SI231B: Matrix Computations, 2024 Fall

Programming Homework-Solution

Acknowledgements:

1) Deadline: **2025-1-8 23:59:59**
2) Please submit the PDF file to gradescope. Course entry code: 8KJ345.
3) Please submit the MATLAB codes with filename YourChineseName_YourStudentID.zip to the link: https://epan.shanghaitech.edu.cn/l/SFrUFZ.
4) You have 5 “free days” in total for all late homework submissions.
5) If your homework is handwritten, please make it clear and legible.
6) All your answers are required to be in English.
7) Please include the main steps in your answer; otherwise, you may not get the points.

Problem 1. (**Cholesky Decomposition, 20 points**)

Let $\mathbf{u} \in \mathbb{R}^n$, $\tau > 0$, and $\mathbf{A} = \mathbf{I}_n + \tau\mathbf{u}\mathbf{u}^T = \mathbf{G}\mathbf{G}^T$ be the Cholesky decomposition of $\mathbf{A}$. Also let $\mathbf{d}$ be an n-dimensional column vector with $\mathbf{d} = \mathrm{diag}(\mathbf{G})$, i.e., $d(i) = \mathbf{G}(i,i)$ for $i = 1, 2, \ldots n$.

Algorithm 1 Diagonalize of the Cholesky factor

```
1: Input: $\mathbf{u} \in \mathbb{R}^n, \tau > 0$.
2: Complete the algorithm here...
3: Output: An $n$-dimensional column vector $\mathbf{d}$ such that $\mathbf{d} = \mathrm{diag}(\mathbf{G})$.
```

1) Determine the vector $\mathbf{d}$ and complete Algorithm 1 while satisfying the following two conditions, respectively.
   (a) Obtain the Cholesky decomposition of $\mathbf{A}$ with $\mathcal{O}(\frac{n^3}{3})$ computational complexity. (**Hint**: you may use the LDL decomposition to obtain the Cholesky decomposition of $\mathbf{A}$) (5 points)
   (b) Determine the vector $\mathbf{d}$ with $\mathcal{O}(n)$ computational complexity. (**Hint**: develop recipes for $d_1 \in \mathbb{R}, \mathbf{v} \in \mathbb{R}^{n-1}$, and the lower triangular $\mathbf{G}_1^{(n-1)\times(n-1)}$ in
$$\mathbf{G}\mathbf{G}^T = \begin{bmatrix} d_1 & \mathbf{0} \\ \mathbf{v} & \mathbf{G}_1 \end{bmatrix} \begin{bmatrix} d_1 & \mathbf{v}^T \\ \mathbf{0} & \mathbf{G}_1^T \end{bmatrix} = \mathbf{I}_n + \tau\mathbf{u}\mathbf{u}^T.)$$
   (7 points)
2) Let $\tau = 2024$ and $\mathbf{u} = [1, 2, 3, 4]^T$. Calculate the n-dimensional column vector $\mathbf{d}$ in **Matlab**. (8 points)

Solution:

1) (a) We derive the Cholesky decomposition via the LDL decompositon.

Algorithm 2 Diagonalize of the Cholesky factor by LDL

```
1: Input: $\mathbf{u} \in \mathbb{R}^n, \tau > 0$.
2: Initialize: $\mathbf{L} = \mathbf{I}_n$, $\hat{\mathbf{d}} = \mathbf{0}_n$, $\mathbf{v} = \mathbf{0}_n$, $\mathbf{A} = \mathbf{I}_n + \tau\mathbf{u}\mathbf{u}^T = \mathbf{G}\mathbf{G}^T$.
3: for $j = 1 : n$ do
4:     for $i = 1 : j-1$ do
5:         $v_i = L_{j,i}\hat{d}_i$.
6:     end for
7:     $v_j = \mathbf{A}_{j,j} - \mathbf{L}_{j,1:j-1} * \mathbf{v}_{1:j-1}$.
8:     $\hat{d}_j = v_j$.
9:     $\mathbf{L}_{j+1:n,j} = (\mathbf{A}_{j+1:n,j} - \mathbf{L}_{j+1:n,1:j-1} * \mathbf{v}_{1:j-1})/v_j$.
10: end for
11: Output: An $n$-dimensional column vector $\mathbf{d} = \sqrt{\hat{\mathbf{d}}}$ such that $\mathbf{d} = \mathrm{diag}(\mathbf{G})$.
```

(b) Comparing blocks in
$$\begin{bmatrix} d_1 & \mathbf{0} \\ \mathbf{v} & \mathbf{G}_1 \end{bmatrix} \begin{bmatrix} d_1 & \mathbf{v}^T \\ \mathbf{0} & \mathbf{G}_1^T \end{bmatrix} = \mathbf{I}_n + \tau\mathbf{u}\mathbf{u}^T = \begin{bmatrix} 1 + \tau u_1 u_1 & \tau u_1 \mathbf{u}_{2:n}^T \\ \tau u_1 \mathbf{u}_{2:n} & \mathbf{I}_n + \tau\mathbf{u}_{2:n}\mathbf{u}_{2:n}^T \end{bmatrix}$$

