ANLA ASSIGNMENT -6

Monday, 9. January 2023 20:45

NAME: ABHIK SARKAR

MATRIKELNUMMER: 23149662

IDM Id : lo88xide

EMAIL: abhik.sarkar@fau.de / abhik.sarkar.718@gmail.com

ANLA ASSIGNMENT - 6

Exercise 1 : Tridiagonal Matrices :

$A \in \mathbb{C}^{m \times m}$ be tridiagonal and hermitian, with all it's sub and superdiagonal enteries non-zero.

To prove : A has distinct m- eigenvalues.

Proof : matrix $A = \begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ c_1 & a_2 & b_2 & \ddots & 0 \\ 0 & c_2 & \ddots & \ddots & b_{m-1} \\ \vdots & & \ddots & c_{m-1} & a_m \end{bmatrix}$

Now, let us assume λ is the real eigen value of A.

$$A - \lambda I = \begin{bmatrix} a_1-\lambda & b_1 & 0 & \cdots & 0 \\ c_1 & a_2-\lambda & b_2 & \ddots & 0 \\ 0 & c_2 & \ddots & \ddots & b_{m-1} \\ \vdots & \ddots & & & \\ 0 & \cdots & 0 & c_{m-1} & a_m-\lambda \end{bmatrix}$$

Now, if we delete the row '1' and column 'm' of the matrix $A \in \mathbb{C}^{m \times m}$, we get a new matrix $A_{m-1} \in \mathbb{C}^{(m-1)\times(m-1)}$

New matrix, $(A-\lambda I)_{m-1} = \begin{bmatrix} c_1 & a_2-\lambda & b_1 & 0 & \cdots & 0 \\ 0 & c_2 & \ddots & b_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & b_{m-2} \\ \vdots & & \ddots & \ddots & \ddots & a_{m-1}-\lambda \\ 0 & \cdots & \cdots & \cdots & 0 & c_{m-1} \end{bmatrix}_{m-1 \times m-1}$

- The above matrix formed is an upper triangular matrix such the it's all diagonal entries $c_1, c_2 \cdots c_{m-1}$ are non-zero elements. (Given in question)
- So, as the determinant of an upper triangular matrix is the product of the diagonal elements of the matrix so, the determinant of the above matrix cannot be zero.

$$|(A-\lambda I)_{m-1}| = c_1 \cdot c_2 \cdot \cdots \cdot c_{m-1} \neq 0.$$

- $\therefore A_{m-1}$ is a non-singular matrix.
- $\therefore$ dim (Kernel$(A-\lambda I)_{m-1}$) = Number of free variables = 1 = Geometric multiplicity of the matrix

$\rightarrow \lambda \in$ Kernel $(A-\lambda I)_{m-1}$.

Now, since the matrix was given as hermitian, so, it is also diagonalizable. Therefore, the algebraic multiplicity is equal to the geometric multiplicity as 1.

$\because$ Algebraic multiplicity is equal to 1, so, we can state that the matrix has distinct eigen values.

(b) A is upper Hessenberg matrix :

$a_{ij} = 0 \quad \forall \quad i > j+1.$

As the sub-diagonal entries are non-zero, so the matrix A, $a_{i+1,i} \neq 0 \quad \forall \; i \in \{1, \ldots, m-1\}$ is an upper unreduced matrix.

If we assume,

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

then, it is following both the conditions -

$a_{ij} = 0 \quad \forall \quad i > j+1.$, $a_{31} = 0.$

$a_{ij} \neq 0 \quad \forall \quad j = i-1$, $a_{32} = -1$, $a_{21} = 1.$

$a_{ij} \neq 0 \quad \forall \quad j > i$, $a_{12} = 1$, $a_{13} = 1$, $a_{23} = 1.$

