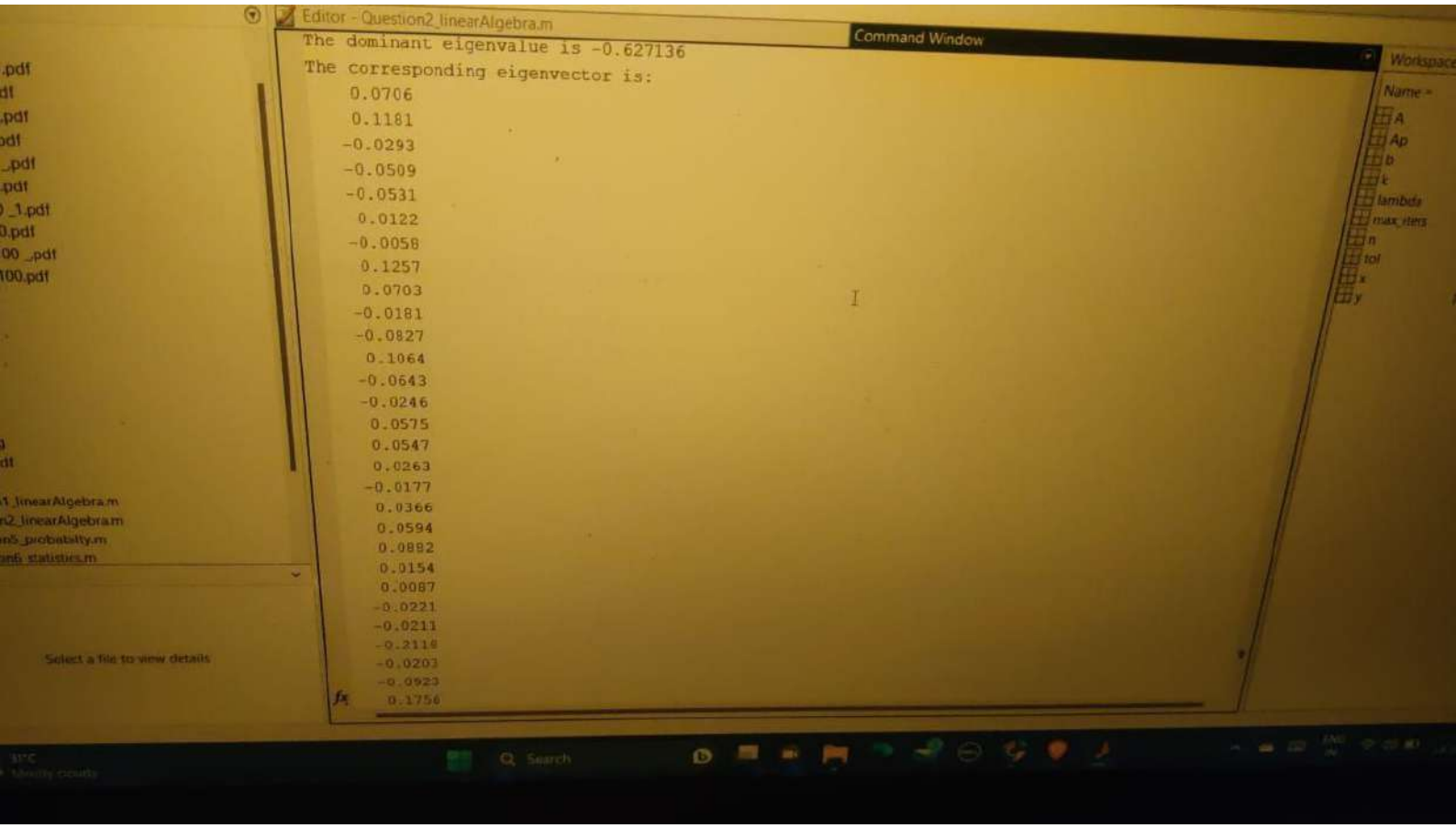
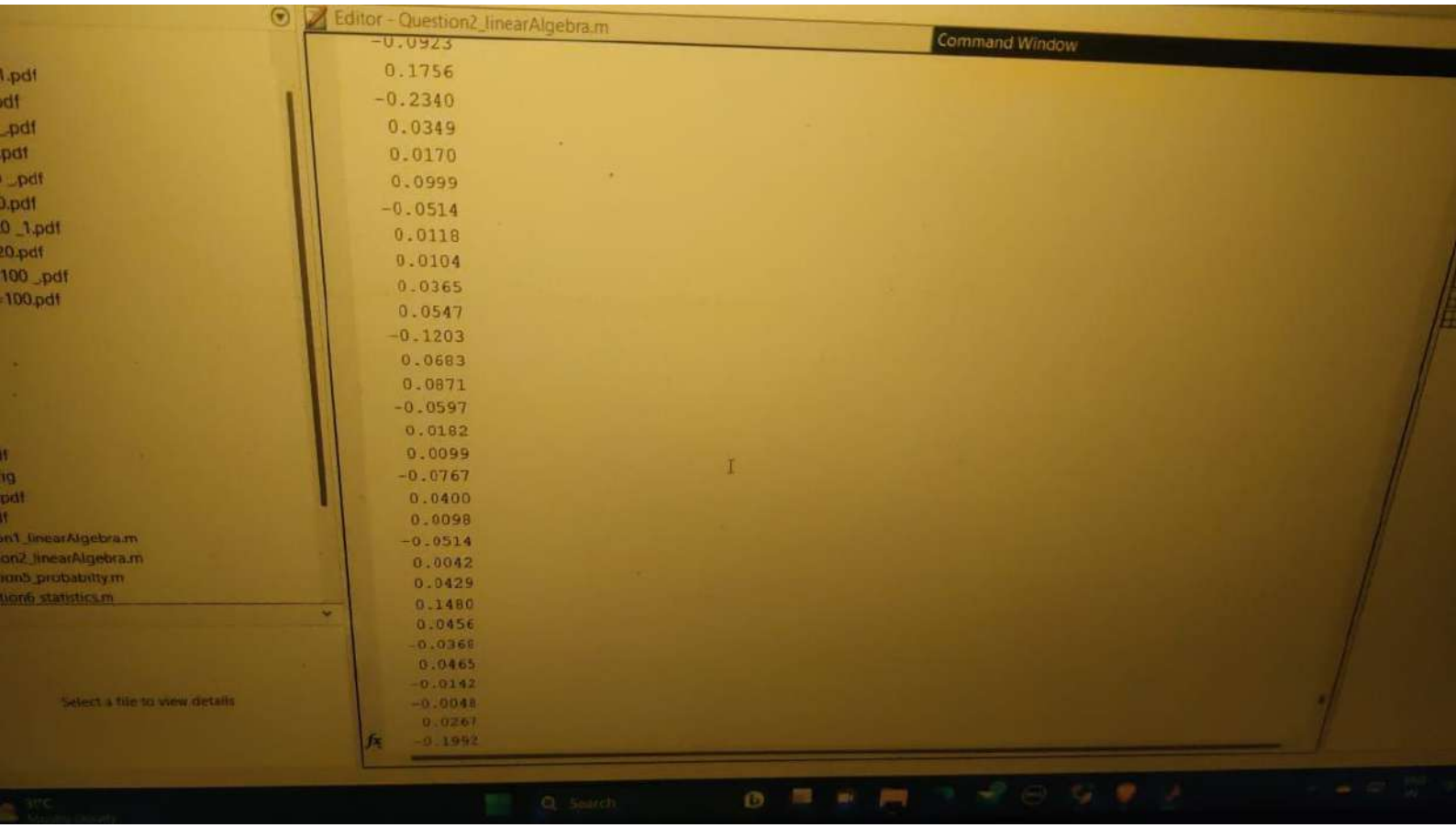
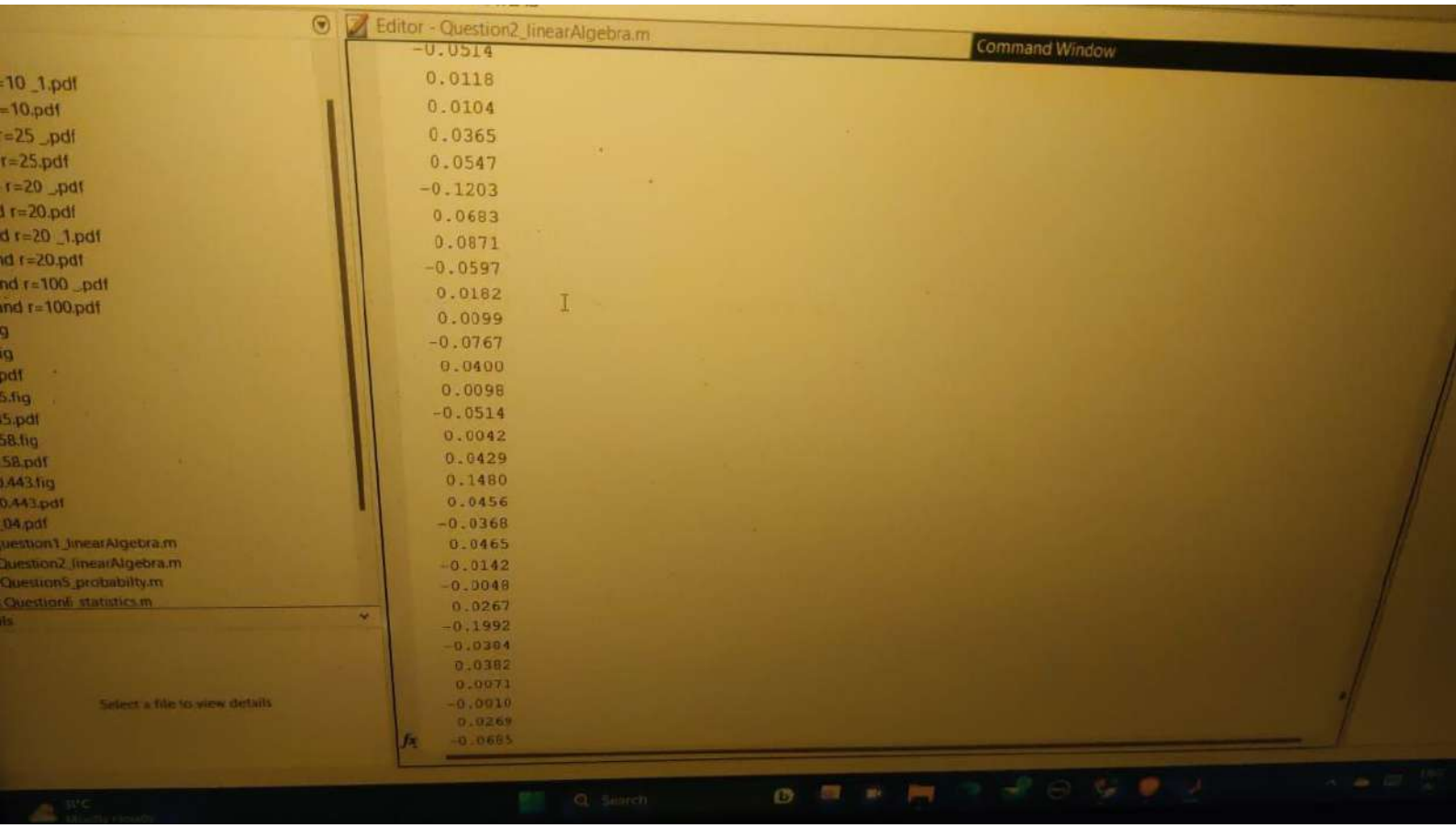
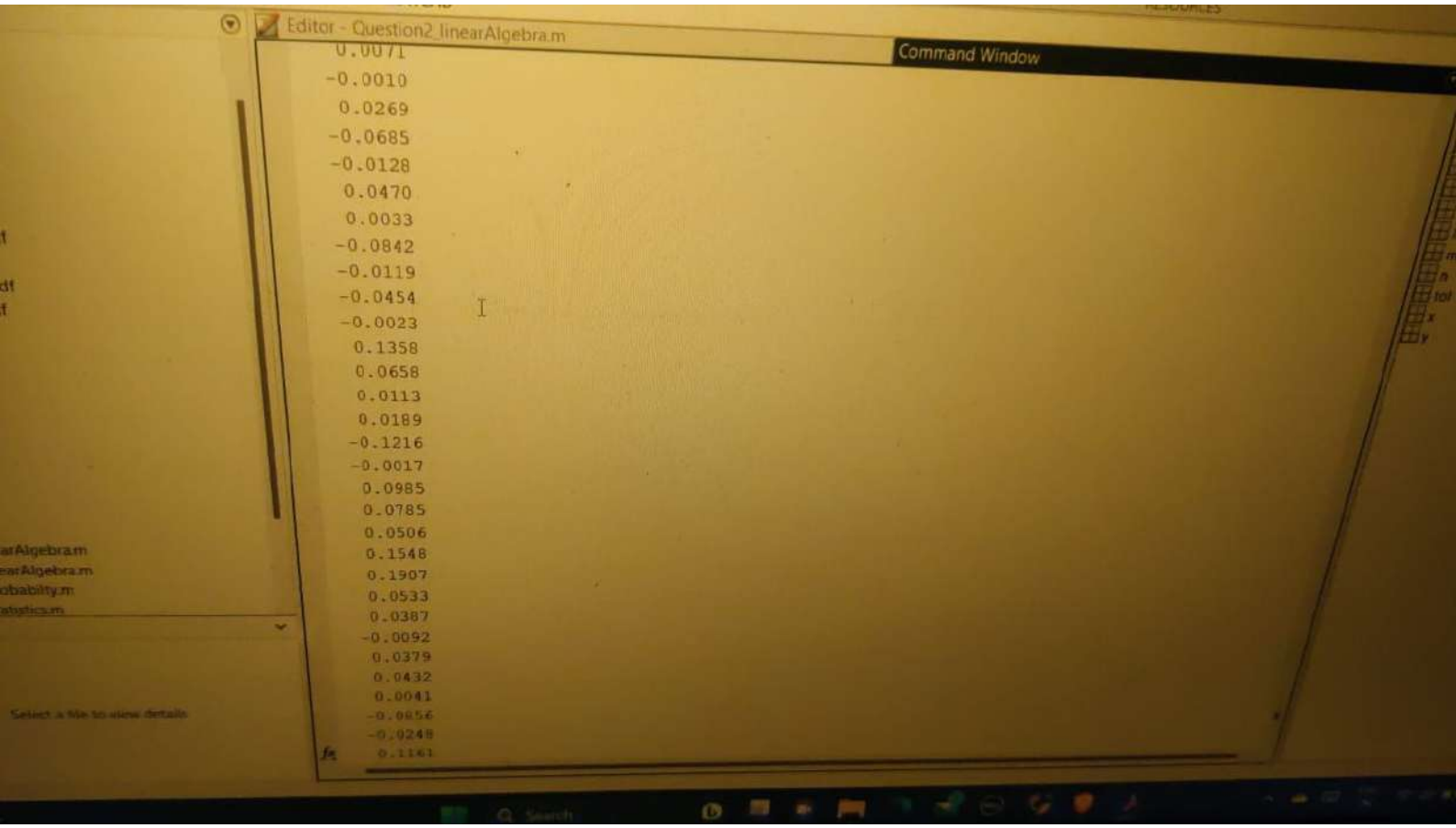
we have a random n-by-n matrix A and a random n-by-1 column vector b, generated using the randn function in MATLAB. You want to find the eigenvalues and eigenvectors of A using the power method, which is an iterative method for finding the dominant eigenvalue and eigenvector of a matrix. The power method involves repeatedly multiplying a vector by A and normalizing the result, until the vector converges to an eigenvector of A. To speed up convergence, you decide to use the shifted power method, which involves subtracting an estimate of the dominant eigenvalue from A before each iteration. To estimate the dominant eigenvalue, you decide to use the Rayleigh quotient method, which involves computing the Rayleigh quotient of a vector and A, and using the result as an estimate of the dominant eigenvalue.