We see that

$$\begin{aligned} d_1 &= \sqrt{1+\tau u_1^2}, \\ \mathbf{v} &= \frac{\tau u_1}{d_1}\mathbf{u}_{2:n}, \\ \mathbf{G}_1\mathbf{G}_1^T &= \mathbf{I} + \tau \mathbf{u}_{2:n}\mathbf{u}_{2:n}^T - \mathbf{v}\mathbf{v}^T \\ &= \mathbf{I} + \frac{\tau}{1+\tau u_1^2}\mathbf{u}_{2:n}\mathbf{u}_{2:n}^T \\ &= \mathbf{I} + \tau_1 \mathbf{u}_{2:n}\mathbf{u}_{2:n}^T, \end{aligned}$$

where $\tau_1 = \frac{\tau}{1+\tau u_1^2}$. Since we want $\mathbf{d} = [d_1, \mathrm{diag}(\mathbf{G}_1)^T]^T$, we repeat this process on the reduced problem $\mathbf{G}_1\mathbf{G}_1^T = \mathbf{I} + \tau_1 \mathbf{u}_{2:n}\mathbf{u}_{2:n}^T$. Hence, we derive the following Algorithm:

Algorithm 3 Diagonalize of the Cholesky factor

```
1: Input: $\mathbf{u} \in \mathbb{R}^n, \tau > 0$.
2: Initialize: $\mathbf{d} = \mathbf{0}_n$.
3: for $k = 1 : n$ do
4:     $d_k = \sqrt{1+\tau u_k^2}$.
5:     $\tau = \frac{\tau}{1+\tau u_k^2}$.
6: end for
7: Output: An $n$-dimensional column vector $\mathbf{d}$ such that $\mathbf{d} = \mathrm{diag}(\mathbf{G})$.
```

2) Applying Algorithm 2 and 3 yields the same result: $\mathbf{d} = [45, 2.2356, 1.6733, 1.4638]^T$.

Problem 2. (QR decomposition for ill-conditioned matrices, 20 points)

Consider the Vandemonde matrix

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_n \\ z_1^2 & z_2^2 & \cdots & z_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ z_1^{m-1} & z_2^{m-1} & \cdots & z_n^{m-1} \end{bmatrix}$$

which can be ill-conditioned sometimes. When $z_i = 1 + 0.001 * (i-1)$, its condition number is large, and the increase in the dimension n causes significant numerical errors in solving linear system.

In this problem, you are encouraged to explore the decomposition stability of the Gram-Schmidt (CGS), Modified Gram-Schmidt (MGS), Householder Reflections, and Givens Rotations methods against ill-conditioned matrices in this case. For $m = 20$, $n = 2, 3, 4, \ldots, 20$, you are required to conduct the above four methods for obtaining the $\mathbf{QR}$ decomposition of $\mathbf{V}$ and plot two figures:

(1) the relationship between the decomposition accuracy $\|\mathbf{QR} - \mathbf{V}\|_2$ of the four methods and the dimension n; (10 points)

(2) the relationship between the orthogonality error $\|\mathbf{Q^T Q} - \mathbf{I}\|_2$ of the four methods and the dimension n. (For Gram-Schmidt and Modified Gram-Schmidt methods, Q is an $m \times n$ matrix. The orthogonality error is $\|\mathbf{Q^T Q} - \mathbf{I}_n\|_2$. For Householder Reflections and Givens Rotations methods, Q is an m-dimensional square matrix. The orthogonality error is $\|\mathbf{Q^T Q} - \mathbf{I}_m\|_2$.) (10 points)

Solution:

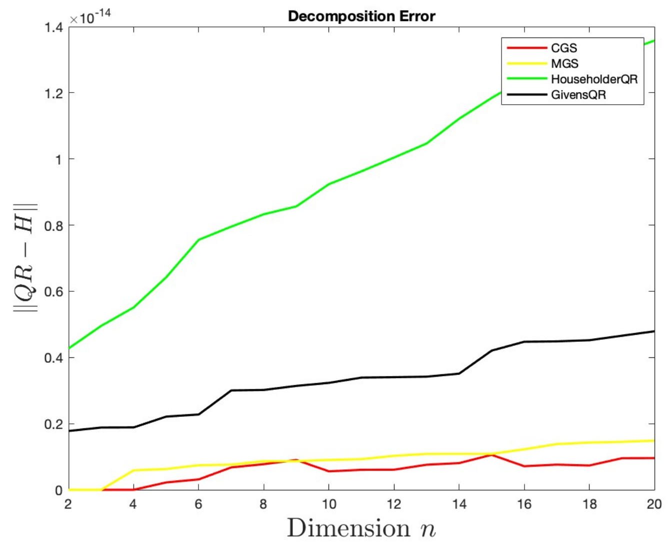


Fig. 1. Figure1. Decomposition error of Gram-Schmidt (CGS), Modified Gram-Schmidt (MGS), Householder Reflections, and Givens Rotations

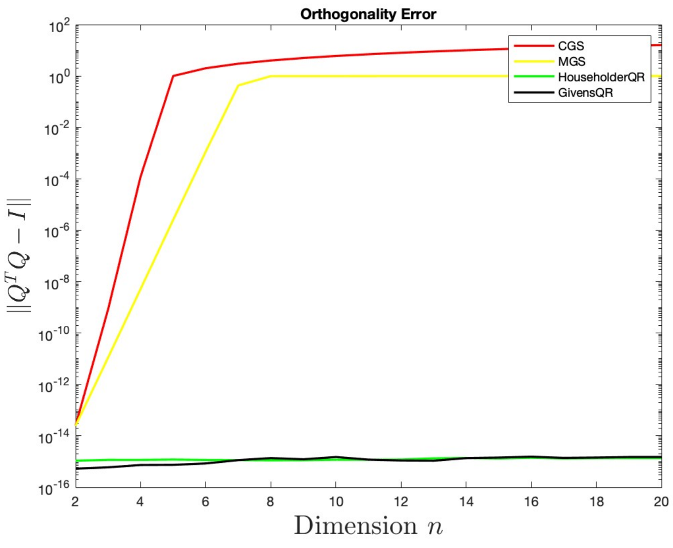


Fig. 2. Figure2. Orthogonal error of Gram-Schmidt (CGS), Modified Gram-Schmidt (MGS), Householder Reflections, and Givens Rotations