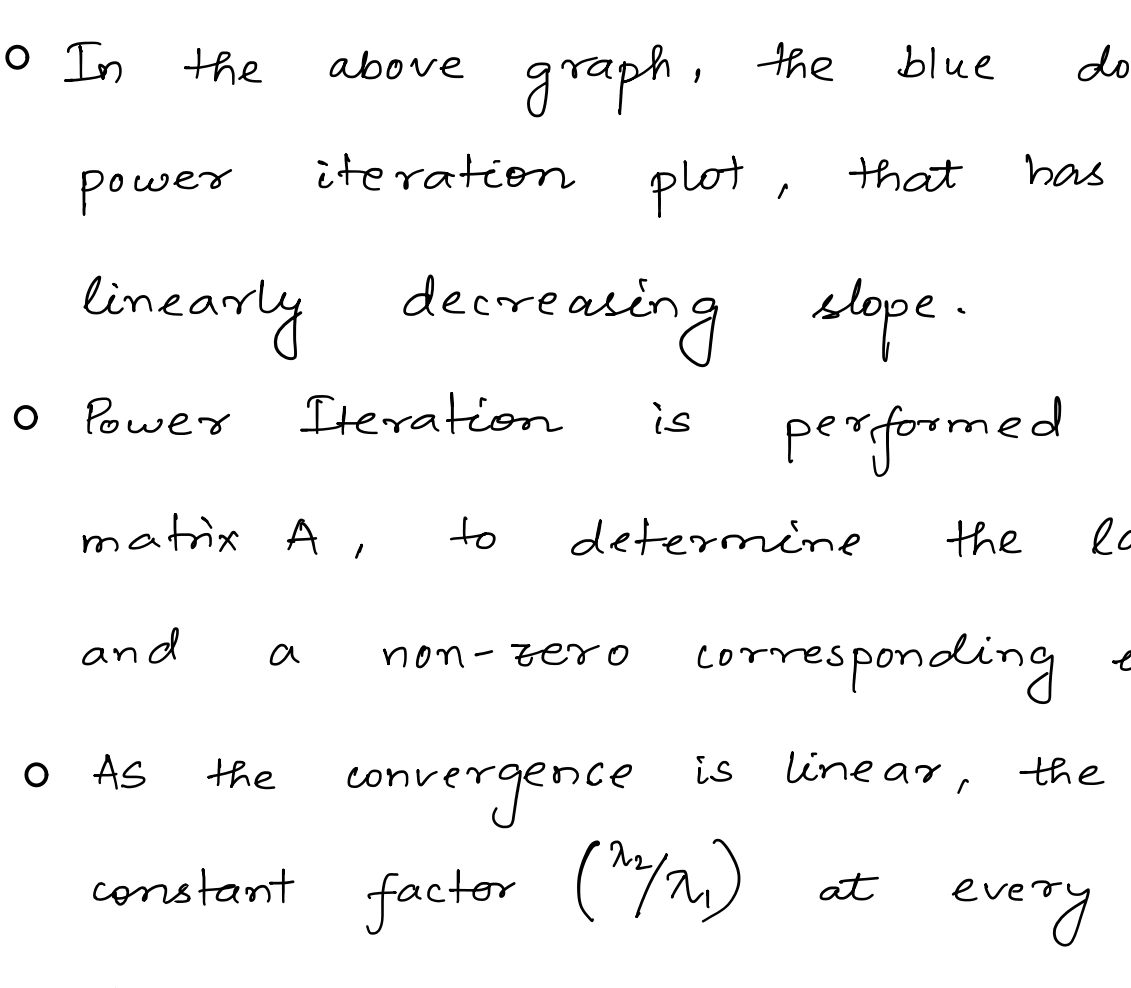
But, here the three eigen values are

$\lambda_1 = 1$, $\lambda_2 = 0$, $\lambda_3 = 0.$

$\therefore$ Eigenvalues of matrix A is not distinct.

