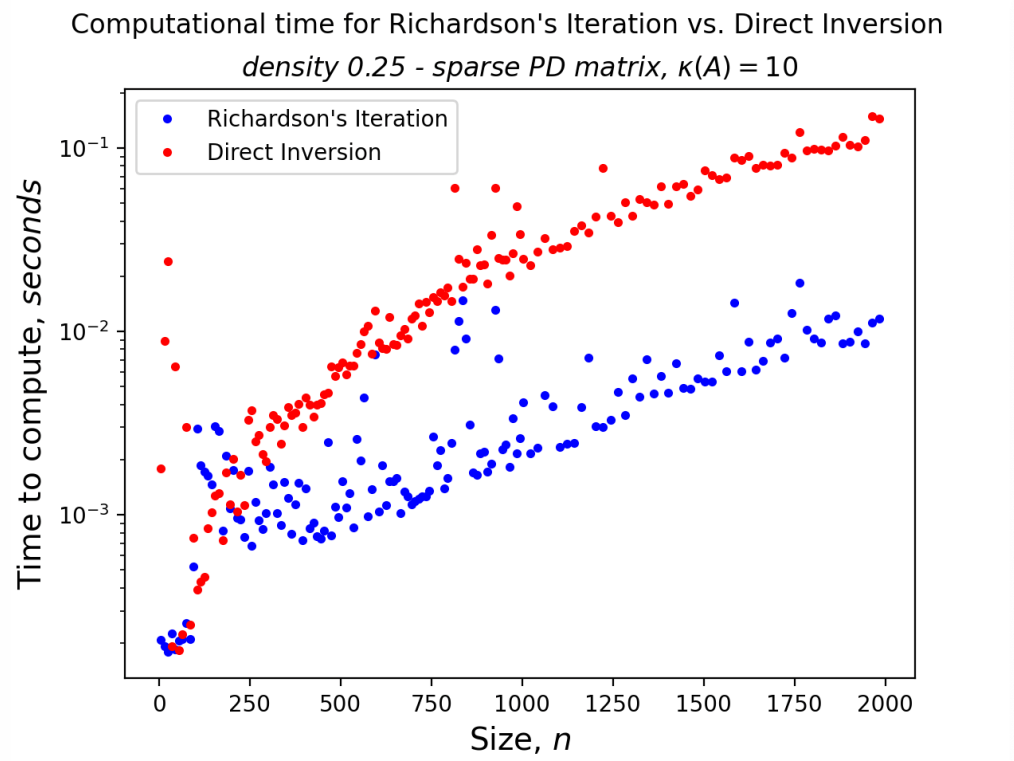
Thank you!

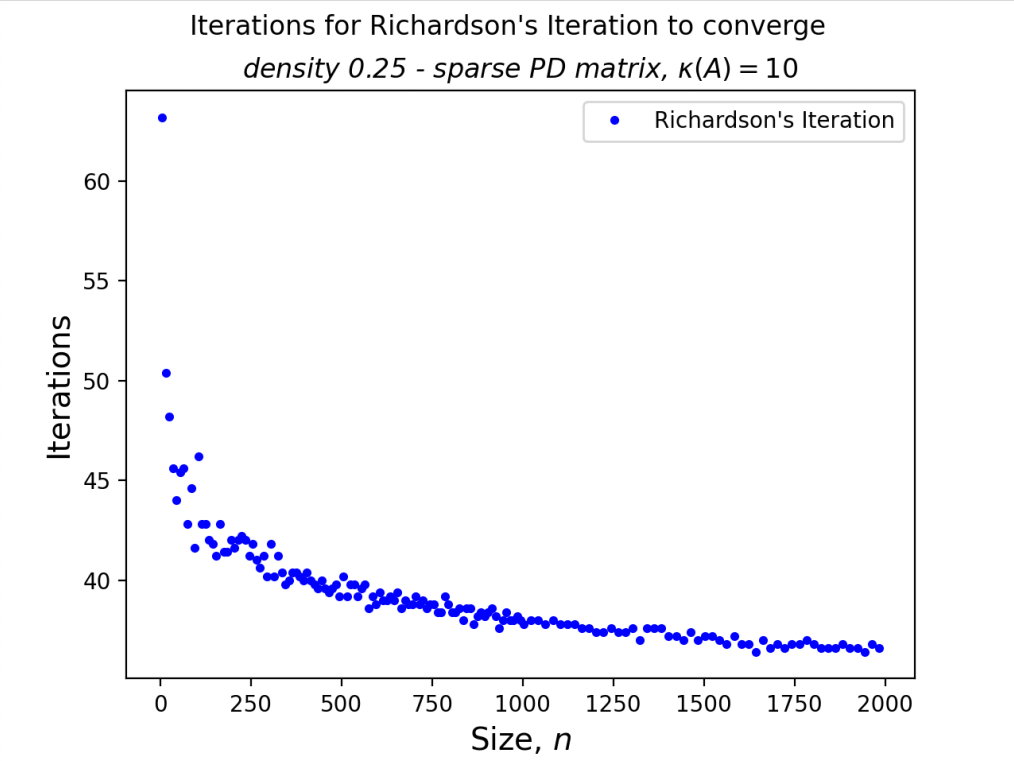
Any Questions?