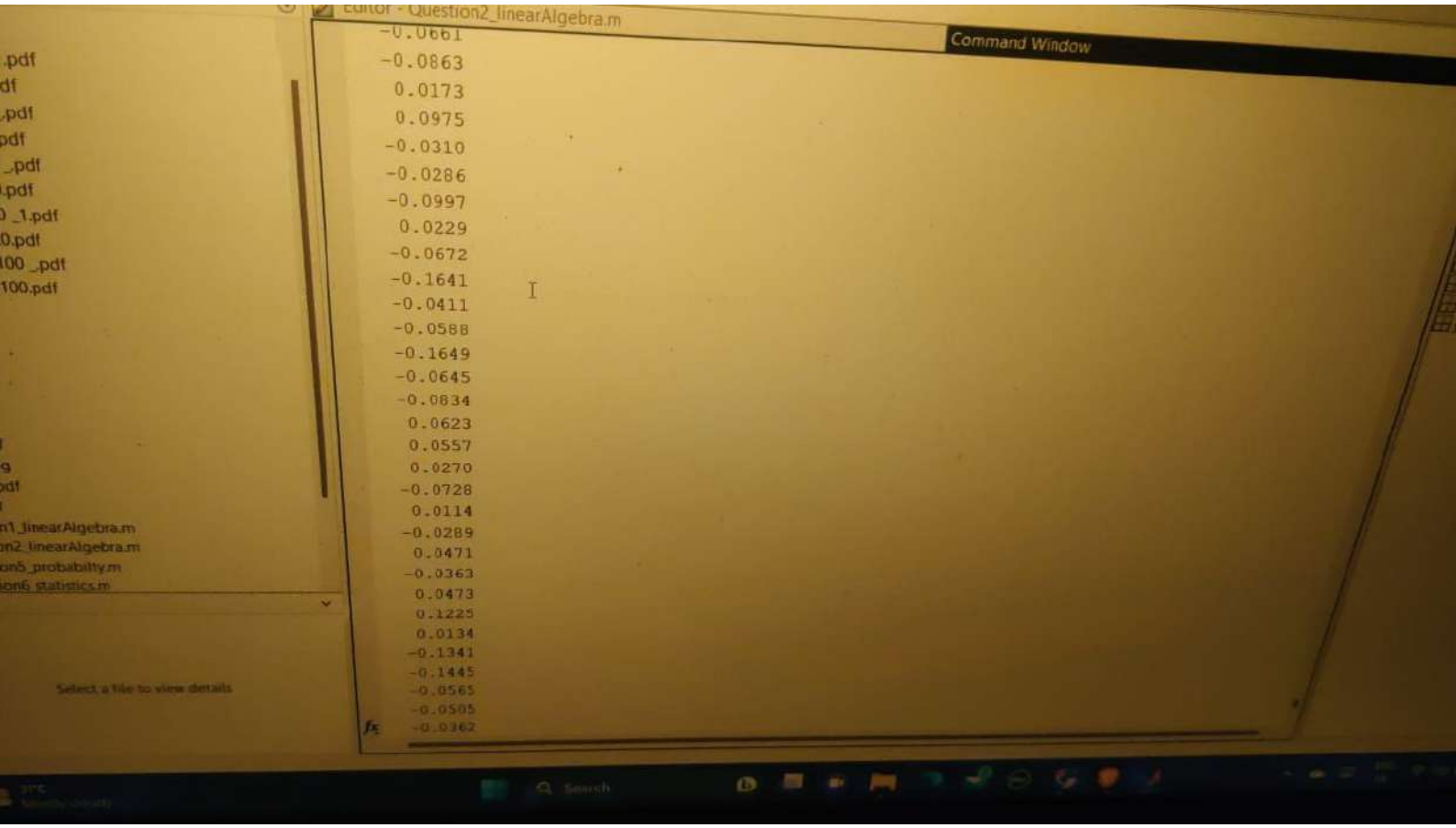
varaint 1 eigen values

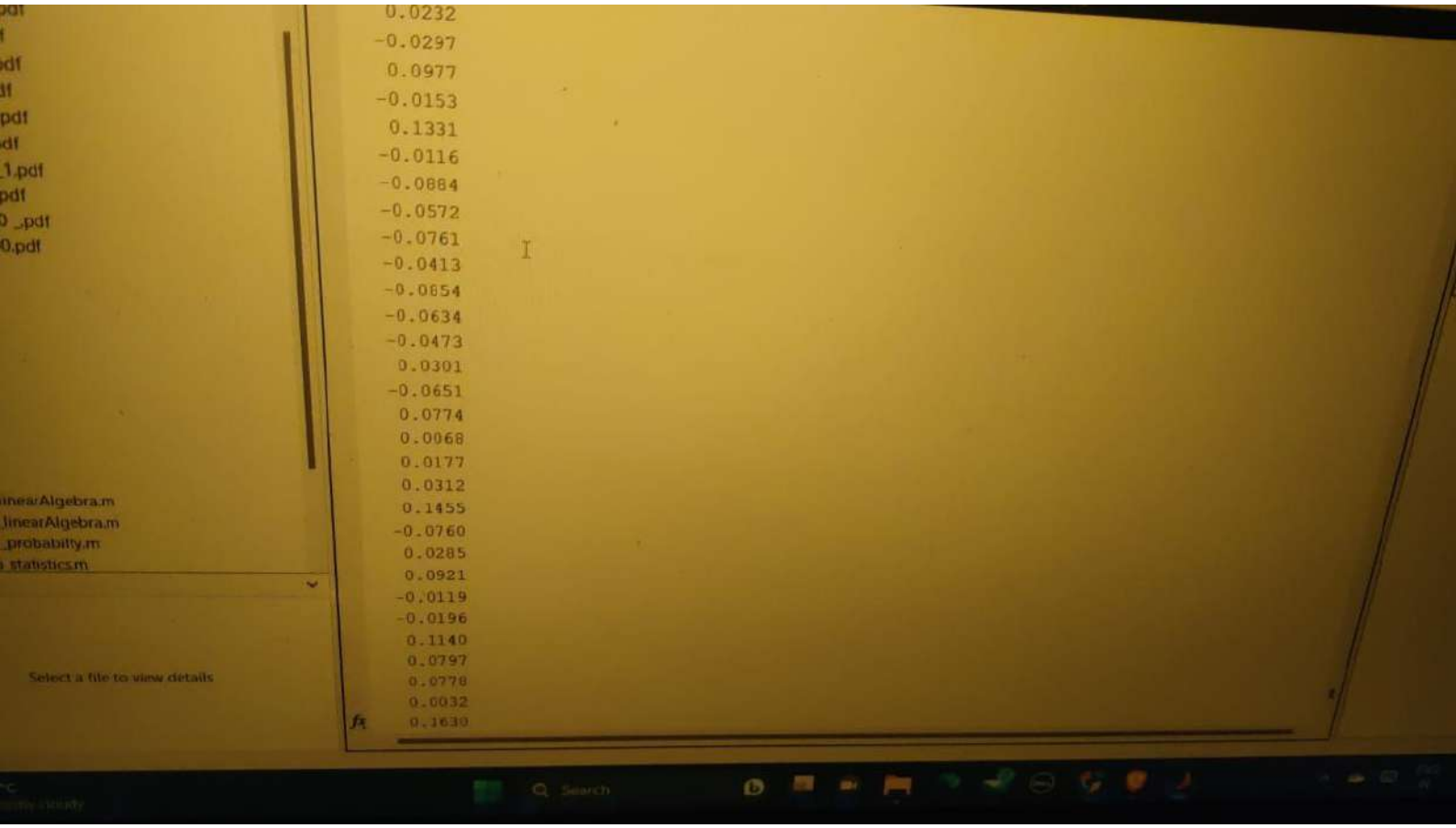


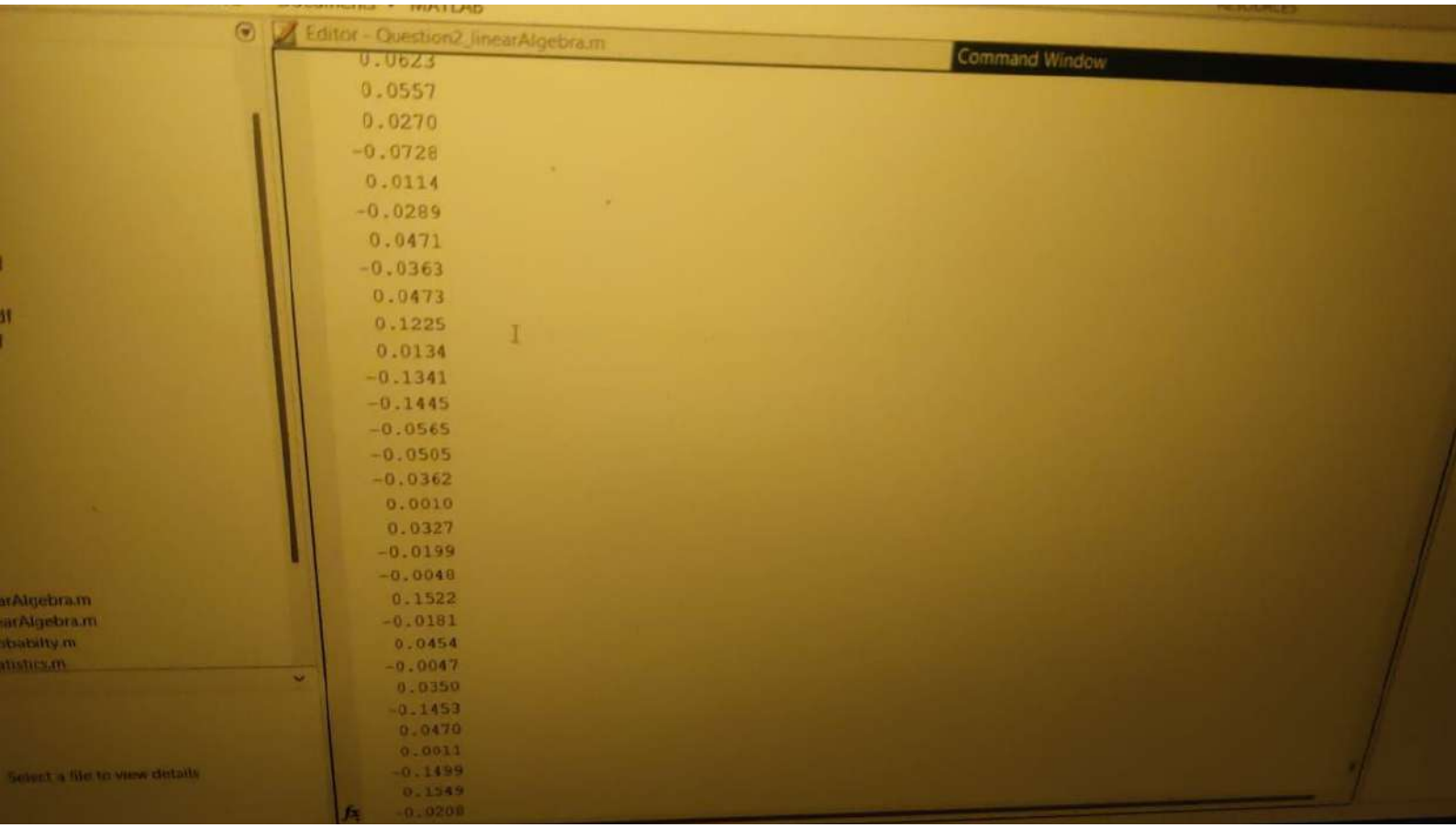

Editor - Question2_linearAlgebra.m
Command Window
-0.0923
0.1756
-0.2340
0.0349
0.0170
0.0999
-0.0514
0.0118
0.0104
0.0365
0.0547
-0.1203
0.0683
0.0871
-0.0597
0.0182
0.0099
-0.0767
0.0400
0.0098
-0.0514
0.0042
0.0429
0.1480
0.0456
-0.0368
0.0465
-0.0142
-0.0048
0.0267
-0.1992
Select a file to view details

