Scientific Computing

Assignment 2

Aditya Chetan (2016217)

October 25, 2018

1 Problem 1

1.1 Problem 1(a)

A possible example could be:

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This being a diagonal matrix, the eigenvalues are 2, 2 and 1. Since the most dominant eigenvalue, i.e. 2, is repeated in this case, the power iteration method fails to converge to 2.

1.2 Problem 1(b)

The output of the code is:

```
Absolute Value of Largest Eigenvalue: 11.0
Normalized Eigenvector: [0.5  1.   0.75]
```

1.3 Problem 1(c)

I used the 3^{rd} strategy given in Section 4.5.4 of the textbook (Heath, second edition). My output was:

```
Absolute Value of Largest Eigenvalue: 11.0
Normalized Eigenvector: [0.5  1.   0.75]
Absolute Value of Second - largest Eigenvalue: 3.0000000000000018
```

2 Problem 2

2.1 Problem 2(b)

Output of the code:

```
Shape of Matrix A = (5, 5)
Relative Error = 8.778210295560723e-17
Condition Number of A = 18.145302142090713
Condition Number of QR = 18.145302142090713


Shape of Matrix A = (10, 10)
Relative Error = 7.066567493571102e-17
Condition Number of A = 256.66074933461044
Condition Number of QR = 256.66074933460953


Shape of Matrix A = (100, 80)
Relative Error = 7.610736030788134e-17
Condition Number of A = 256.66074933461044
Condition Number of QR = 131.78216781279517
```

2.2 Problem 2(c)

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Results for Degree: 1



------------------------------------------------------------------------------
Strategy: np.linalg.lstsq
------------------------------------------------------------------------------
Approximate Polynomial:


a * x + b

a = -0.03557566681922611
b = 80.22060628097395



Relative Residual: 0.015929705634960642



------------------------------------------------------------------------------
Strategy: get_QR()
------------------------------------------------------------------------------
Approximate Polynomial:


a * x + b

a = -0.03557566681922585
b = 80.22060628097384



Relative Residual: 0.01592970563496064



++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Results for Degree: 2



------------------------------------------------------------------------------
Strategy: np.linalg.lstsq
------------------------------------------------------------------------------
Approximate Polynomial:


a * x^2 + b * x + c

a = 6.273083504929056e-05
b = -0.052763915622735635
c = 81.0084010178019



Relative Residual: 0.015244734332359157
```

```
------------------------------------------------------------------------
Strategy: get_QR()
------------------------------------------------------------------------
Approximate Polynomial:


a * x^2 + b * x + c

a = 6.273083504926963e-05
b = -0.05276391562272574
c = 81.00840101780094



Relative Residual: 0.015244734332359159



++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Results for Degree: 3



------------------------------------------------------------------------
Strategy: np.linalg.lstsq
------------------------------------------------------------------------
Approximate Polynomial:


a * x^3 + b * x^2 + c * x + d

a = -1.2047260092018508e-06
b = 0.0005578732248385403
c = -0.10713079096742381
d = 82.26111126400068



Relative Residual: 0.0139525317487036



------------------------------------------------------------------------
Strategy: get_QR()
------------------------------------------------------------------------
Approximate Polynomial:


a * x^3 + b * x^2 + c * x + d

a = -1.2047260091970657e-06
b = 0.0005578732248292594
c = -0.10713079096577004
d = 82.26111126394433



Relative Residual: 0.013952531748703596
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Results for Degree: 4


------------------------------------------------------------------------------
Strategy: np.linalg.lstsq
------------------------------------------------------------------------------
Approximate Polynomial:


a * x^4 + b * x^3 + c * x^2 + d * x + e

a = 2.0809783618513933e-08
b = -1.2608487410130376e-05
c = 0.0025690228688658406
d = -0.2301113980342502
e = 83.97365489026379



Relative Residual: 0.011891820265493213



------------------------------------------------------------------------------
Strategy: get_QR()
------------------------------------------------------------------------------
Approximate Polynomial:


a * x^4 + b * x^3 + c * x^2 + d * x + e

a = 2.080978360737391e-08
b = -1.2608487426041043e-05
c = 0.0025690228702602296
d = -0.23011139774827463
e = 83.97365484363354