Extra Results

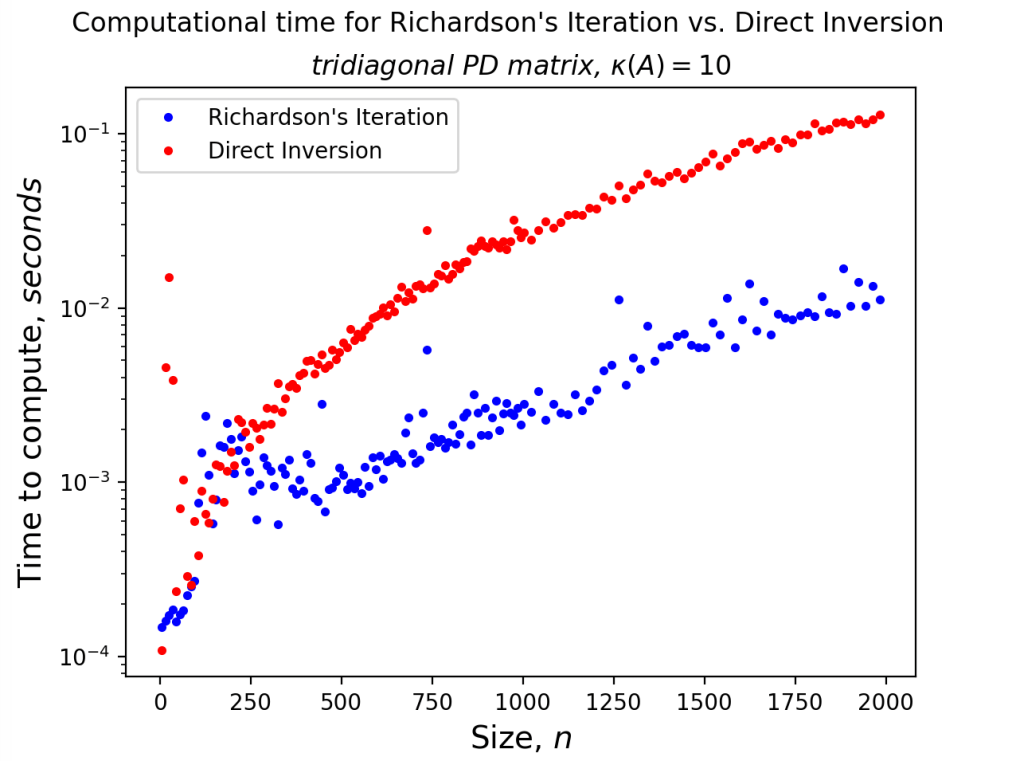
If needed

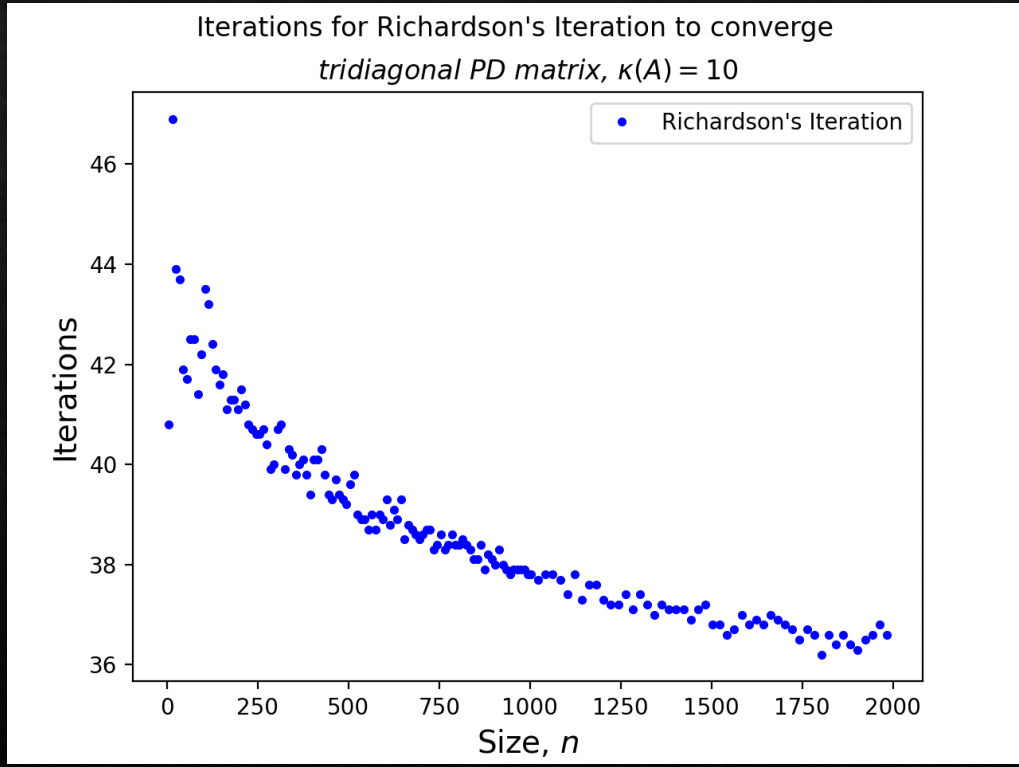
Richardson's Iteration – Random Sparse





Richardson's Iteration – Tridiagonal Sparse