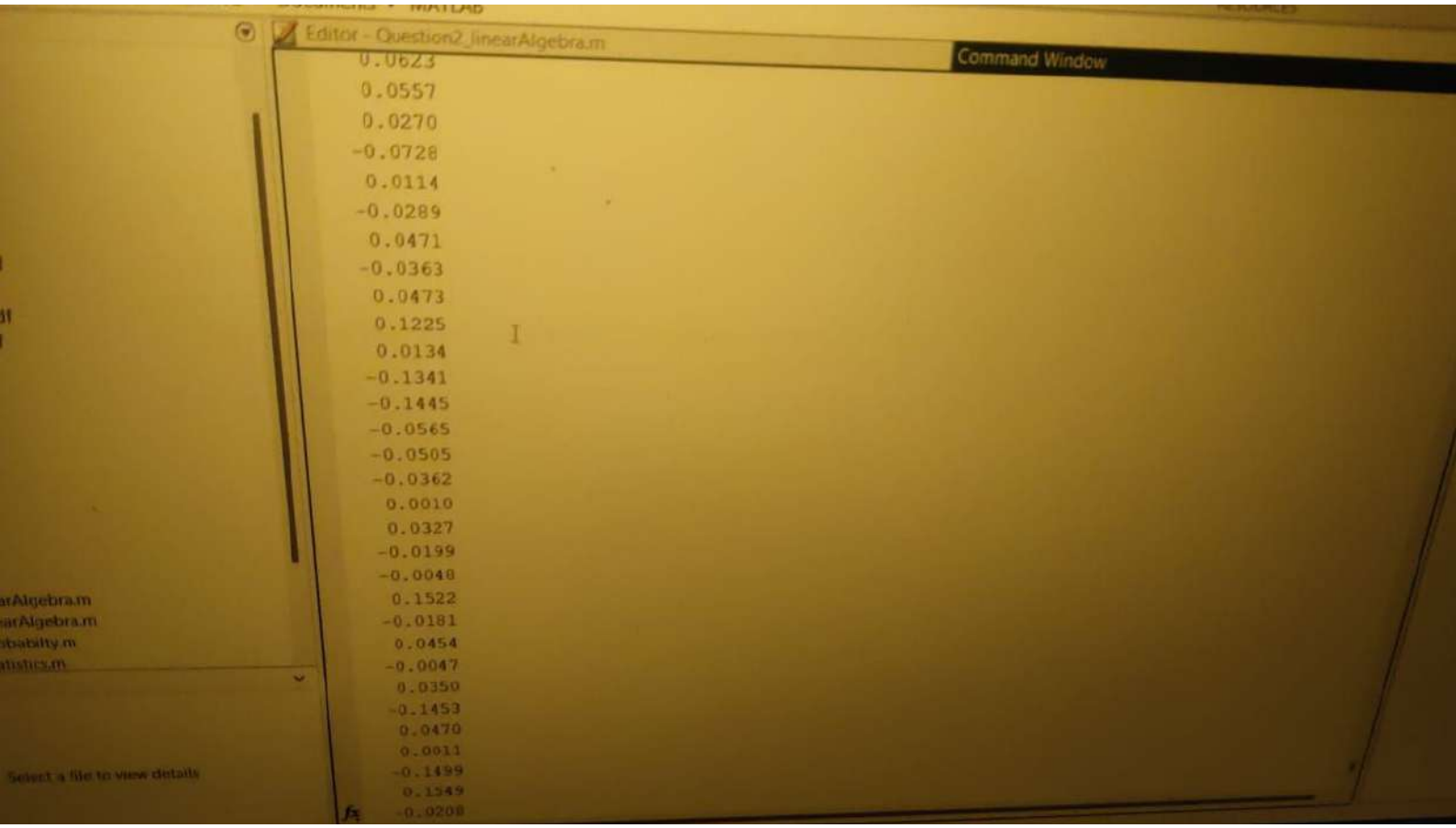

Editor - Question2_linearAlgebra.m
Command Window
-0.0514
0.0118
0.0104
0.0365
0.0547
-0.1203
0.0683
0.0871
-0.0597
0.0182
0.0099
-0.0767
0.0400
0.0098
-0.0514
0.0042
0.0429
0.1480
0.0456
-0.0368
0.0465
-0.0142
-0.0048
0.0267
-0.1992
-0.0304
0.0382
0.0071
-0.0010
0.0269
-0.0685
Question1_linearAlgebra.m
Question2_linearAlgebra.m
Question5_probability.m
Question6_statistics.m
Select a file to view details


Editor - Question2_linearAlgebra.m
Command Window
0.0071
-0.0010
0.0269
-0.0685
-0.0128
0.0470
0.0033
-0.0842
-0.0119
-0.0454
-0.0023
0.1358
0.0658
0.0113
0.0189
-0.1216
-0.0017
0.0985
0.0785
0.0506
0.1548
0.1907
0.0533
0.0387
-0.0092
0.0379
0.0432
0.0041
-0.0856
-0.0248
0.1161

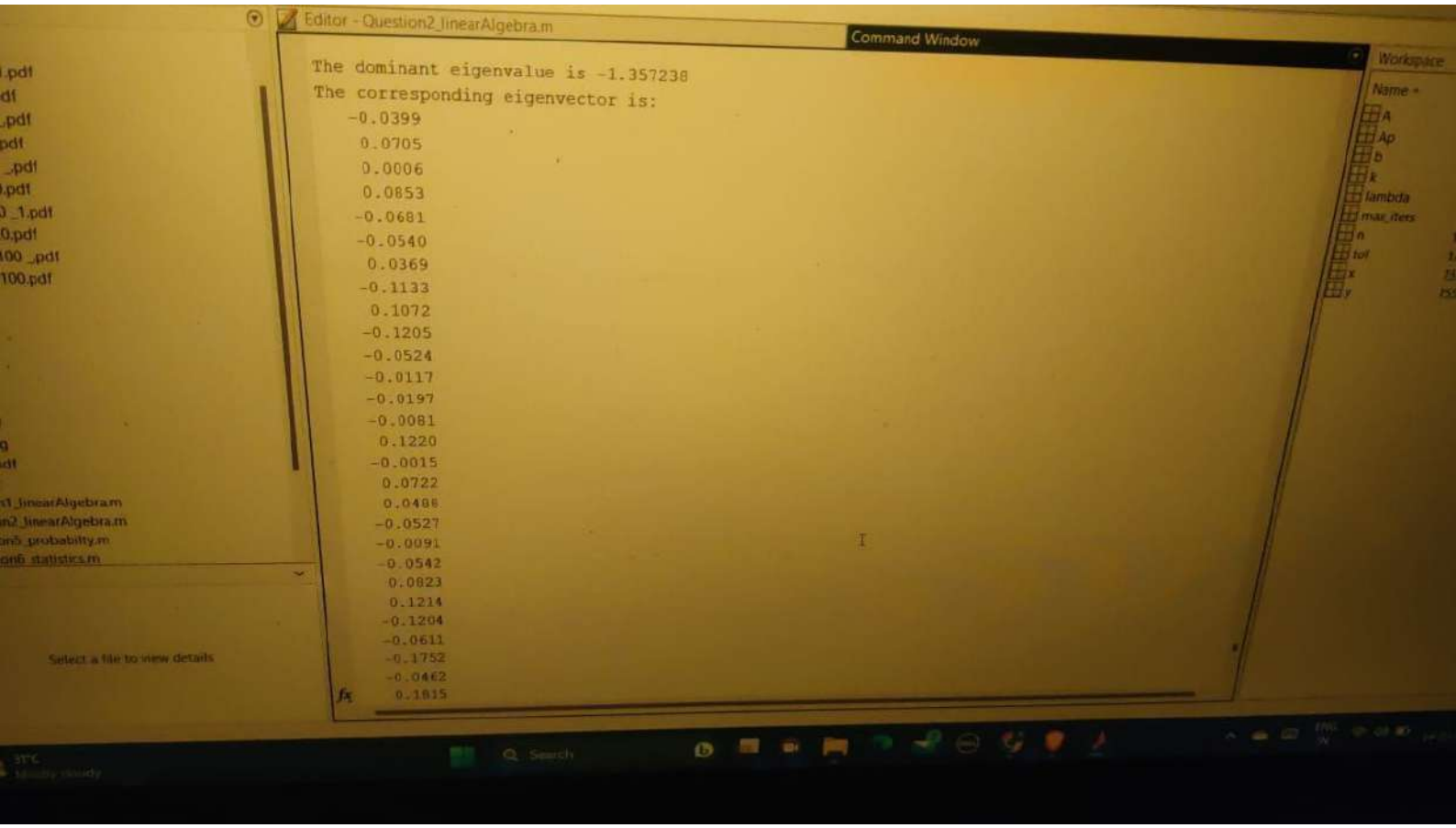

Editor - Question2_linearAlgebra.m
-0.0661
-0.0863
0.0173
0.0975
-0.0310
-0.0286
-0.0997
0.0229
-0.0672
-0.1641
-0.0411
-0.0588
-0.1649
-0.0645
-0.0834
0.0623
0.0557
0.0270
-0.0728
0.0114
-0.0289
0.0471
-0.0363
0.0473
0.1225
0.0134
-0.1341
-0.1445
-0.0565
-0.0505
-0.0362
Command Window
n1_linearAlgebra.m
n2_linearAlgebra.m
on5_probability.m
on6_statistics.m
Select a file to view details

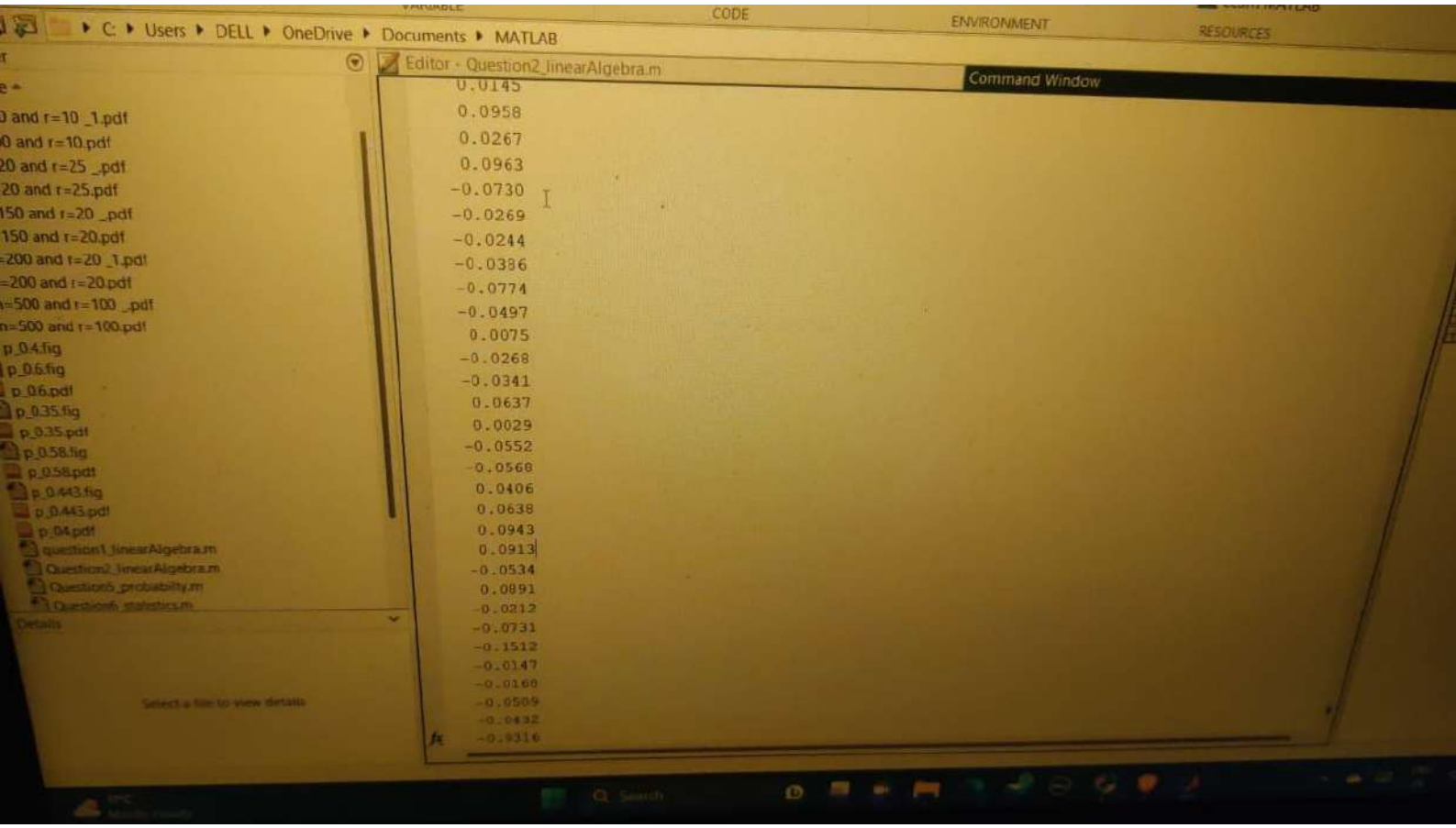

0.0232
-0.0297
0.0977
-0.0153
0.1331
-0.0116
-0.0884
-0.0572
-0.0761
-0.0413
-0.0854
-0.0634
-0.0473
0.0301
-0.0651
0.0774
0.0068
0.0177
0.0312
0.1455
-0.0760
0.0285
0.0921
-0.0119
-0.0196
0.1140
0.0797
0.0778
0.0032
0.1630
Select a file to view details