Relative Residual: 0.011891820265493161



++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Results for Degree: 5


------------------------------------------------------------------------------
Strategy: np.linalg.lstsq
------------------------------------------------------------------------------
Approximate Polynomial:


a * x^5 + b * x^4 + c * x^3 + d * x^2 + e * x + f
```

```
a = -2.739836135745577e-10
b = 2.084886258574628e-07
c = -5.836089336036795e-05
d = 0.007283081136799612
e = -0.41599053323218416
f = 85.71480175612375



Relative Residual: 0.00989910296471503



------------------------------------------------------------------------
Strategy: get_QR()
------------------------------------------------------------------------
Approximate Polynomial:


a * x^5 + b * x^4 + c * x^3 + d * x^2 + e * x + f

a = -2.7398362004125777e-10
b = 2.084885633332005e-07
c = -5.8360859922290795e-05
d = 0.007283075899652939
e = -0.4159903618497009
f = 85.71482398834198



Relative Residual: 0.009899102962613315
```

Here is the obtained plot in figure 1
Evaluation:

- Yes, the methods differ very slightly in their accuracy. In my case, my own implementation of QR factorisation gives a slightly lower relative residual than `numpy.linalg.lstsq()`
- Yes, the polynomial degrees differ in their relative residuals obtained. Degree 5 Polynomial has the least residual and hence the best accuracy. This is probably because a degree 5 polynomial can capture more features of the input data and hence can better approximate it.

3 Problem 3

3.1 Problem 3(a)

Here is the obtained output:

```
Inverse Iteration
------------------


Seed: 1
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Number of iterations: 13
```

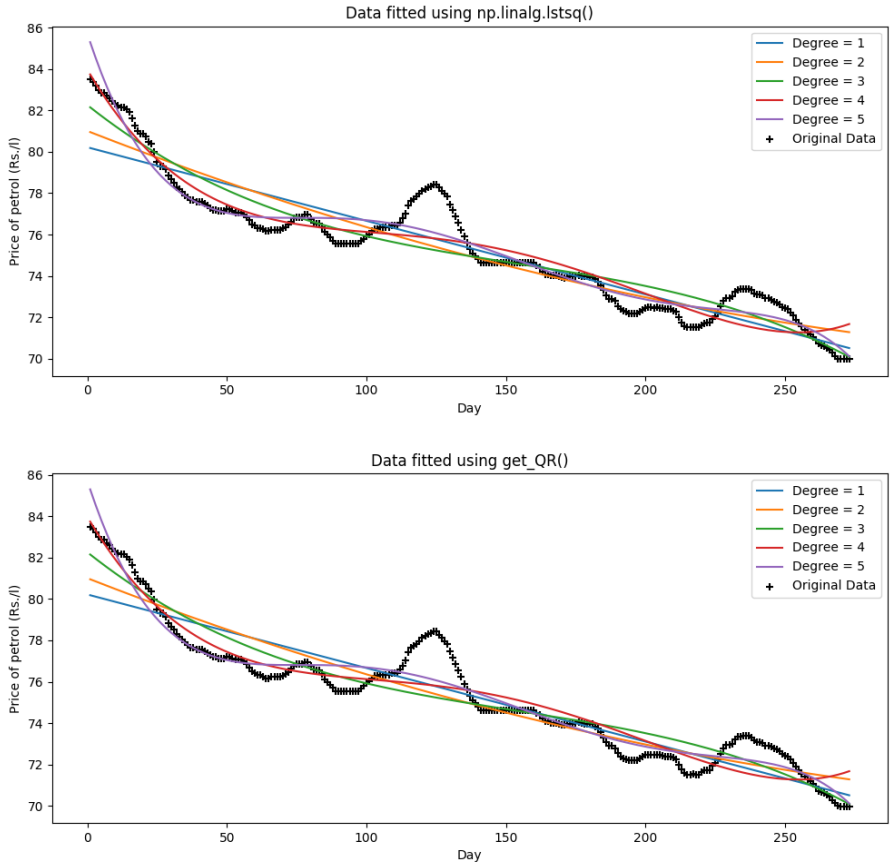

Figure 1: Plots of the polynomial by method.