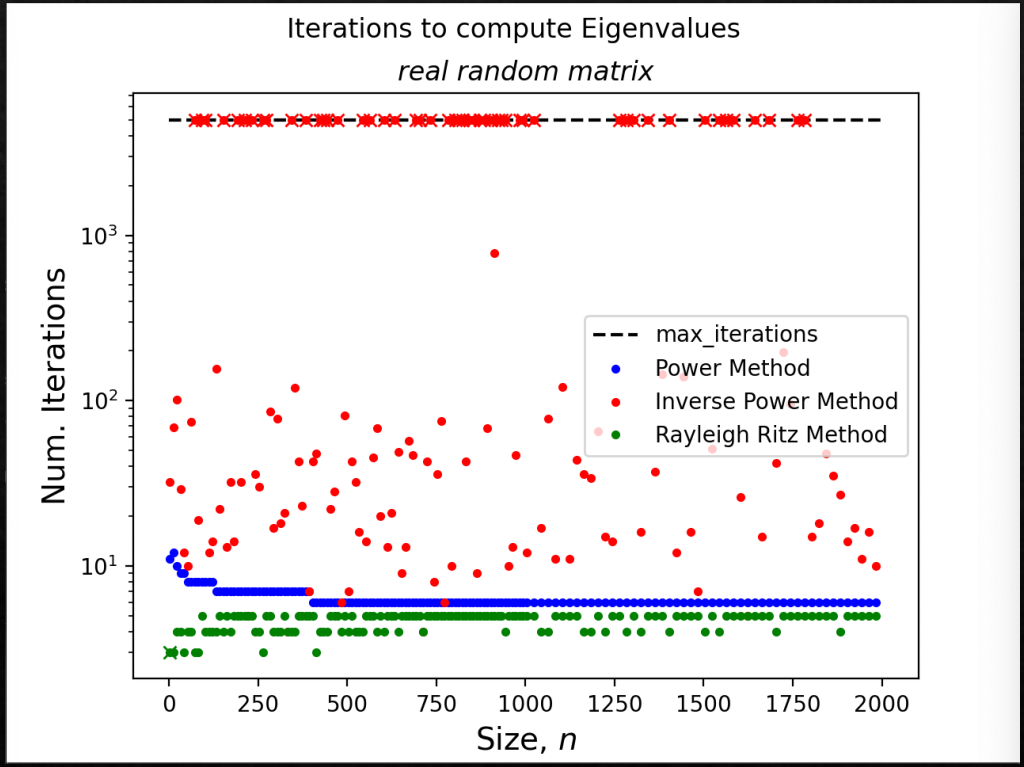
GMRES Results: Interesting Case

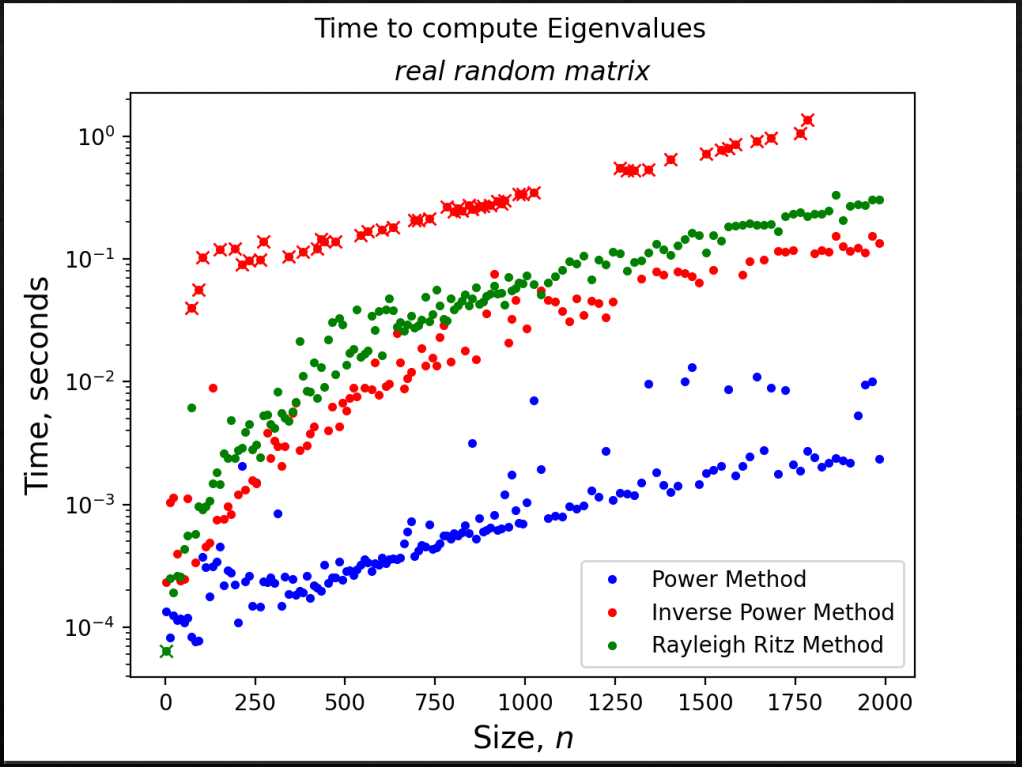
TABLE 6. Comparison of Algorithms: Matrix Scenario 6

Algorithm	GMRES_ours	GMRES_scipy	numpy.linalg.inv