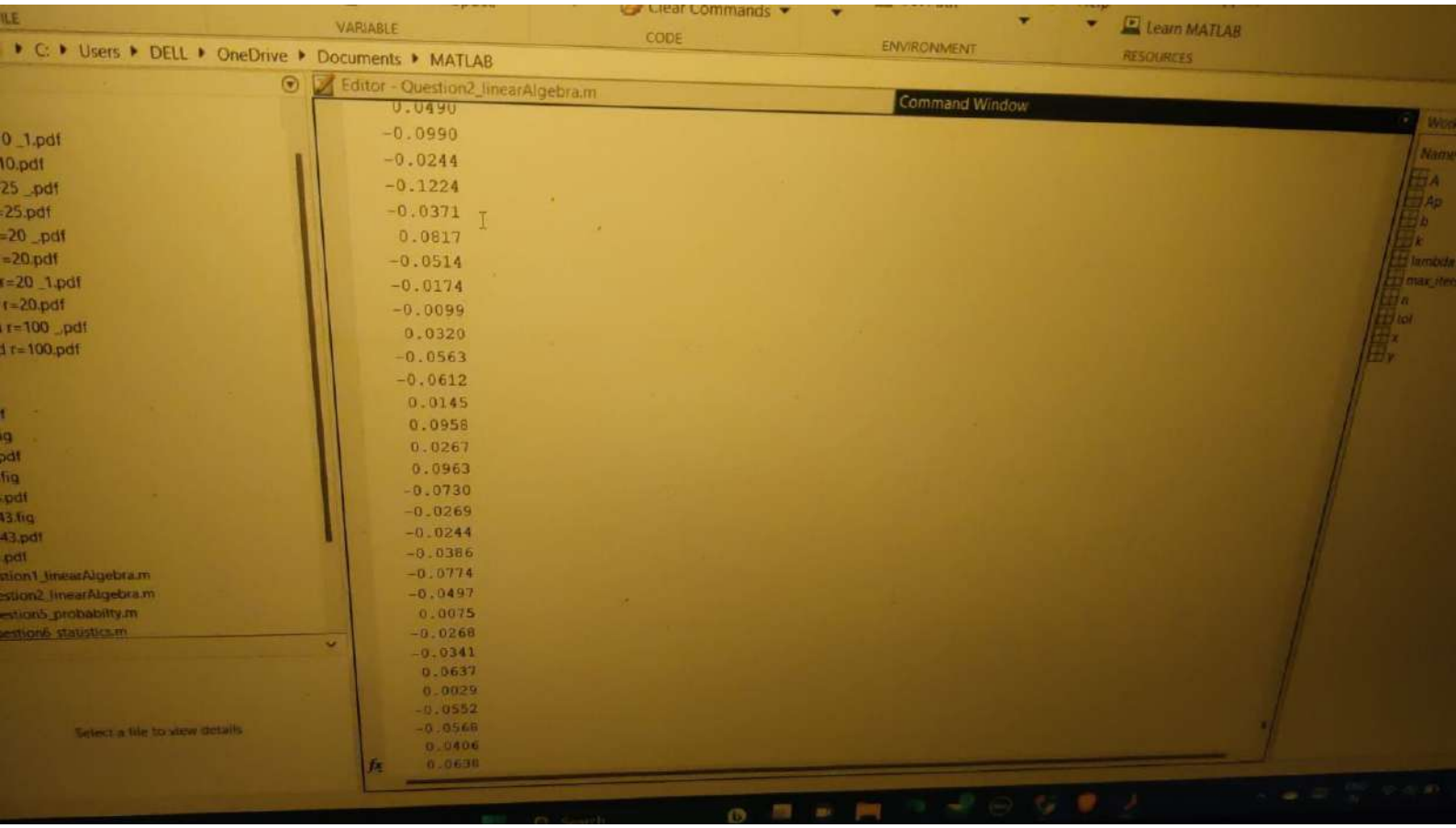

Editor - Question2_linearAlgebra.m
Command Window
0.0623
0.0557
0.0270
-0.0728
0.0114
-0.0289
0.0471
-0.0363
0.0473
0.1225
0.0134
-0.1341
-0.1445
-0.0565
-0.0505
-0.0362
0.0010
0.0327
-0.0199
-0.0048
0.1522
-0.0181
0.0454
-0.0047
0.0350
-0.1453
0.0470
0.0011
-0.1499
0.1549
-0.0208
Select a file to view details

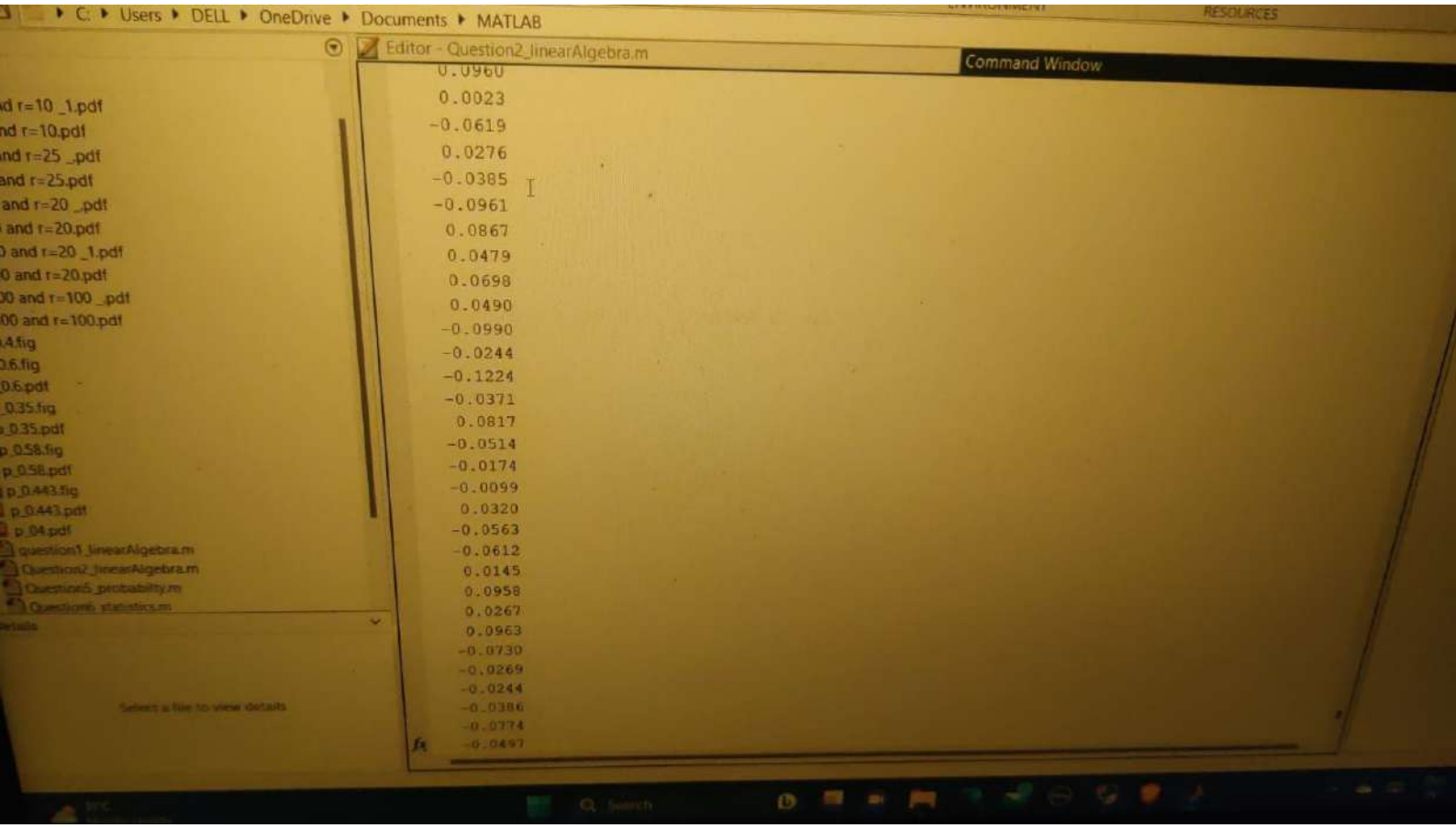

Editor - Question2_linearAlgebra.m
Command Window
0.0623
0.0557
0.0270
-0.0728
0.0114
-0.0289
0.0471
-0.0363
0.0473
0.1225
0.0134
-0.1341
-0.1445
-0.0565
-0.0505
-0.0362
0.0010
0.0327
-0.0199
-0.0048
0.1522
-0.0181
0.0454
-0.0047
0.0350
-0.1453
0.0470
0.0011
-0.1499
0.1549
-0.0208
Select a file to view details

varaiant 2 eigen values




CODE
ENVIRONMENT
RESOURCES
C: ▸ Users ▸ DELL ▸ OneDrive ▸ Documents ▸ MATLAB
Editor - Question2_linearAlgebra.m
Command Window
0.0145
0.0958
0.0267
0.0963
-0.0730
-0.0269
-0.0244
-0.0386
-0.0774
-0.0497
0.0075
-0.0268
-0.0341
0.0637
0.0029
-0.0552
-0.0568
0.0406
0.0638
0.0943
0.0913
-0.0534
0.0891
-0.0212
-0.0731
-0.1512
-0.0147
-0.0160
-0.0509
-0.0432
-0.9316
p_0.4.fig
p_0.6.fig
p_0.6.pdf
p_0.35.fig
p_0.35.pdf
p_0.58.fig
p_0.58.pdf
p_0.443.fig
p_0.443.pdf
p_04.pdf
question1_linearAlgebra.m
Question2_linearAlgebra.m
Question5_probability.m
Question6_statistics.m
Details
Select a file to view details


VARIABLE
CODE
ENVIRONMENT
RESOURCES
Clear Commands
Learn MATLAB
C: ▸ Users ▸ DELL ▸ OneDrive ▸ Documents ▸ MATLAB
Editor - Question2_linearAlgebra.m
Command Window
0.0490
-0.0990
-0.0244
-0.1224
-0.0371
0.0817
-0.0514
-0.0174
-0.0099
0.0320
-0.0563
-0.0612
0.0145
0.0958
0.0267
0.0963
-0.0730
-0.0269
-0.0244
-0.0386
-0.0774
-0.0497
0.0075
-0.0268
-0.0341
0.0637
0.0029
-0.0552
-0.0568
0.0406
0.0638
Select a file to view details


C: ▸ Users ▸ DELL ▸ OneDrive ▸ Documents ▸ MATLAB
RESOURCES
Editor - Question2_linearAlgebra.m
Command Window
0.0960
0.0023
-0.0619
0.0276
-0.0385
-0.0961
0.0867
0.0479
0.0698
0.0490
-0.0990
-0.0244
-0.1224
-0.0371
0.0817
-0.0514
-0.0174
-0.0099
0.0320
-0.0563
-0.0612
0.0145
0.0958
0.0267
0.0963
-0.0730
-0.0269
-0.0244
-0.0386
-0.0774
-0.0497
d r=10 _1.pdf
nd r=10.pdf
nd r=25 _.pdf
and r=25.pdf
and r=20 _.pdf
and r=20.pdf
0 and r=20 _1.pdf
0 and r=20.pdf
00 and r=100 _.pdf
00 and r=100.pdf
4.fig
0.6.fig
0.6.pdf
_0.35.fig
_0.35.pdf
p_0.58.fig
p_0.58.pdf
p_0.443.fig
p_0.443.pdf
p_04.pdf
question1_linearAlgebra.m
Question2_linearAlgebra.m
Question5_probability.m
Question6_statistics.m
Select a file to view details

Given a matrix A of size n x n, where n is an odd positive integer, and a vector b of size n x 1, find the determinant of the matrix B = A + b*b', where b' is the transpose of b.