Exercise 2 :

(f) convergence comparision



→ Power Iteration Method :

- In the above graph, the blue dotted line illustrates the power iteration plot, that has a convergence with a linearly decreasing slope.
- Power Iteration is performed on a given diagonalizable matrix A, to determine the largest absolute eigen value λ and a non-zero corresponding eigen vector v.
- As the convergence is linear, the error is reduced by a constant factor (λ_2/λ_1) at every iteration. for minimizing the error, the efficiency of the factor depends upon the largest eigenvalue which should be significantly larger than the other eigenvalues.
- Therefore, the algorithm converges very slowly. However, if the largest two eigenvalues are very close in magnitude, the convergence takes a lot of time and becomes really slow.
- Hence, we can apply this algorithm effectively for a very large sparse matrix with proper implementation.

→ Inverse Iteration Method (Random μ).

- This method is used to estimate the approximate eigen vector, by taking an approximation to the corresponding eigen value.
- we take an approximate eigen value $\mu \in \mathbb{R}$, which is not an absolute eigenvalue of A.

Hence, we assume eigen value of A as λ, which results the eigen values of $(A-\mu I)^{-1}$ as $(\lambda - \mu I)^{-1}$.

- Therefore, for μ being very closer to the eigen value of A, the algorithm will converge very fast and rapidly as the largest eigen value of $(A-\mu I)^{-1}$ will be much larger than the rest.
- likewise Power Iteration, inverse iteration exhibits only linear convergence. This is shown by the orange dotted line in the above graph having the convergence as a linearly decreasing slope.
- As we calculate the eigen vector v, by supplying an estimated μ of the corresponding eigen value, so the rate of linear convergence can be controlled depending upon the value of μ.
- Therefore, this method is used when a good approximation of the eigen value is known. An incorrect choice of μ might also lead to very slow convergence.

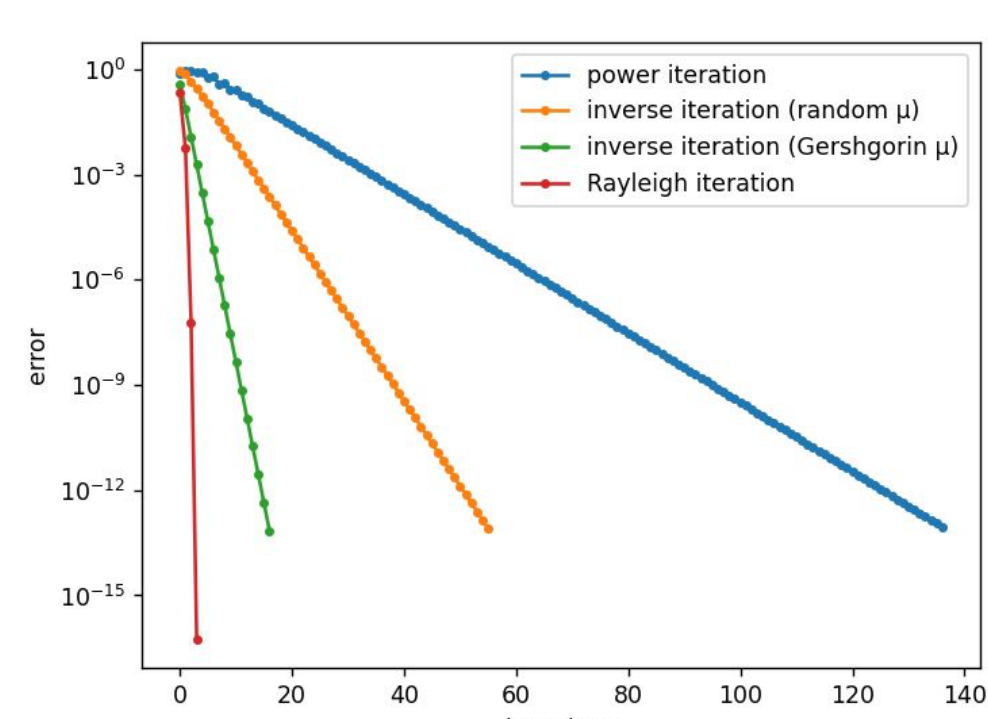
→ Inverse Iteration Method (Gershgorin μ) :

- In the above graph, this iteration is shown in green dotted line that is converging linearly as a decreasing slope.
- Here the μ is estimated from the gershgorin circle theorem which directly returns λ_{max} in it's algorithm. As, this returns the best possible value for $\mu = \lambda_{max}$, so the graph is taking less number of iterations to find the largest eigen value.