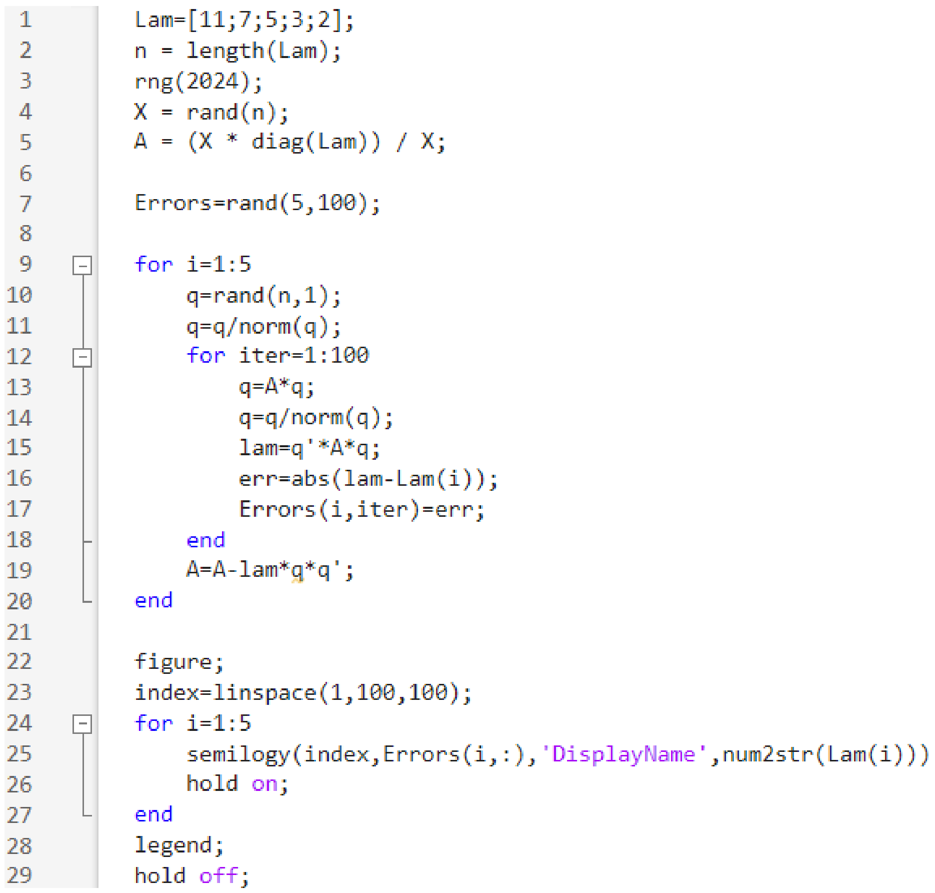
Problem 3. (**Eigendecomposition Algorithms, 20 points**)

1) Construct a square matrix with eigenvalues {2,3,5,7,11}, and then use the power iteration to find the five eigenvalues. Record the error for each iteration and plot the convergence curve (i.e. error of eigenvalue versus iteration). Matlab code is also required. (Hint: Randomly generate a matrix $\mathbf{X}$ and let $\mathbf{A} = \mathbf{X}\Lambda\mathbf{X}^{-1}$. After using the power method to find the largest eigenvalue, modify $\mathbf{A}$ using $\lambda_1\mathbf{v}_1\mathbf{v}_1^T$, where $\mathbf{v}_1$ is the eigenvector corresponding to largest eigenvalue. You can design your own stopping rules; it can be the number of iterations or convergence of eigenvalues) (5 points)
2) Use Rayleigh Quotient iteration to find an eigenvalue of the matrix you construct in question 1). Plot the convergence curve and compare it with the curve generated by power iteration. (Note. You need to plot two curves in one figure.) Matlab code is also required. (5 points)
3) Apply QR iteration, and plot the convergence curve of all eigenvalues. (Use diagonal element of matrix $\mathbf{A}^{(k)}$ during iteration as eigenvalues.) Matlab code is also required. (10 points)

Solution:

1) Code:

   Convergence curve:



```
Lam=[11;7;5;3;2];
n = length(Lam);
rng(2024);
X = rand(n);
A = (X * diag(Lam)) / X;

Errors=rand(5,100);

for i=1:5
    q=rand(n,1);
    q=q/norm(q);
    for iter=1:100
        q=A*q;
        q=q/norm(q);
        lam=q'*A*q;
        err=abs(lam-Lam(i));
        Errors(i,iter)=err;
    end
    A=A-lam*q*q';
end

figure;
index=linspace(1,100,100);
for i=1:5
    semilogy(index,Errors(i,:),'DisplayName',num2str(Lam(i)))
    hold on;
end
legend;
hold off;
```

Fig. 3. Code of question 1).