Here is one possible answer:

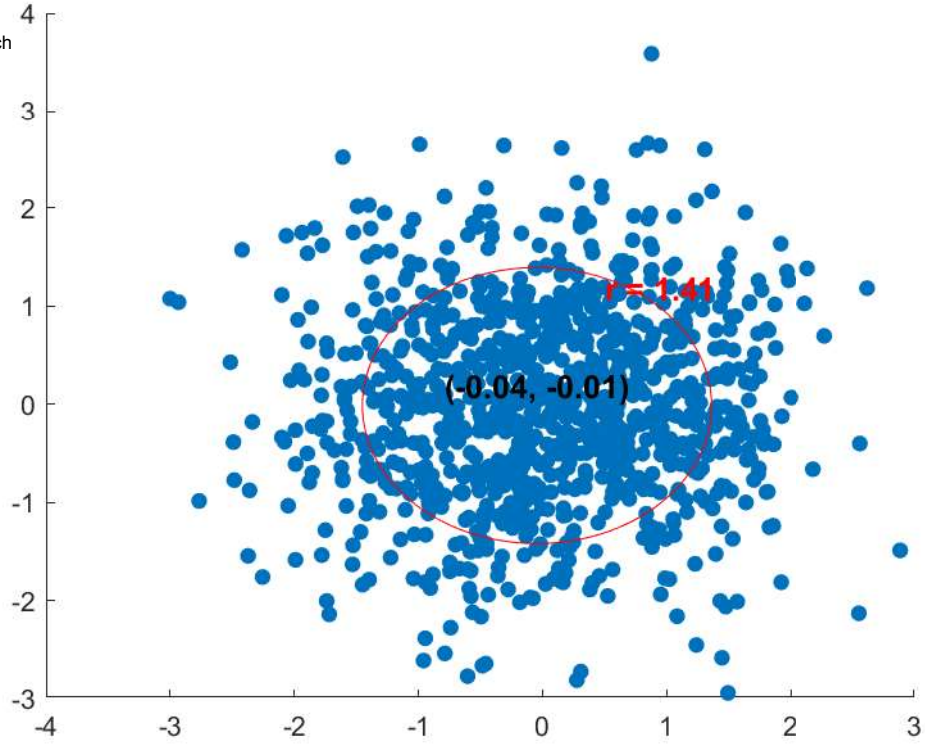
The determinant of B can be found using the matrix determinant lemma, which states that det(B) = det(A) + b'*adj(A)*b, where adj(A) is the adjugate of A. The adjugate of A is the transpose of the matrix of cofactors of A. The cofactor of A(i,j) is (-1)^(i+j) times the determinant of the submatrix obtained by deleting the i-th row and j-th column of A.
The problem was to generate random x and y values with 1000 samples each, plot them on a scatter plot, fit a circle to the points using the least squares method, plot the circle on the same scatter plot, compute the area of the circle, and compute the average distance between the points and the circle.

To solve this problem, we first generated random x and y values using the randn function in MATLAB. We then plotted these points on a scatter plot using the scatter function. Next, we used the least squares method to fit a circle to the points by constructing a linear system of equations and solving it using the pseudoinverse function pinv.

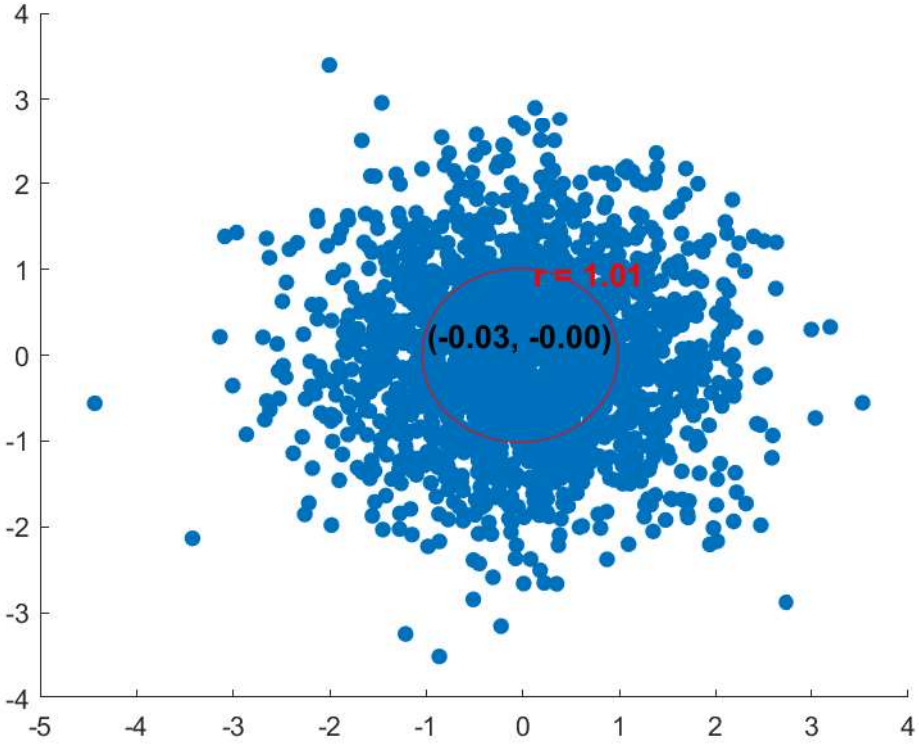
Once we had the center and radius of the fitted circle, we plotted the circle on the same scatter plot using the plot function. We also added labels for the center and radius of the circle to make them more visible in the plot.

To compute the area of the circle, we used the formula for the area of a circle: area = pi*r^2. To compute the average distance between the points and the circle, we first computed the distances between each point and the center of the circle using the Pythagorean theorem. We then took the absolute difference between these distances and the radius of the circle, and computed the mean of these differences.

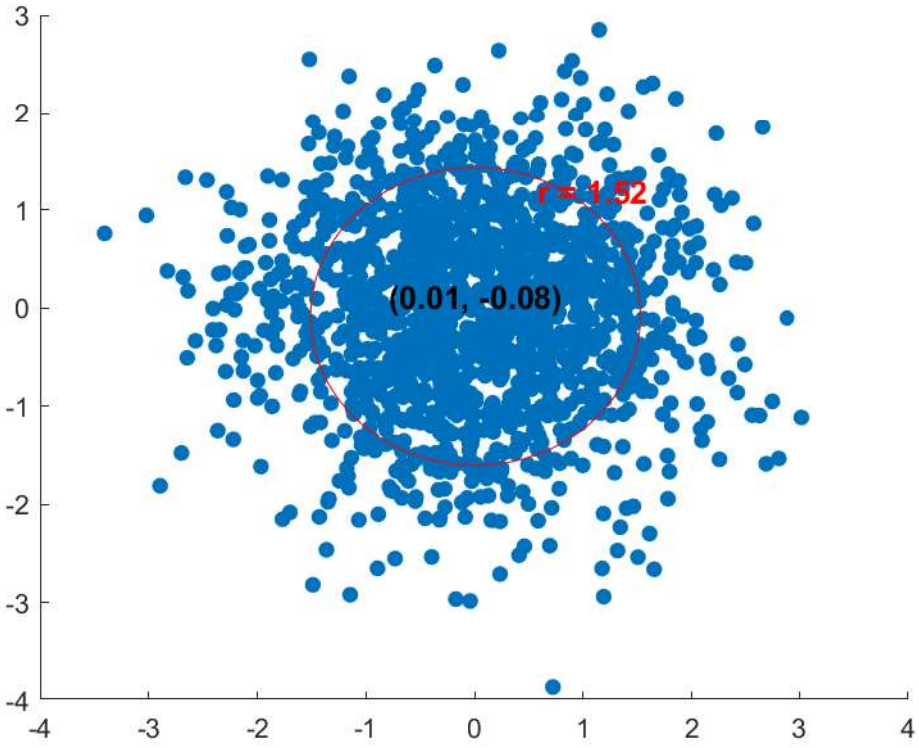
variant 1:x and y 1000 samples each



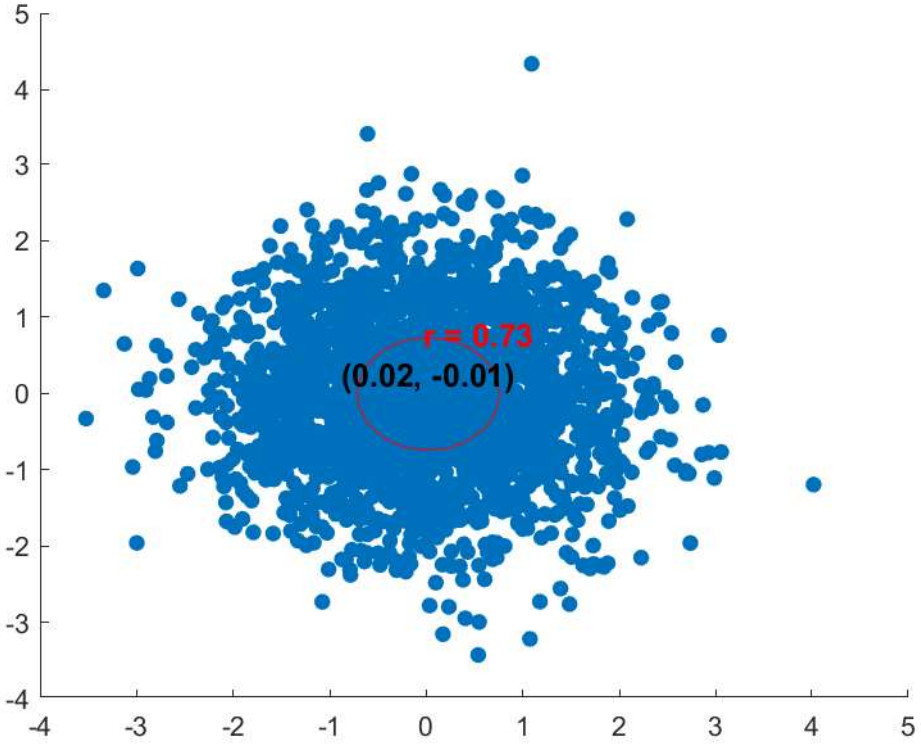
variant 2:x and y 2000 samples each



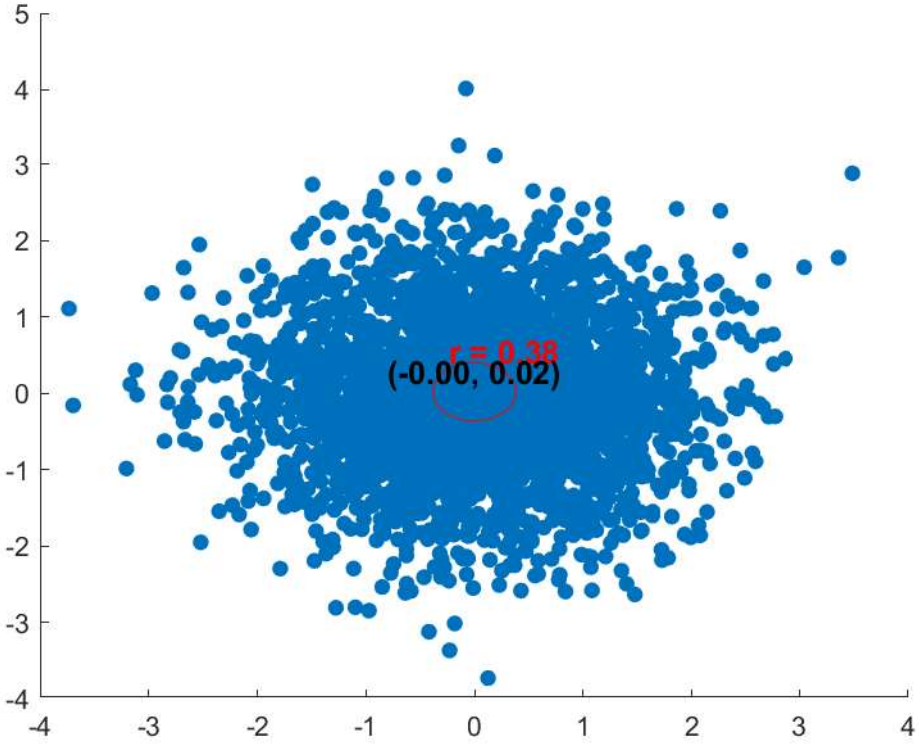
variant 3: x and y 1500 samples each



variant 4:x and y 2500 samples each



variant 5: x and y 3000 samples each



Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To operate on each element of the matrix individually, use TIMES (.*) for elementwise multiplication.

error:T = diag(alpha) + diag(beta,1) + diag(conj(beta),-1);