→ Rayleigh Quotient Iteration Method.

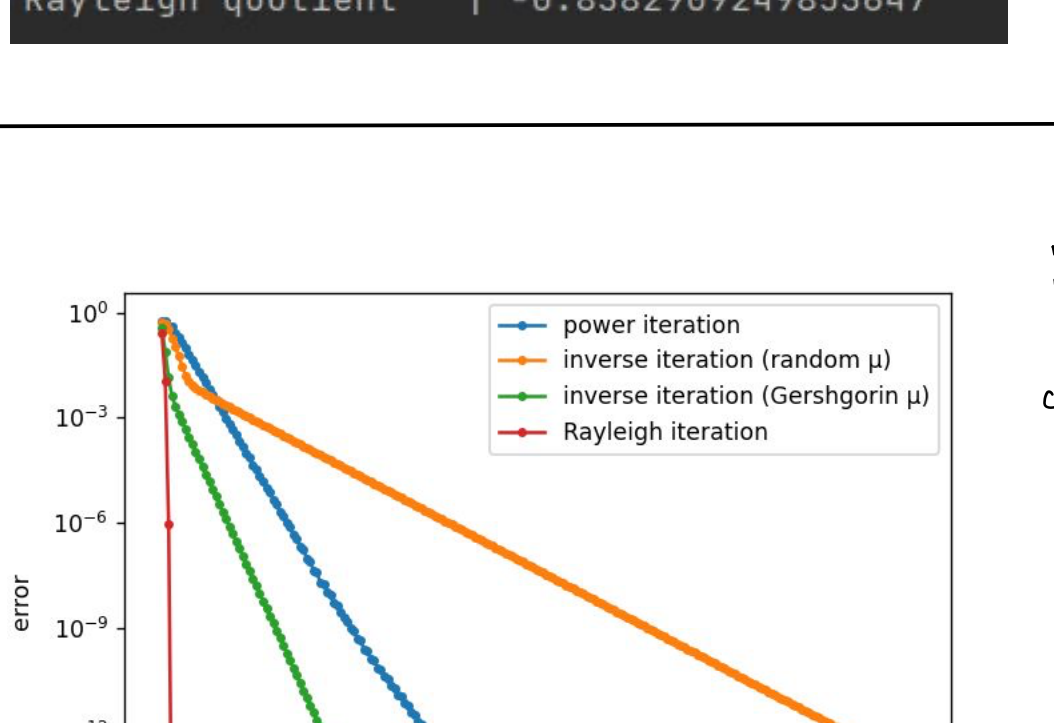
- This algorithm is similar to inverse iteration, but replaces the estimated eigen value at the end of every iteration.
- That is, it is continually improving the eigen value estimates to increase the rate of convergence of inverse iterations after each and every step.
- Therefore, the algorithm converges at a very faster rate about each iteration tripling the amount of accuracy for calculating λ. This leads to less number of iteration as compared to power & inverse method.
- The Rayleigh quotient iteration Method line depicted in red colour converges cubically for Hermitian Matrices, given the initial vector very close to the calculated eigen vector of the matrix. There after, this algorithm produces the most accurate eigen value estimates.

(f) Below the following graphs with different matrices :




```
A =[[ 0.59707624 -0.80857599]
 [-0.80857599 -0.3828011 ]]

v =[0.02500417 0.99968735]

μ =[3.58010157]

algorithm            |   λ
-------------------------------------------
power                | 1.0525660614229781
inverse (random μ)   | 1.0525660614229781
inverse (Gershgorin)| 1.052566061422978
Rayleigh quotient    | -0.8382909249853647
```

Here, in this graph we have taken a new matrix such that the power iteration method is taking the most number of iterations, and the Rayleigh iteration method is taking about 4-5 iterations in total to find the accurate eigen value. This shows the efficiency of the rate of convergence of the Rayleigh Iteration.