Converged	True	True	-
Time (sec)	26	47	0.29
$\mathbf{Ax - b}$	5.5×10^{-16}	1.17×10^{-2}	6.13×10^{4}

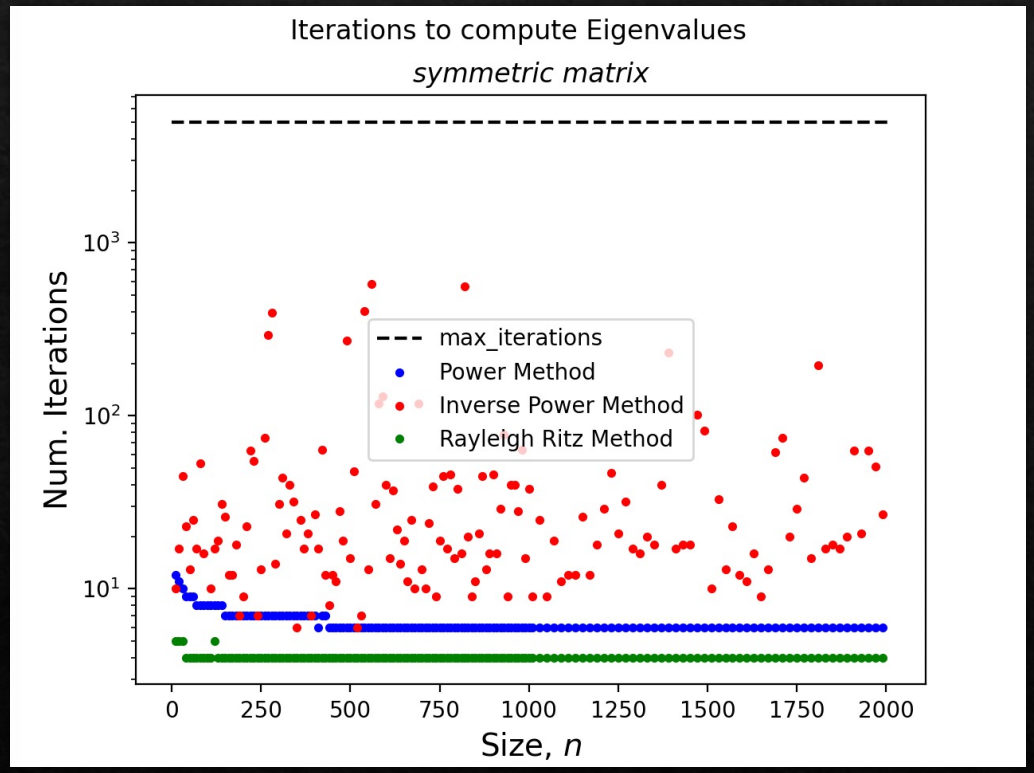
6) A matrix with one zero eigenvalue & $\boldsymbol{b} \in range(\boldsymbol{A})$.