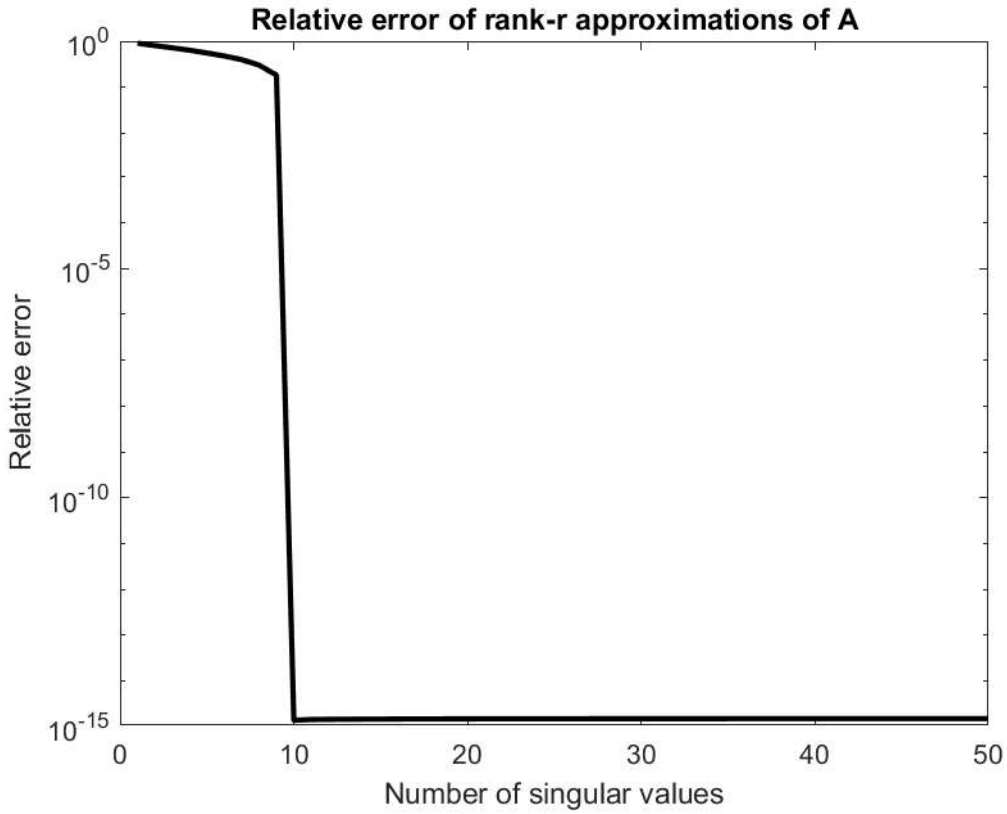
linear algebra
The code is written in MATLAB and uses the built-in svd function to compute the singular value decomposition (SVD) of a randomly generated matrix. The size of the matrix and the rank of its approximation can be easily changed by modifying the values assigned to the n and rank_approx variables at the beginning of the code.

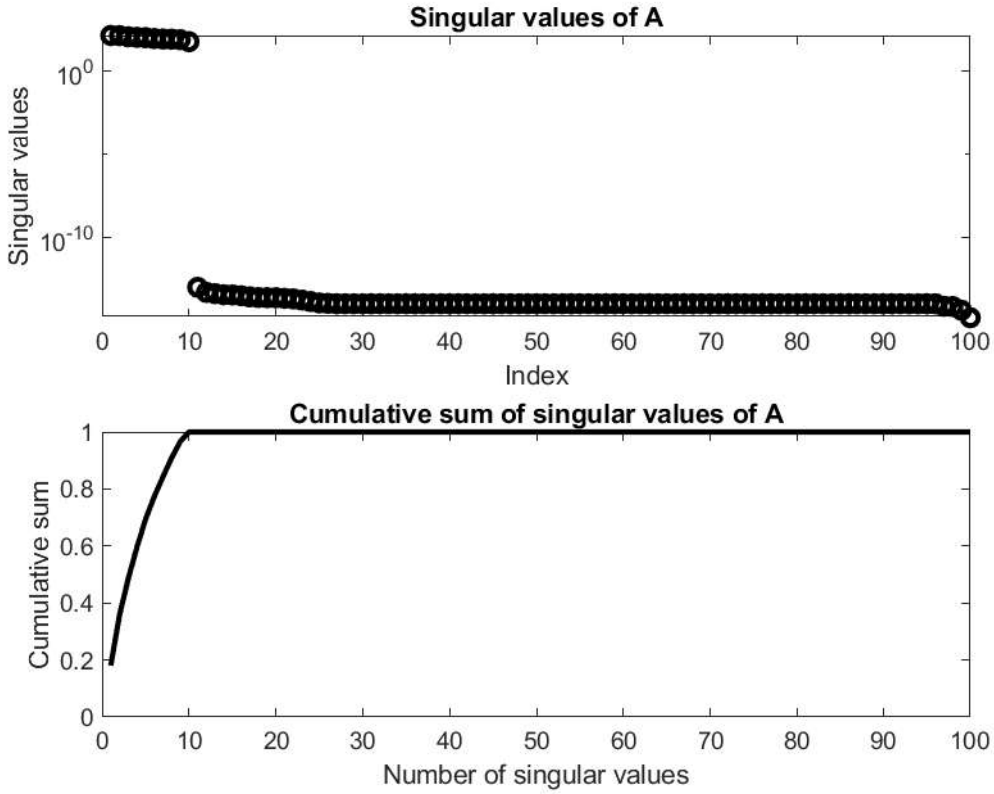
After generating the matrix and computing its SVD, the code computes a rank-rank_approx approximation of the matrix using its SVD. It then computes and displays the relative error between the original matrix and its rank-rank_approx approximation.

The code also plots the singular values of the matrix and their cumulative sum, as well as the relative error of the rank-r approximation of the matrix as a function of the number of singular values used. The maximum rank used for the plot can be adjusted by modifying the max_rank variable in the code.

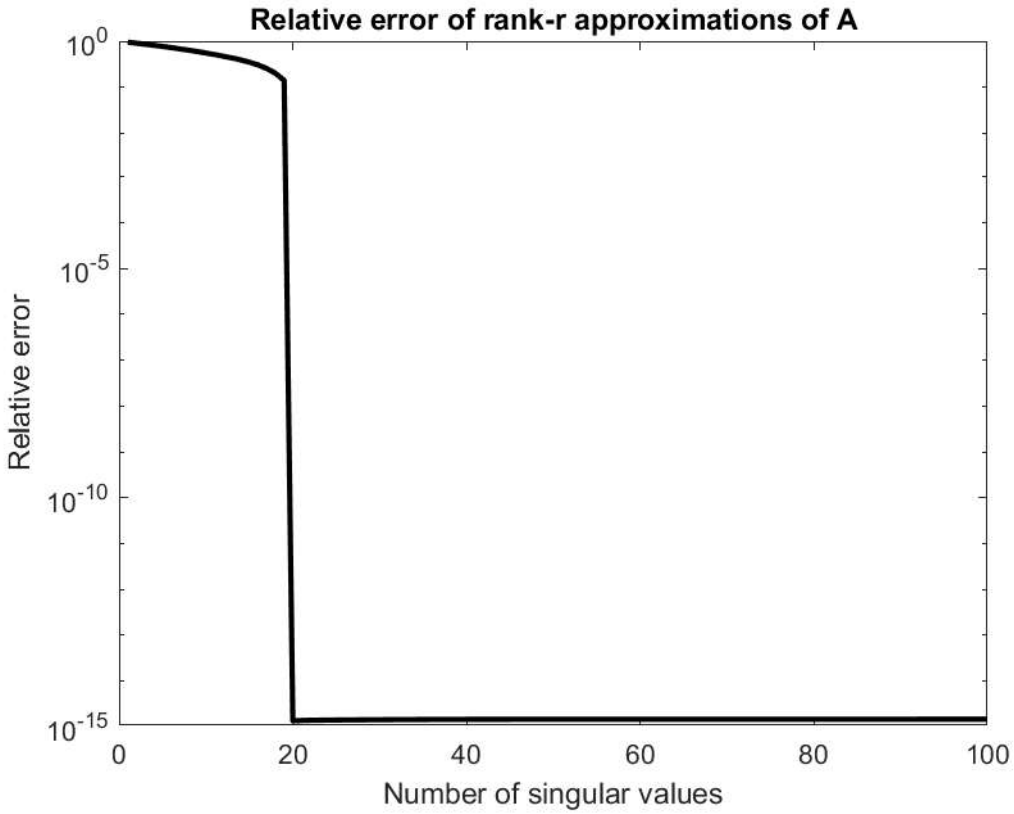
Overall, the code demonstrates the use of the SVD to compute a low-rank approximation of a matrix and how to analyze the quality of the approximation using the relative error and plots of the singular values and their cumulative sum.

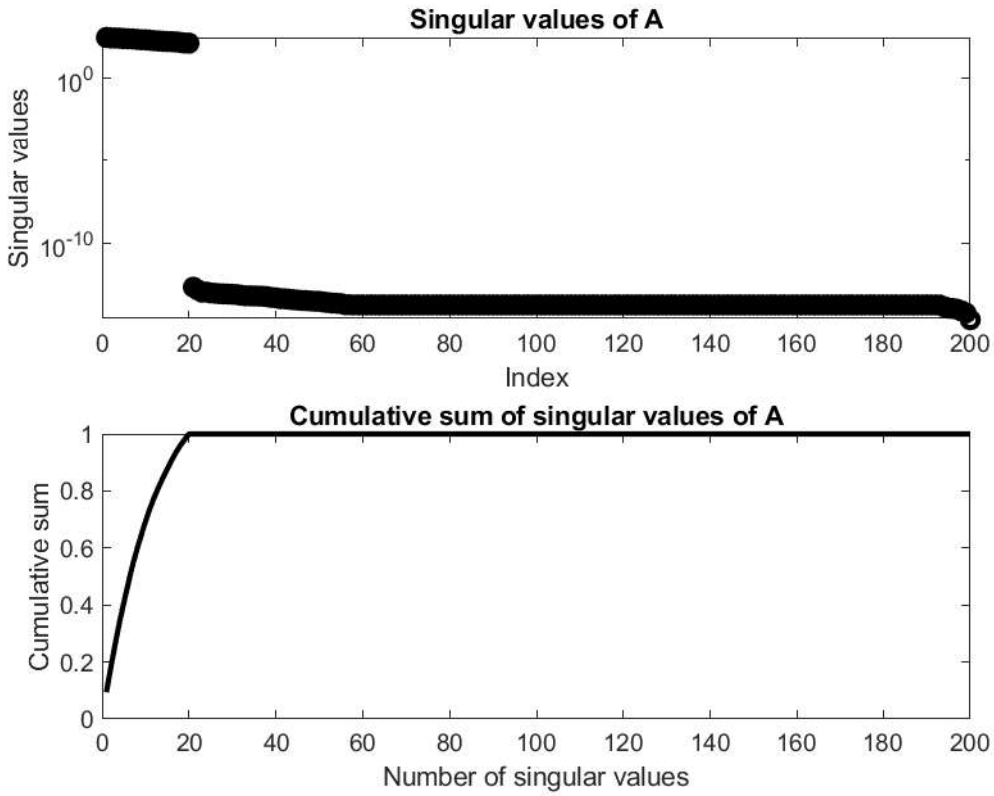
variant 1:n=100 and rank approx=10



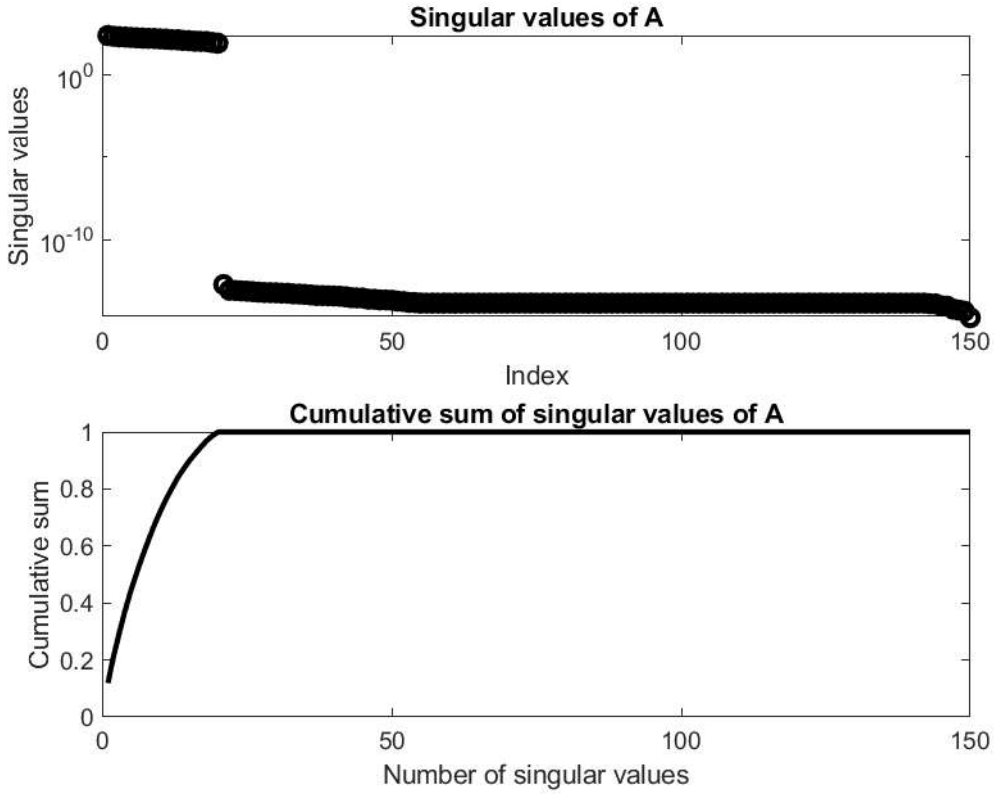

Singular values of A
Singular values
10^{0}
10^{-10}
0
10
20
30
40
50
60
70
80
90
100
Index
Cumulative sum of singular values of A
Cumulative sum
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
60
70
80
90
100
Number of singular values