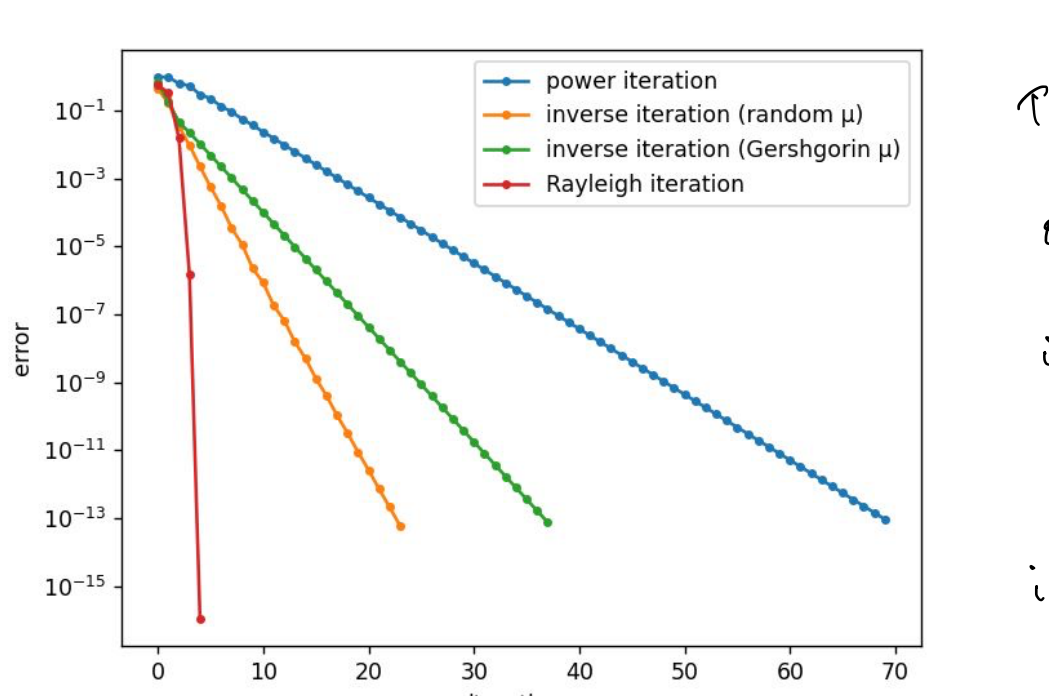



```
A =[[-0.37807511  0.24056876  0.30142707]
 [ 0.24056876  0.50745632 -0.09631941]
 [ 0.30142707 -0.09631941  0.68013434]]

v =[-0.67644236  0.19046531  0.71144129]

μ =[2.36153231]

algorithm            |   λ
-------------------------------------------
power                | 0.7648568091385108
inverse (random μ)   | 0.7648568091385106
inverse (Gershgorin)| 0.7648568091385106
Rayleigh quotient    | -0.5222118887370194
```

Now, in this matrix, the inverse iteration with random μ is having highest number of iterations as the estimated value of eigenvalue differs a lot from the actual eigen value.

Actual eigen values of this matrix is in the range -0.5 to 0.7, but the estimated random $\mu = 2.36$.

Therefore, this is causing more number of iterations.




```
A =[[-0.38369587 -0.02131508  0.86038931]
 [-0.02131508  0.72461628  0.28882134]
 [ 0.86038931  0.28882134  0.77837625]]

v =[-0.5870214   0.46405637  0.66336835]

μ =[0.36239934]

algorithm            |   λ
-------------------------------------------
power                | 1.3356196498281176
inverse (random μ)   | 0.639891620204652
inverse (Gershgorin)| 1.3356196498281176
Rayleigh quotient    | -0.8562146032860403
```

This example is just the opposite of the one. In this matrix, the estimated μ is very close to the eigen values of the matrix and so, the number of iterations is much less than the gerschgorin μ.

So, from both the above examples, we can say that correct estimation of random μ is very important.