2) Code:

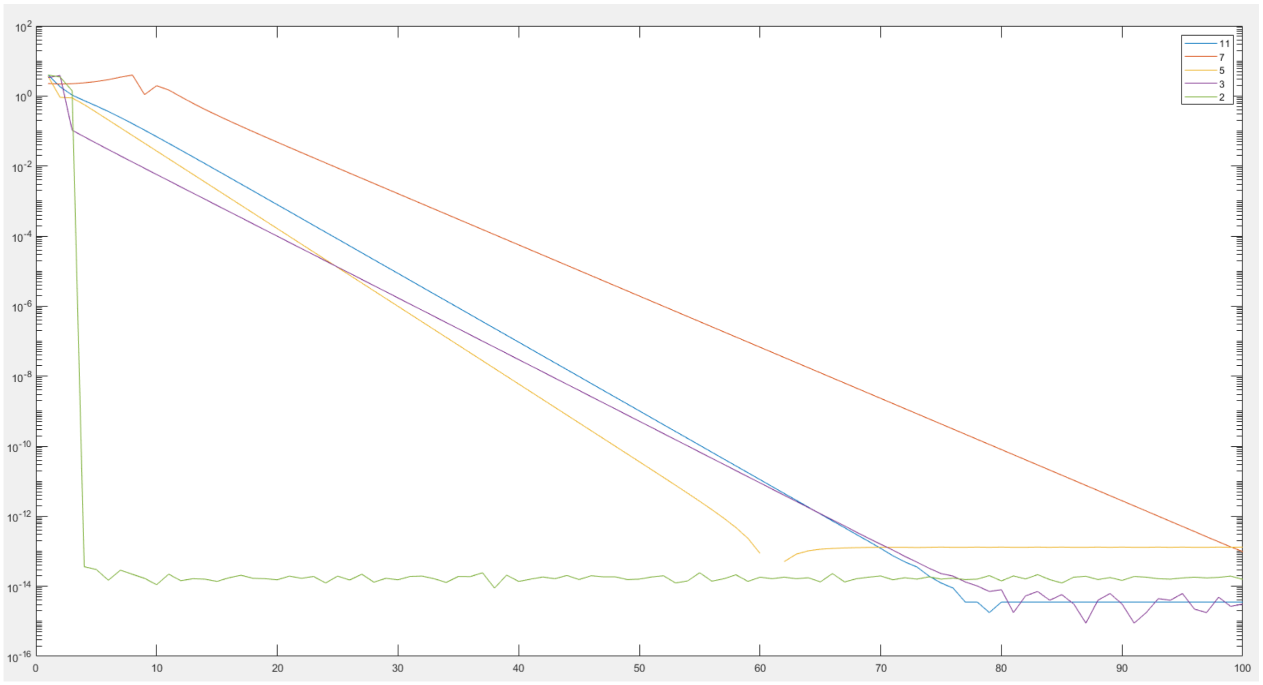
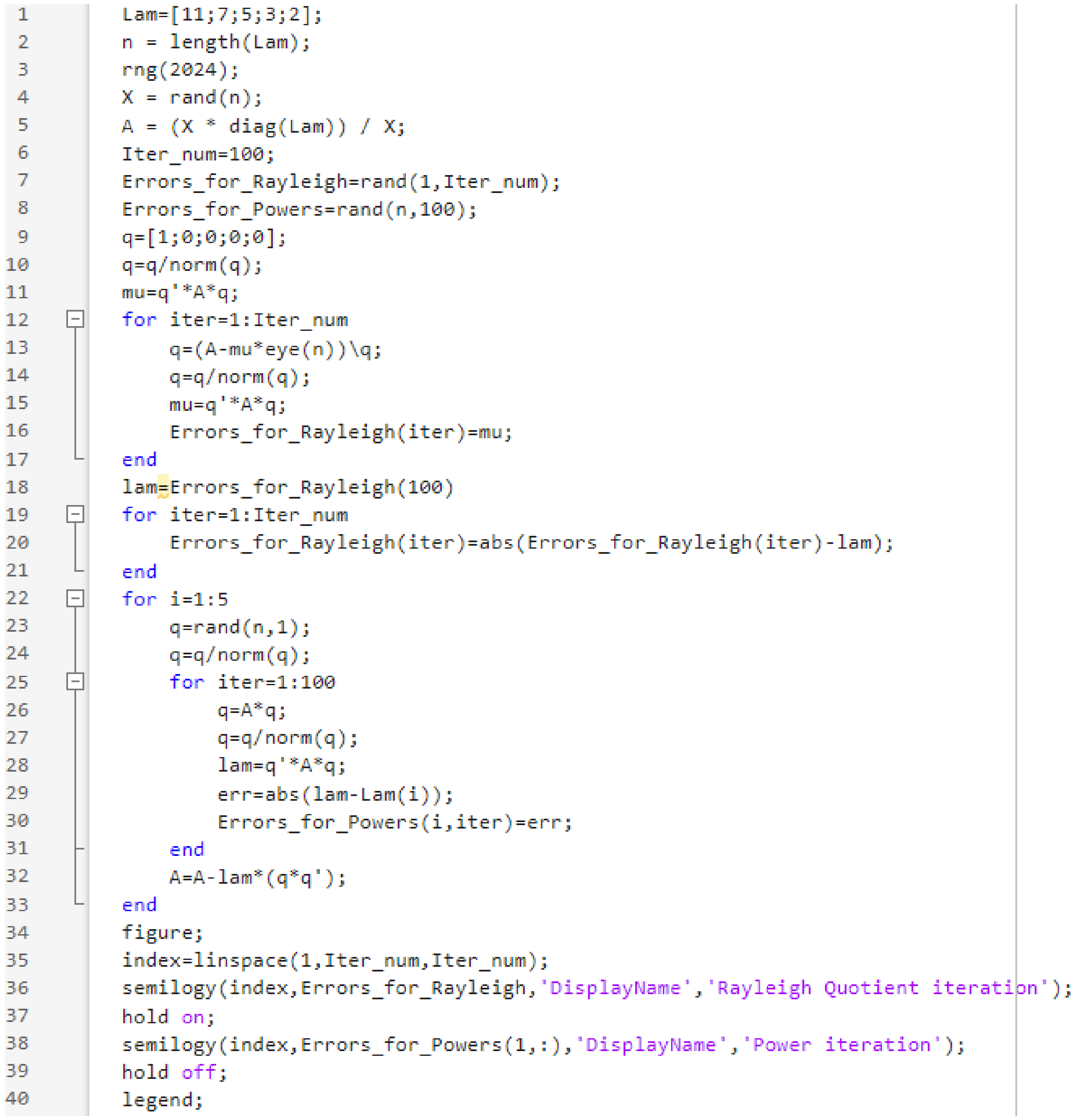