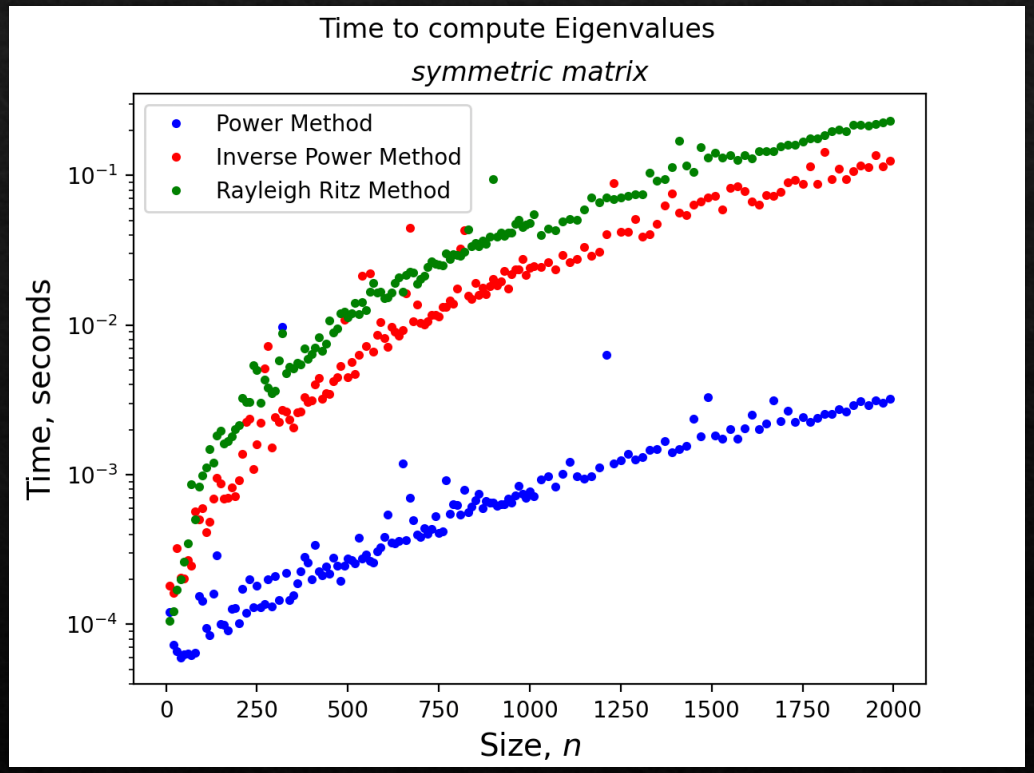
Power Methods – Random





Power Methods – Symmetric

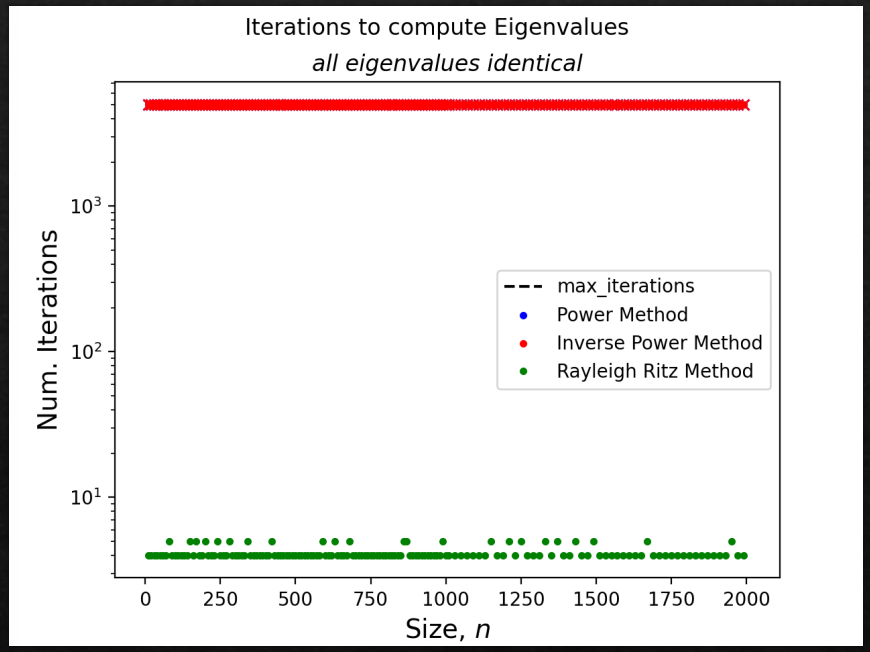




Performance – Power Methods

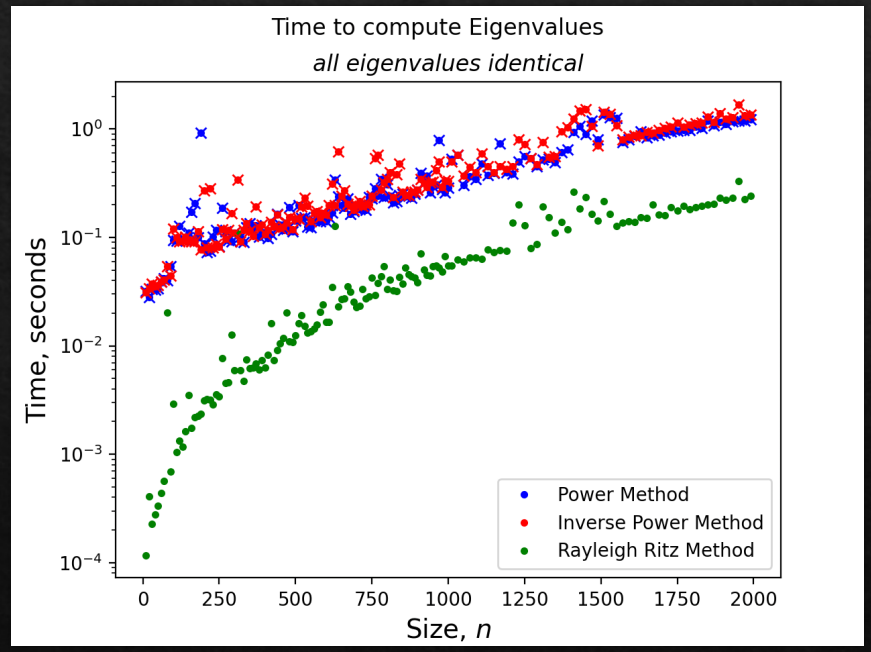
Identical Eigenvalues - Iterations

$$|\lambda| = (n, n, \ldots, n, n)$$



Identical Eigenvalues - Time

$$|\lambda| = (n, n, \ldots, n, n)$$



Power and Inverse Method required unique dominant eigenvalue.
Rayleigh-Ritz shift removes this requirement.