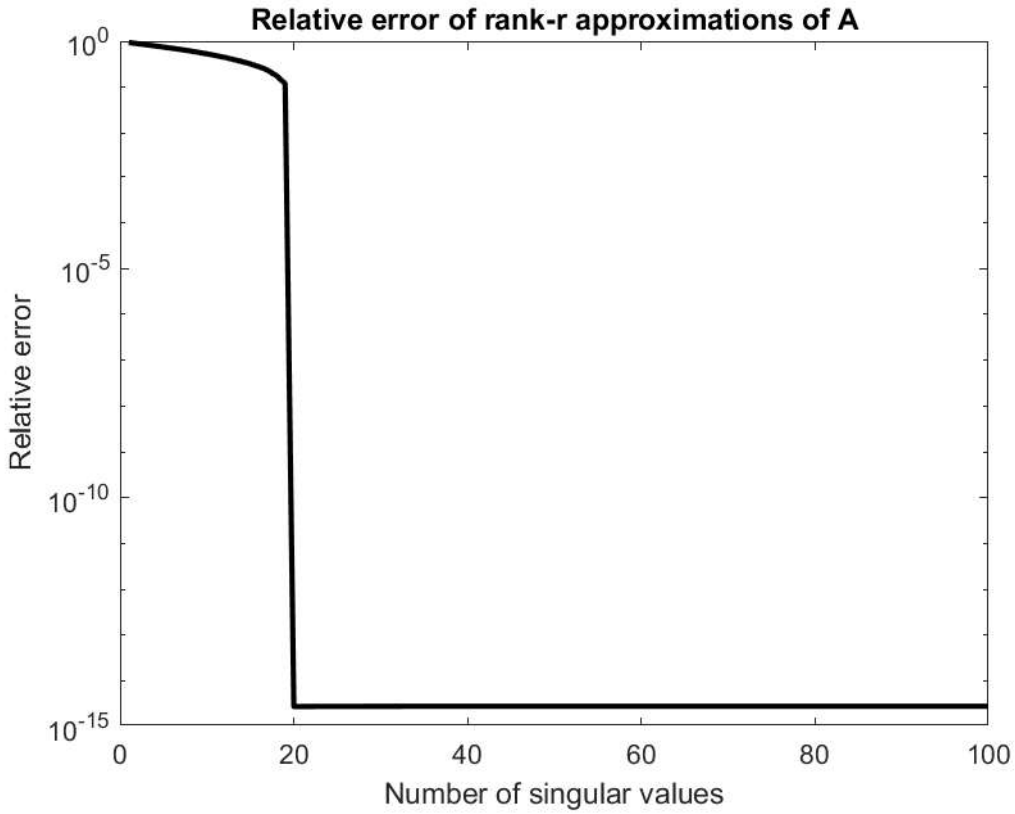
variant 2:n=200 and rank_approx=20



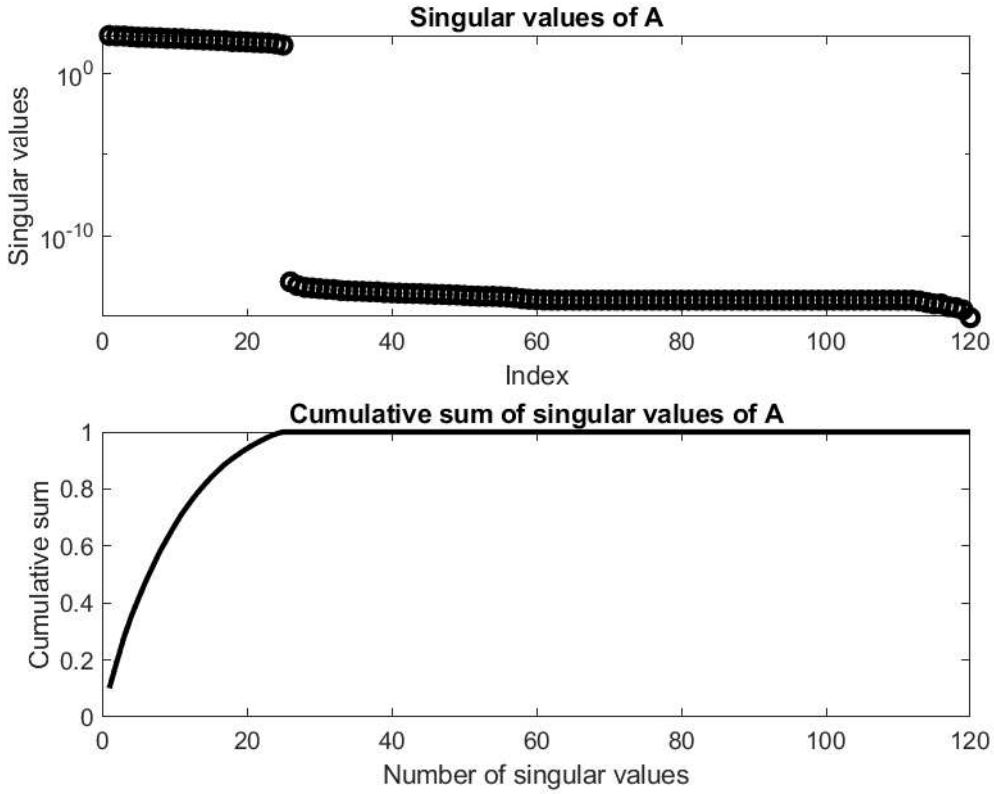

Singular values of A
Singular values
10^{0}
10^{-10}
0
20
40
60
80
100
120
140
160
180
200
Index
Cumulative sum of singular values of A
Cumulative sum
1
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100
120
140
160
180
200
Number of singular values

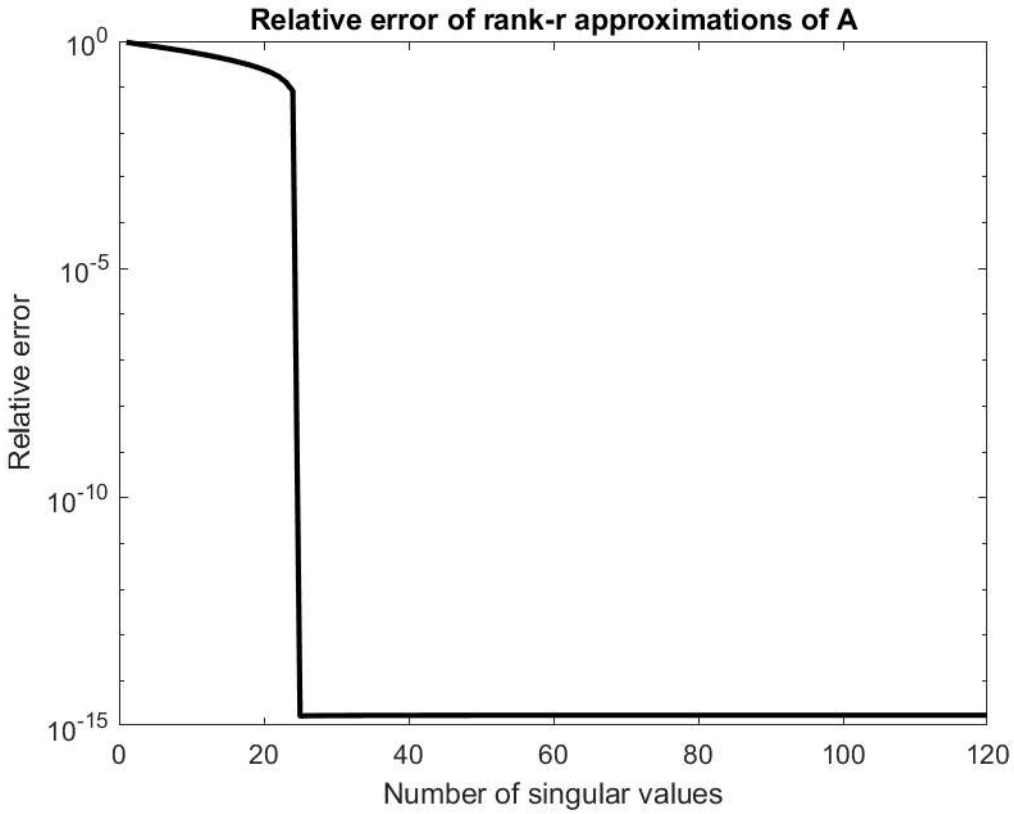
variant 3:n=150 and rank_approx=20



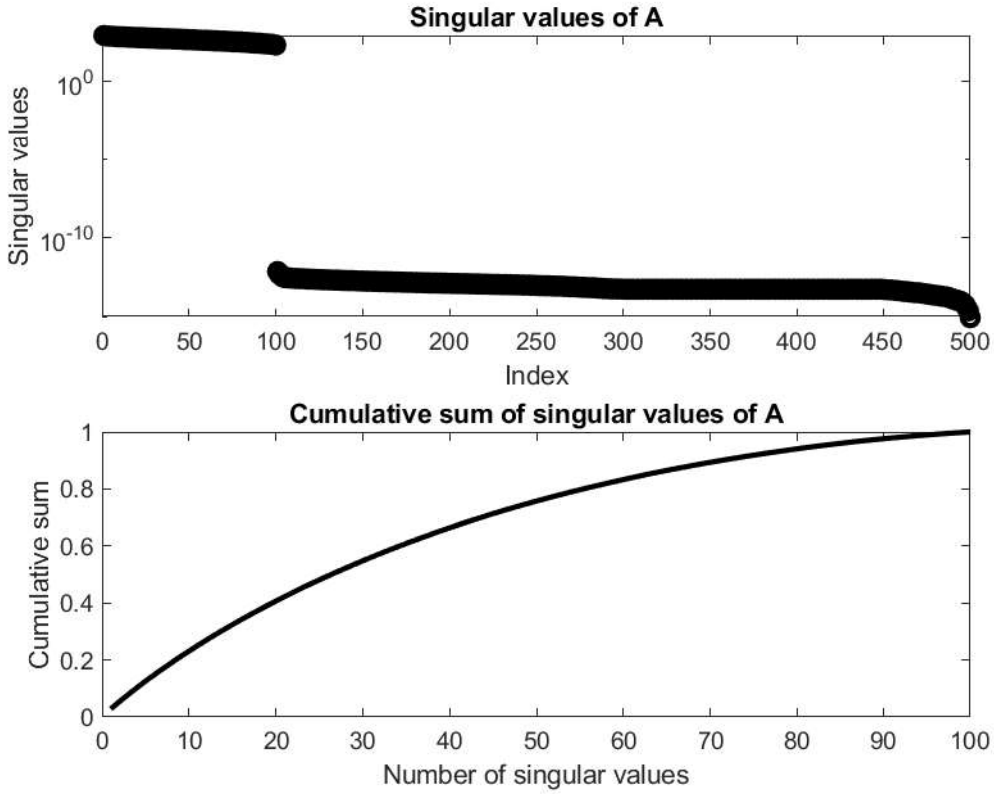

Relative error of rank-r approximations of A
Relative error
Number of singular values
10^0
10^-5
10^-10
10^-15
0
20
40
60
80
100

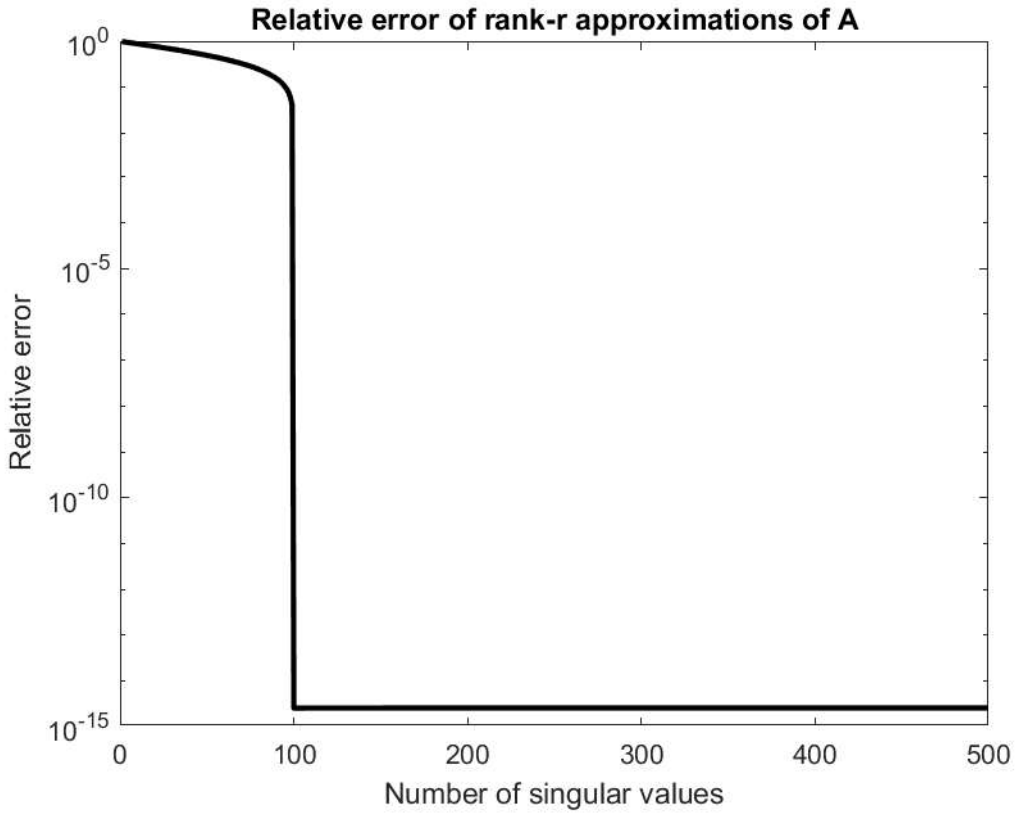
variant 4:n=120 and rank_approx=25




Relative error of rank-r approximations of A
Relative error
Number of singular values
10^0
10^-5
10^-10
10^-15
0
20
40
60
80
100
120