Fig. 4. Convergence curve of question 1). Logarithmic axis is used.

Convergence curve:



```matlab
Lam=[11;7;5;3;2];
n = length(Lam);
rng(2024);
X = rand(n);
A = (X * diag(Lam)) / X;
Iter_num=100;
Errors_for_Rayleigh=rand(1,Iter_num);
Errors_for_Powers=rand(n,100);
q=[1;0;0;0;0];
q=q/norm(q);
mu=q'*A*q;
for iter=1:Iter_num
    q=(A-mu*eye(n))\q;
    q=q/norm(q);
    mu=q'*A*q;
    Errors_for_Rayleigh(iter)=mu;
end
lam=Errors_for_Rayleigh(100)
for iter=1:Iter_num
    Errors_for_Rayleigh(iter)=abs(Errors_for_Rayleigh(iter)-lam);
end
for i=1:5
    q=rand(n,1);
    q=q/norm(q);
    for iter=1:100
        q=A*q;
        q=q/norm(q);
        lam=q'*A*q;
        err=abs(lam-Lam(i));
        Errors_for_Powers(i,iter)=err;
    end
    A=A-lam*(q*q');
end
figure;
index=linspace(1,Iter_num,Iter_num);
semilogy(index,Errors_for_Rayleigh,'DisplayName','Rayleigh Quotient iteration');
hold on;
semilogy(index,Errors_for_Powers(1,:),'DisplayName','Power iteration');
hold off;
legend;
```

Fig. 5. Code of question 2).