```
Seed: 89
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Number of iterations: 14



Seed: 98
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Number of iterations: 13



Seed: 23
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Number of iterations: 13



Seed: 88
```

```
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Number of iterations: 13



Seed: 91
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Number of iterations: 13



Seed: 101
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Number of iterations: 13



Seed: 11
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Number of iterations: 14



Seed: 17
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Number of iterations: 12



Seed: 19
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Number of iterations: 12
```

3.2 Problem 3(b)

Here is the obtained output:

```
Rayleigh Quotient Iteration
---------------------------


Seed: 1
Computed Eigenvalue: 2.133074475348525
```

```
Computed Eigenvector:
    [ 0.49742503 -0.8195891  -0.28432735] ’

Number of iterations: 6



Seed: 89
Computed Eigenvalue: 7.287992138960421
Computed Eigenvector:
    [-0.86643225 -0.45305757 -0.20984279] ’

Number of iterations: 5



Seed: 98
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Number of iterations: 7



Seed: 23
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Number of iterations: 4



Seed: 88
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Number of iterations: 5



Seed: 91
Computed Eigenvalue: 7.287992138960422
Computed Eigenvector:
  [-0.86643225 -0.45305757 -0.20984279] ’

Number of iterations: 5



Seed: 101
Computed Eigenvalue: 7.287992138960422
Computed Eigenvector:
  [0.86643225 0.45305757 0.20984279] ’

Number of iterations: 4
```

```
Seed: 11
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [ 0.49742503 -0.8195891  -0.28432735] '

Number of iterations: 8



Seed: 17
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [ 0.49742503 -0.8195891  -0.28432735] '

Number of iterations: 5



Seed: 19
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [ 0.49742503 -0.8195891  -0.28432735] '

Number of iterations: 6
```

3.3 Problem 3(c)

Here are my observations:

- Inverse iteration always converges to the same eigenvalue which is also the smallest eigenvalue of the matrix. On the contrary Rayleigh Quotient iteration may converge to different eigenvalues with different starting vectors.
- Rayleigh Quotient iteration takes relatively fewer number of iterations to reach convergence.

3.4 Problem 3(d)

Here is my output:

```
Inverse Iteration
------------------


Seed: 1
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 13



Seed: 89
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 14
```

```
Seed: 98
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 13



Seed: 23
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0
Number of iterations: 13



Seed: 88
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 13



Seed: 91
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 13



Seed: 101
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 13



Seed: 11
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
```

```
  [-0.49742503  0.8195891   0.28432735] ’

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 14



Seed: 17
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 12



Seed: 19
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0
Number of iterations: 12



Rayleigh Quotient Iteration
------------------------------


Seed: 1
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [ 0.49742503 -0.8195891  -0.28432735] ’

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 3.72380122987091e-16
Number of iterations: 6



Seed: 89
Computed Eigenvalue: 7.287992138960421
Computed Eigenvector:
  [-0.86643225 -0.45305757 -0.20984279] ’

Relative Error in Eigenvalues: 6.093437004082842e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 5



Seed: 98
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] ’
```

```
Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 7



Seed: 23
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 4



Seed: 88
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 2.0000000000000004
Number of iterations: 5



Seed: 91
Computed Eigenvalue: 7.287992138960422
Computed Eigenvector:
  [-0.86643225 -0.45305757 -0.20984279] '

Relative Error in Eigenvalues: 7.31212440489941e-16
Relative Error in Eigenvectors: 2.0
Number of iterations: 5



Seed: 101
Computed Eigenvalue: 7.287992138960422
Computed Eigenvector:
  [0.86643225 0.45305757 0.20984279] '

Relative Error in Eigenvalues: 7.31212440489941e-16
Relative Error in Eigenvectors: 1.922962686383564e-16
Number of iterations: 4



Seed: 11
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [ 0.49742503 -0.8195891  -0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 3.5544479789666735e-16
Number of iterations: 8
```

```
Seed: 17
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [ 0.49742503 -0.8195891  -0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 3.33066907387547e-16
Number of iterations: 5



Seed: 19
Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [ 0.49742503 -0.8195891  -0.28432735] '