QR Iteration Test - Random Matrices

Size	SI Iter.	LS Iter.	CS Iter.	SI Time	LS Time	CS Time	eig() time
2	11.55	28.9	3.85	2.465e-04	4.605e-04	7.365e-05	2.640e-05
5	*17022.15*	*17001.6*	*15805.8*	*3.07e-01*	*2.494e-01*	*2.471e-01*	1.195e-04
10	-	-	-	-	-	-	3.644e-04

Note: italics indicate only some simulations converged. – indicates none converged

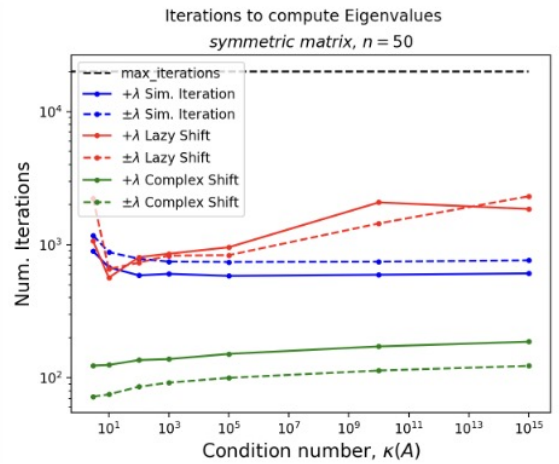
Eigenvalues for random matrices are not all real. Our iterations converge to all real eigenvalues of the matrices, but fail to discover the complex ones

QR Iteration Test - Duplicate Eigenvalues

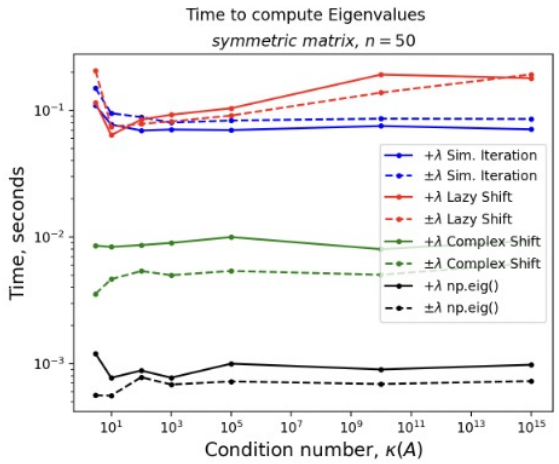
Type	SI Iter.	LS Iter.	CS Iter.	SI Time	LS Time	CS Time	eig() time
1 distinct ($+\lambda$)	2	2	1	2.717e-4	2.519e-4	1.596e-4	2.86e-4
3 distinct ($+\lambda$)	43.25	36.85	7.3	5.81367e-3	4.70212e-3	9.32e-4	3.6832e-4
1 distinct ($\pm\lambda$)	20000	20000	6.05	1.733	1.378	5.214e-04	2.934e-04
3 distinct ($\pm\lambda$)	20000	20000	14.85	1.733	1.376	9.878e-04	4.018e-04

QR Iteration

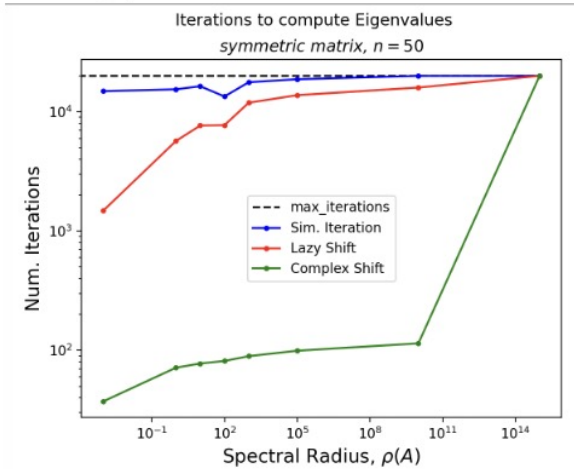
Condition Number + Spectral Radius



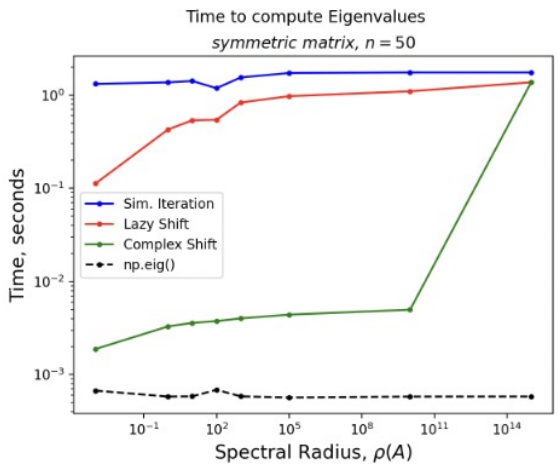
(A) Condition Number - Iterations



(B) Condition Number - Time



(C) Spectral Radius - Iterations



(D) Spectral Radius - Time

Iterative Solvers