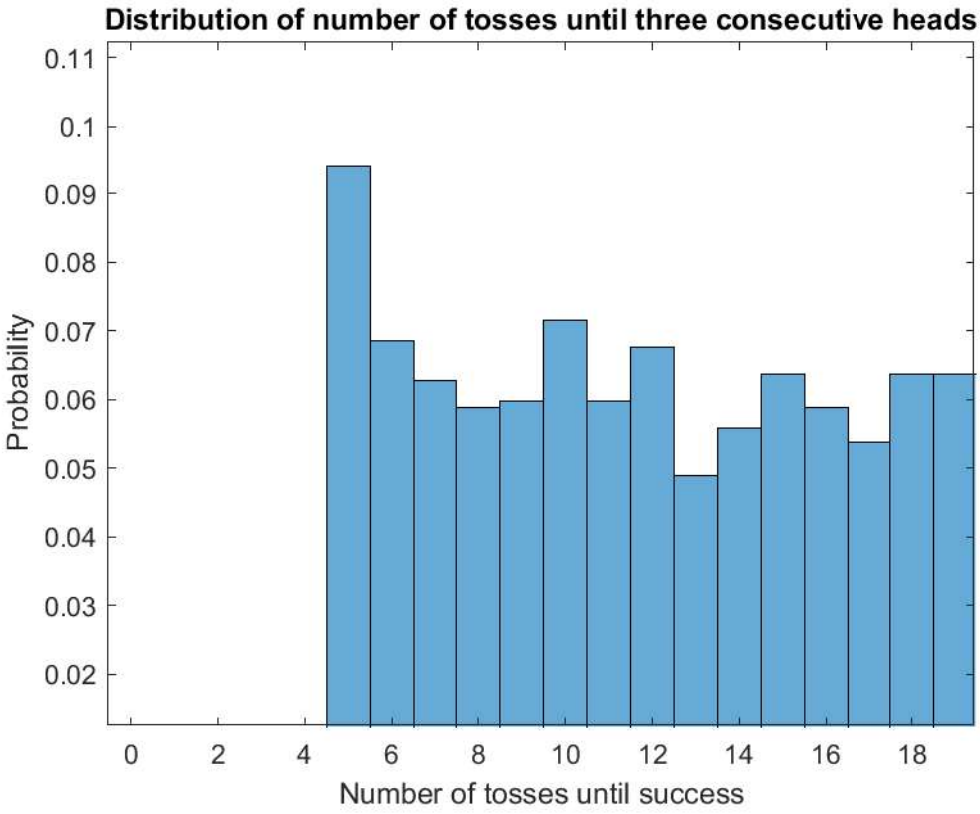
variant 5:n=500 and r=100



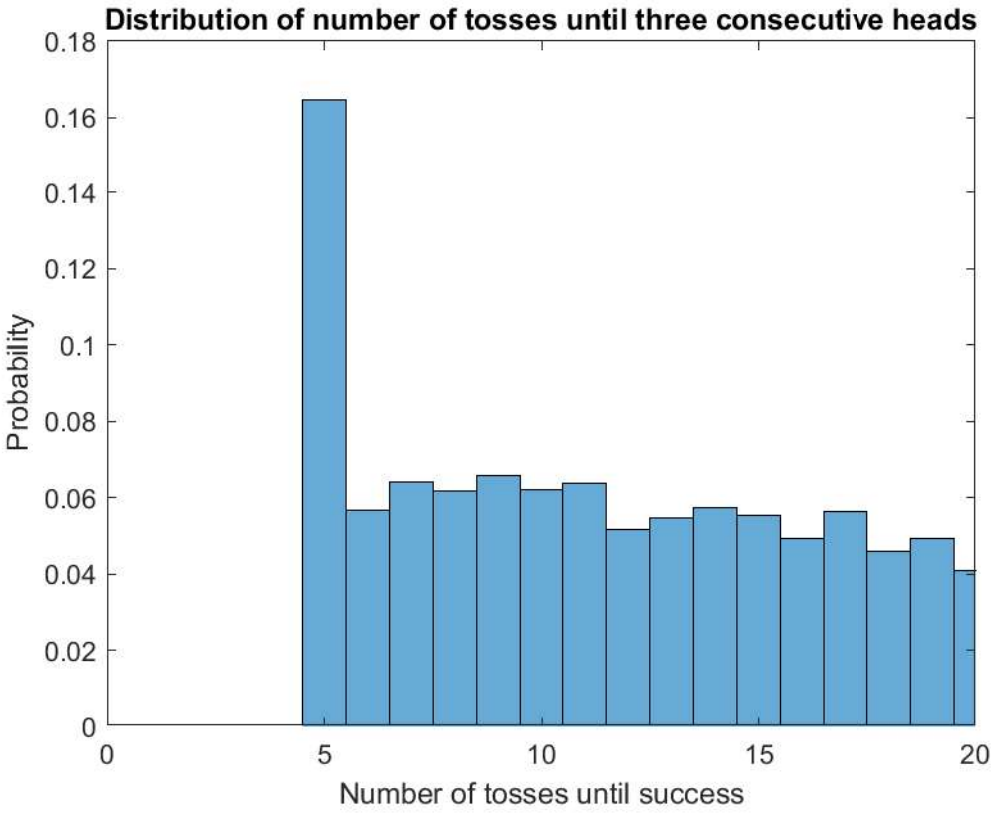

Relative error of rank-r approximations of A
Relative error
Number of singular values
10^0
10^-5
10^-10
10^-15
0
100
200
300
400
500

*The problem involved computing the probability of observing a sequence of five consecutive heads when tossing a biased coin that lands on heads with probability p. The goal was to determine the probability of observing this sequence within the first 20 tosses of the coin.
To solve this problem, we simulated the coin tosses using the rand() function in MATLAB, which generates a uniformly distributed random number between 0 and 1. We considered a value less than p to be heads and a value greater than or equal to p to be tails. We then repeated this simulation many times to estimate the probability of the desired outcome.
The MATLAB code We provided included a for loop that simulated the coin tosses for a specified number of trials. Within each trial, the coin was tossed a maximum of 20 times, and we checked for the occurrence of five consecutive heads after each toss. If five consecutive heads were observed, we recorded the number of tosses required to observe this sequence and moved on to the next trial.
After all trials were completed, we computed the estimated probability of success as the ratio of the number of trials in which the desired sequence was observed within the first 20 tosses to the total number of trials.
The MATLAB code also included plotting of the simulation results using a histogram to visualize the distribution of the number of tosses required to observe the desired sequence. We also printed the estimated probability of success to the console.
Overall, this problem and its solution in MATLAB demonstrate how to use simulation to estimate the probability of a random process and how to visualize the results using MATLAB's built-in plotting functions.

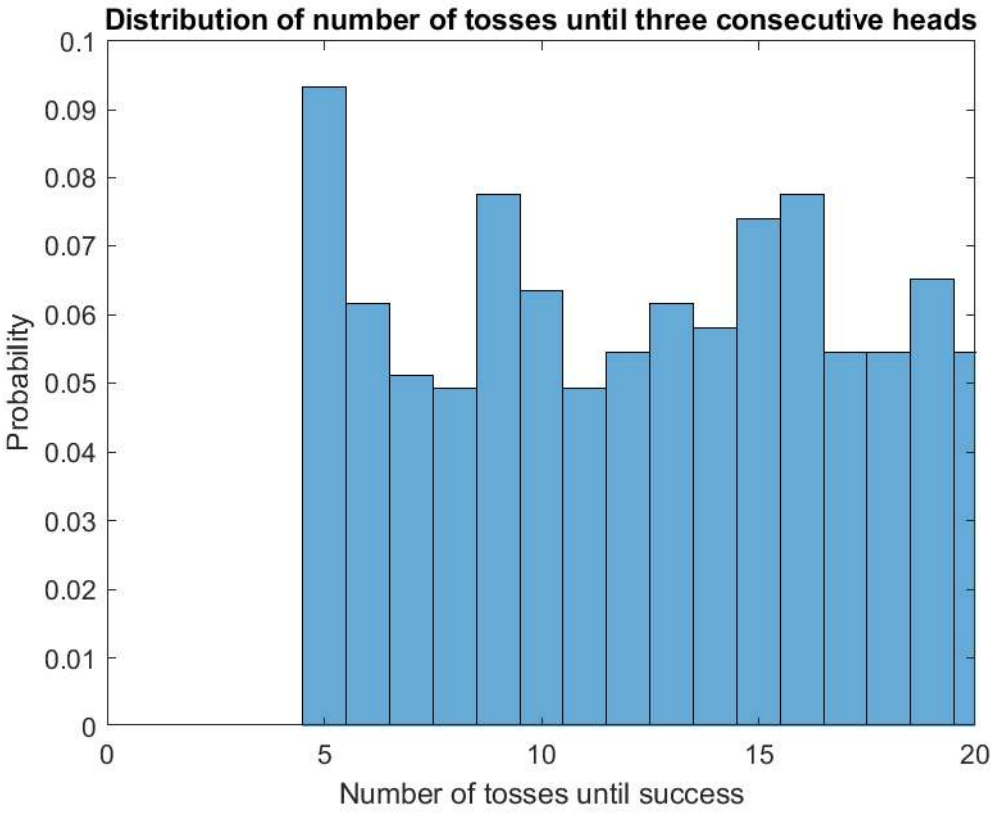
variant 1:p=0.



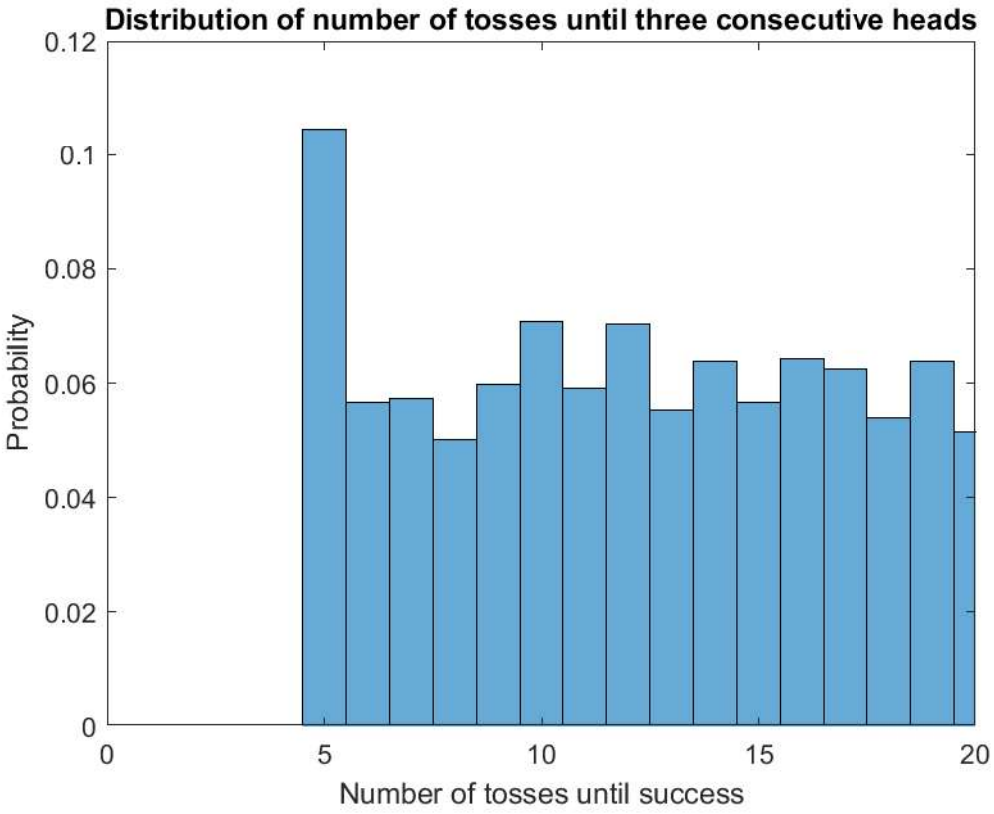
variant 2:p=0.6



variant 3:p=0.35



variant 4:p=0.443



variant 5:p=0.58