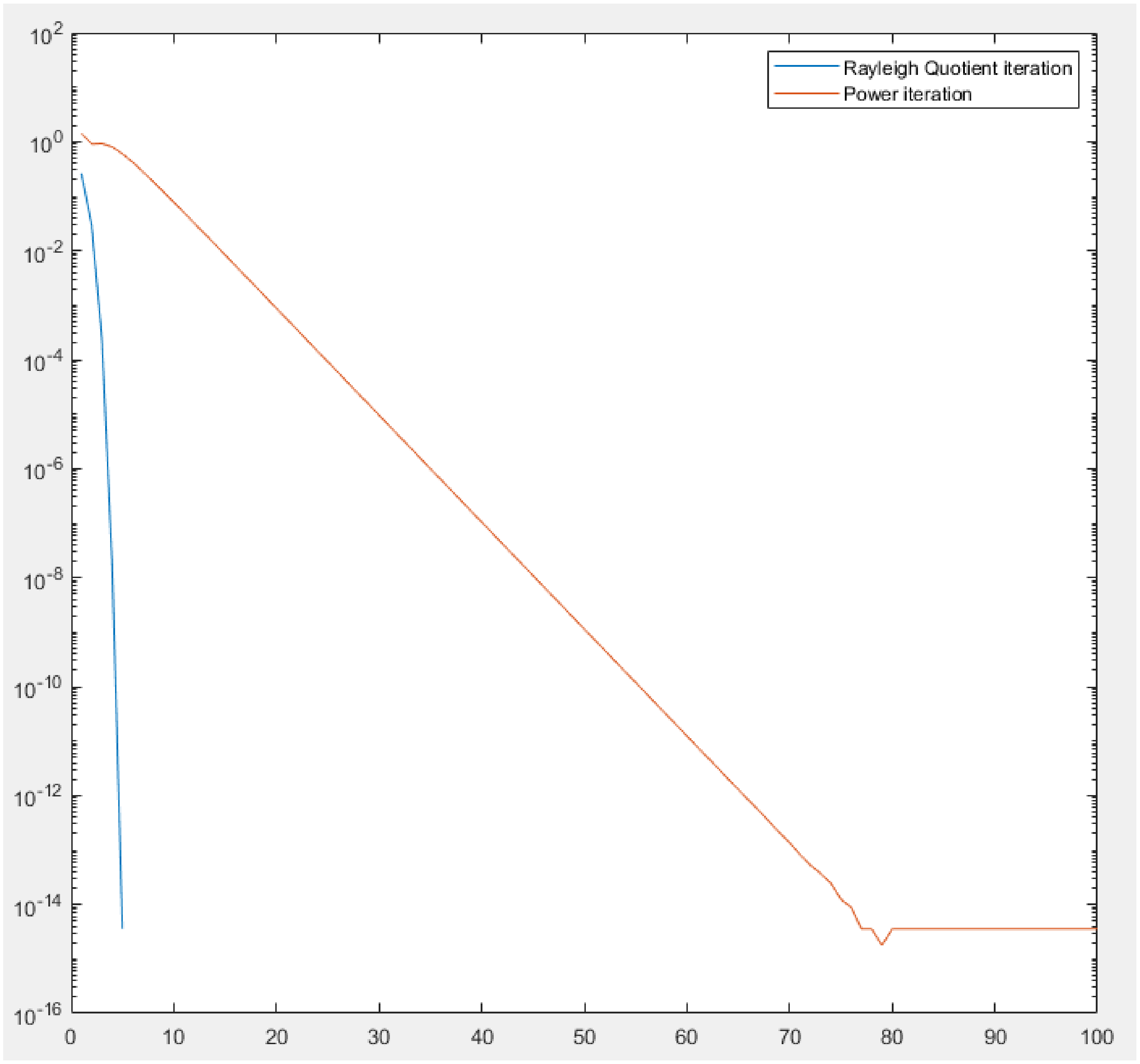
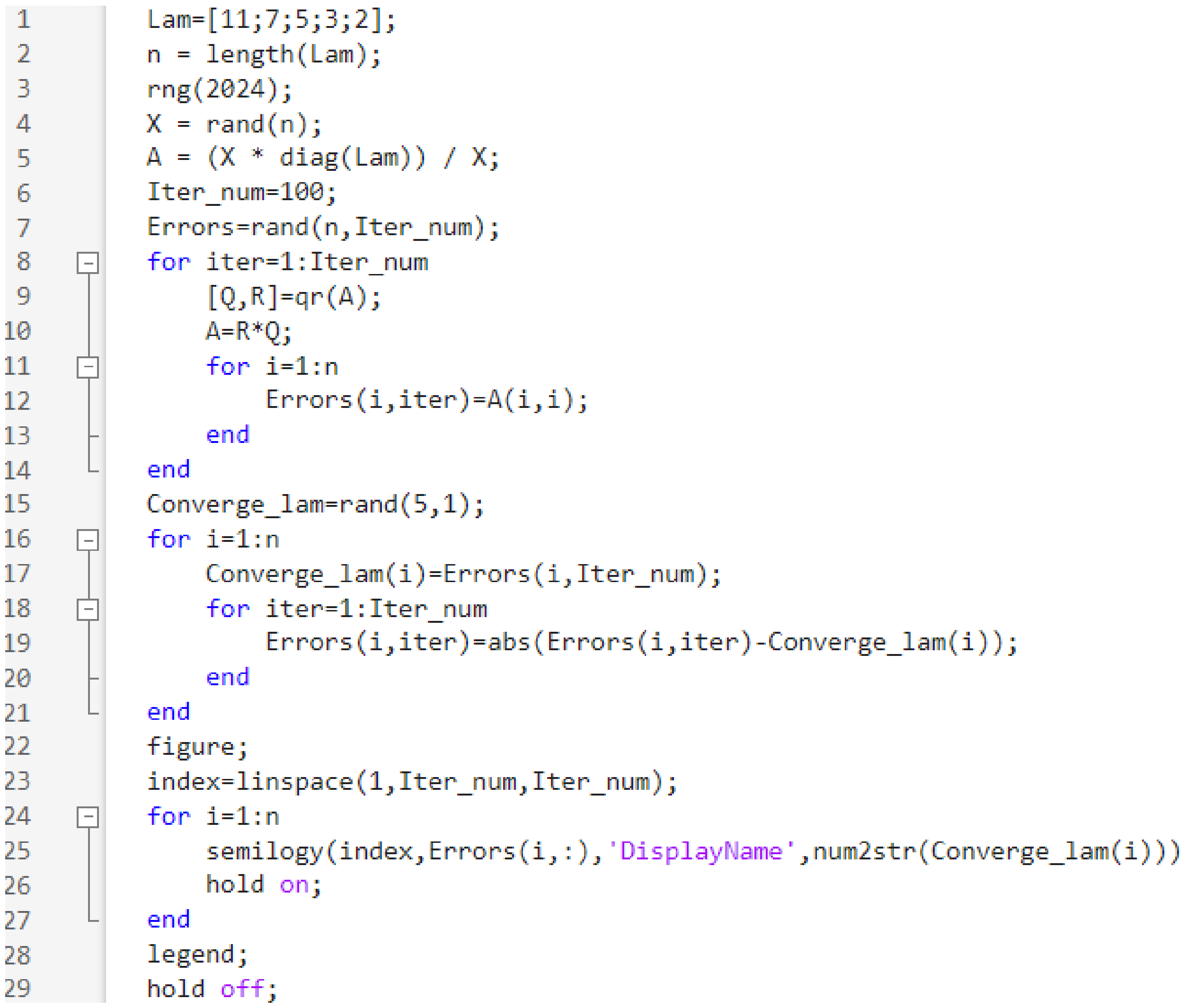


Fig. 6. Convergence curves of question 2). Logarithmic axis is used. With the initial value $\mathbf{q}^{(0)} = [1,0,0,0,0]^T$, Rayleigh Quotient iteration converge to eigenvalue 11.