Relative Error in Eigenvalues: 4.163841581550989e-16
Relative Error in Eigenvectors: 3.5544479789666735e-16
Number of iterations: 6
```

3.5 Problem 3(e)

My output is:

```
Inverse Iteration
-------------------


Initial Array:
  [1 4 2] '

Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [-0.49742503  0.8195891   0.28432735] '

Number of iterations: 12



Rayleigh Quotient Iteration
-----------------------------


Initial Array:
  [1 4 2] '

Computed Eigenvalue: 2.133074475348525
Computed Eigenvector:
  [ 0.49742503 -0.8195891  -0.28432735] '

Number of iterations: 7
```

4 Problem 4

4.1 Problem 4(a)

The plot of the data is given in figure 2.

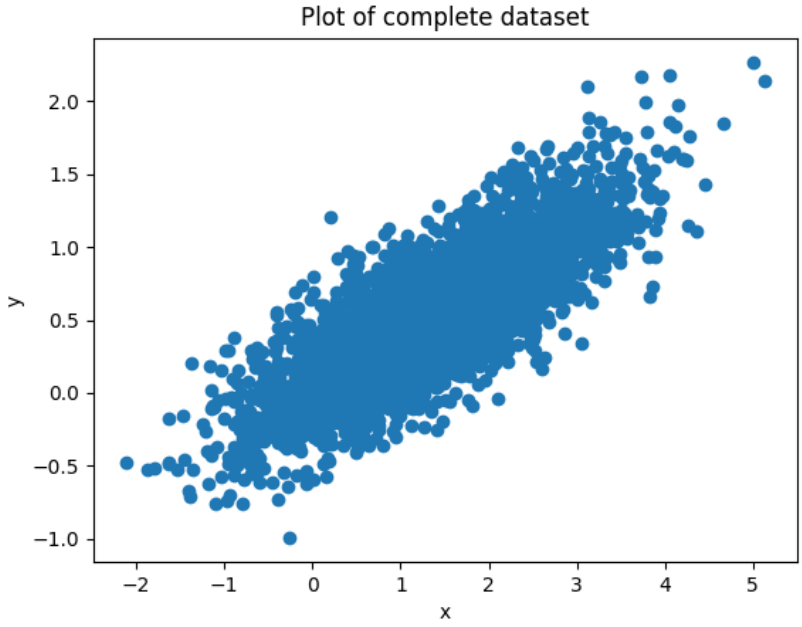

Figure 2: Plot of the Dataset

4.2 Problem 4(b)

The plot given in figure 3.

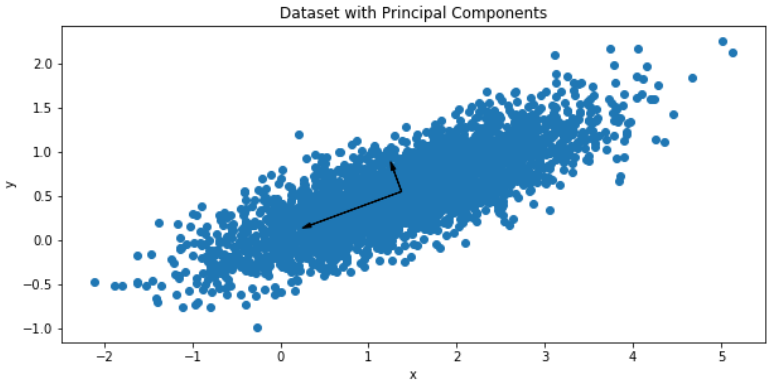

Figure 3: Dataset with Principal Components.

4.3 Problem 4(c)

The output of the code is:

```
Relative error of Y: 1.2936635080314248e-15
```

4.4 Problem 4(d)

The plot obtained is given in figure 4.

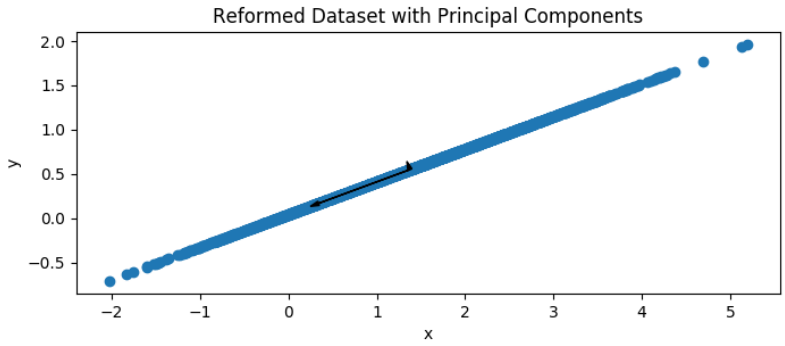

Figure 4: Changed (Noiseless) Dataset with Principal Components.

My interpretation here is as follows:

- When we made the bottom right corner of Σ to 0, we essentially removed one principal component of the data.
- Consequently, on reconstructing the data from these changed principal components, we obtain data points which have no component along the nullified principal component.
- This could be helpful in filters for removing noise.
- If we knew the direction of the noise in our dataset, we could simply take out the principal components, nullify the direction of noise and reconstruct the dataset.
- This would also mean that we would not even have to store features of the data points in the direction of the noise and hence, preserve memory.

5 Problem 5

My output is as follows:

```
Matrix:
 [[ 2  3  2]
 [10  3  4]
 [ 3  6  1]]


Computed eigenvalues:  [11. -3. -2.]
Actual eigenvalues:  [11. -2. -3.]



Matrix:
 [[6 2 1]
 [2 3 1]
 [1 1 1]]


Computed eigenvalues:  [7.28799214 2.13307448 0.57893339]
Actual eigenvalues:  [7.28799214 2.13307448 0.57893339]
```

6 Problem 6

6.1 Problem 6(a)

- **(i)** We know that, $H = Q^T AQ$. Hence,

$$H^T = (Q^T AQ)^T = Q^T A^T (Q^T)^T = Q^T A^T Q = Q^T AQ = H \text{ [As A is symmetric]}$$
$$\implies H^T = H$$

  Hence, H is symmetric.

- **(ii)** We know that H is upper Hessenberg. Hence, entries below the sub-diagonal of H are 0. Now, we also know that H is symmetric. Hence, for any entry a_{ij} above the super diagonal, it must be equal to entry a_{ji} below the sub-diagonal. But entris below the sub-diagonal are all 0. Hence, all entries above the super diagonal must also be 0.

- **(iii)** The outer loop in Algorithm 4.9 iterates over the columns of H whereas the inner loop iterates to all rows uptil the current column, k. Now we know that H is upper Hessenberg. So the only elements in a column, k that are non-zero are $h_{k-1,k}, h_{k,k}, h_{k+1,k}$.

  But due to the inner loop, the only two elements that will be filled are: $h_{k-1,k}, h_{k,k}$. Thus, only two iterations would suffice in the inner loop once we know that A is symmetric.

  In addition, when on column k, the steps of significant note are:

  - when $\mathbf{j = k - 1}$:

$$h_{k-1,k} = q_{k-1}^H u_k$$

    and,

$$u_k = u_k - h_{k-1,k} q_{k-1}$$

  - when $\mathbf{j = k}$:

$$h_{k,k} = q_k^H u_k$$

    and,

$$u_k = u_k - h_{k,k} q_k$$

    , which, using the previous change to u_k, can be written as,

$$u_k = u_k - h_{k-1,k} q_{k-1} - h_{k,k} q_k$$

Things to note here:

- $h_{k,k} = q_k^H u_k$ clearly resembles the step, $\alpha_k = q_k^H u_k$ in Lanczos iteration. Thus, $h_{k,k} = \alpha_k$.
- Similarly comparing, $u_k = u_k - h_{k-1,k} q_{k-1} - h_{k,k} q_k$ with $u_k = u_k - \beta_{k-1} q_{k-1} - \alpha k q_k$, we can say that $h_{k-1,k} = \beta_{k-1}$

From these observations, we can deduce that Arnoldi iteration reduces to Lanczos iteration when A is symmetric.

6.2 Problem 6(b)

```
Answer for part 6(a) (i): 1.0000000000000033
Answer for part 6(a) (ii): 1.0000000000000016
```

6.3 Problem 6(c)

The plot is shown in figure 5.

Figure 5: Ritz values Vs. No. of iterations