3) Code:



```
Lam=[11;7;5;3;2];
n = length(Lam);
rng(2024);
X = rand(n);
A = (X * diag(Lam)) / X;
Iter_num=100;
Errors=rand(n,Iter_num);
for iter=1:Iter_num
    [Q,R]=qr(A);
    A=R*Q;
    for i=1:n
        Errors(i,iter)=A(i,i);
    end
end
Converge_lam=rand(5,1);
for i=1:n
    Converge_lam(i)=Errors(i,Iter_num);
    for iter=1:Iter_num
        Errors(i,iter)=abs(Errors(i,iter)-Converge_lam(i));
    end
end
figure;
index=linspace(1,Iter_num,Iter_num);
for i=1:n
    semilogy(index,Errors(i,:),'DisplayName',num2str(Converge_lam(i)))
    hold on;
end
legend;
hold off;
```

Fig. 7. Code of question 3).

Convergence curve:

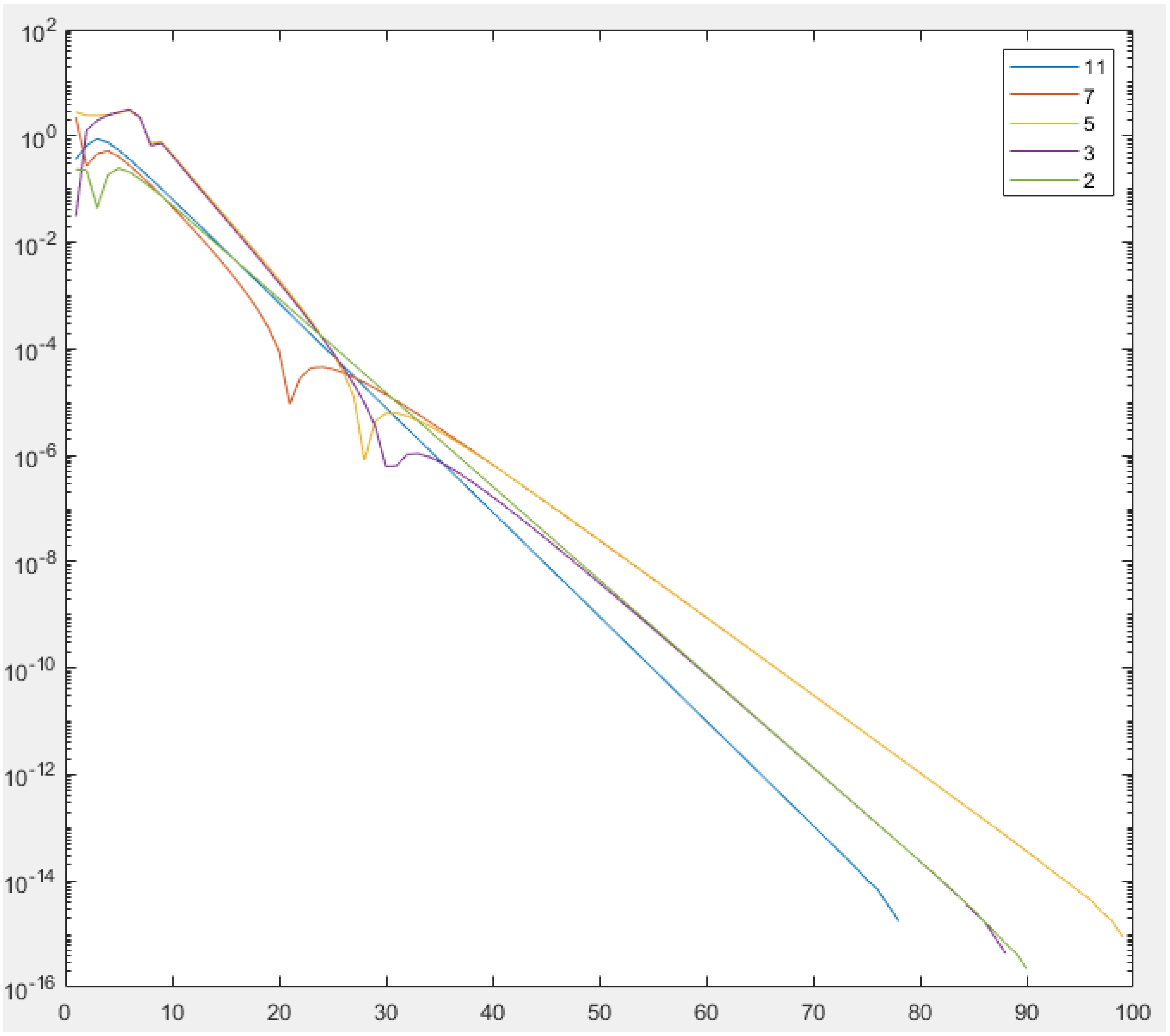


Fig. 8. Convergence curves of question 3). Logarithmic axis is used.

Problem 4. **(SVD, 20 points)**

In this problem, we will see an application of SVD: eigenfaces. The dataset (stored in the file "allFaces.mat") contains 2410 images of faces from 38 persons. Each image is 192×168 (vectorized into a 32256-dimensional face vector). Using SVD, we will embed the high-dimensional data into a low-dimensional space, and perform a simple classification with the embedded data.

1) In the file "eigenfaces.mlx", we have loaded the dataset and plotted the average face. Now perform SVD on mean-subtracted data to get $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{M} \in \mathbb{R}^{32256 \times 2410}$ and each column $\mathbf{m}_i$ is a image subtracting the average face, the main diagonal of $\mathbf{\Sigma}$ consists of $\sigma_1, \cdots, \sigma_{2410}$ with $\sigma_1 > \sigma_2 > \cdots > \sigma_{2410}$. The columns of $\mathbf{U}$ are called "eigenfaces". Plot the first 6 eigenfaces. (5 points)

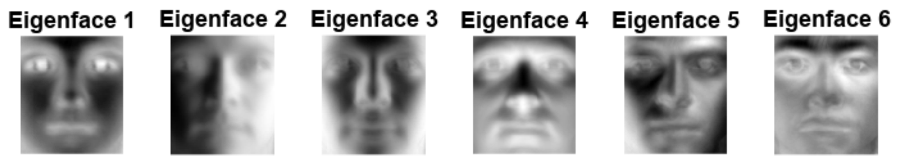
2) In fact, as indicated by the magnitudes of singular values, most of the variance in the dataset is captured by the first few eigenfaces (also called the principle components). For example, $\sum_{i=1}^{150} \sigma_i > 0.5 \sum_{i=1}^{2410} \sigma_i$. Therefore, to compress the dataset, it is reasonable to embed the high-dimensional face data into a rather low-dimensional eigenface space. Let us take the first 50 eigenfaces, and compute the embedded dataset in this $50-$dimensional space. The coordinates of the embedding of image i is given by $\bar{\mathbf{m}}_i = [\langle \mathbf{m}_i, \mathbf{u}_1 \rangle, \cdots, \langle \mathbf{m}_i, \mathbf{u}_{50} \rangle]^T$. Then use the compressed dataset and eigenfaces to recover the average faces of the first 5 persons. Plot these 5 average faces. (5 points)
3) In the eigenface space, images of the same person tend to form a cluster, so we can do a simple classification using the average faces in the last problem. (To see this, you can try to plot the images of the first two persons on a 2-dimensional plane, using the 5th coordinate $\langle \mathbf{m}_i, \mathbf{u}_5 \rangle$ as x-axis and the 6th coordinate $\langle \mathbf{m}_i, \mathbf{u}_6 \rangle$ as y-axis. This plot does not count for points.)
   Specifically, randomly draw 100 samples of the images of the first 5 persons, and we want to classify which person do they belong to. To do this, compute their euclidean distances with the 5 average faces in the r-dimensional eigenface space. The estimated person would be the closest average face. Compute the accuracy rates with different r's and complete the following table. (10 points)

TABLE I
ACCURACY RATES

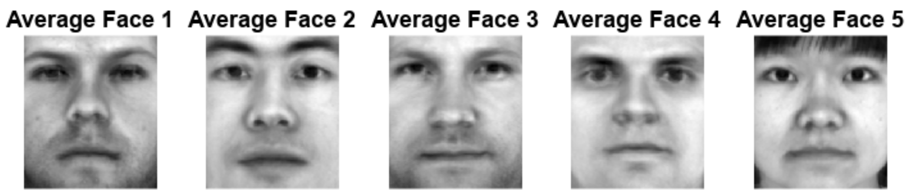
r	6	10	30	80	150
accuracy rate					

Solution:

1) See the reference code.



2) See the reference code.



3) See the reference code.

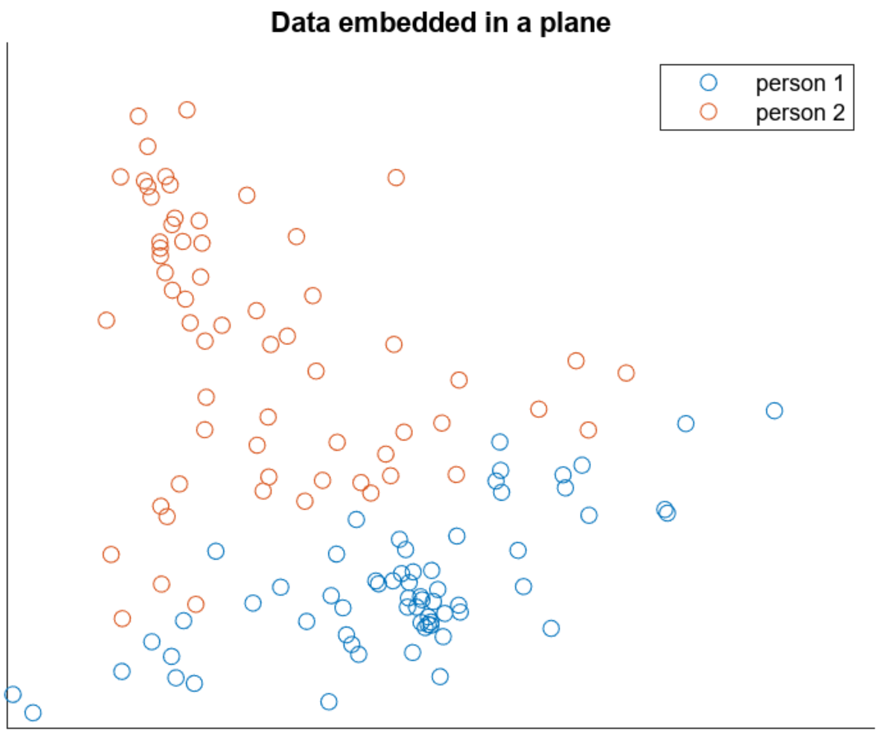


TABLE II

ACCURACY RATES

r	6	10	30	80	150
accuracy rate	33%	44%	53%	69%	66%