HW3

Graded

Student

Manshi Sagar

Total Points

60 / 78 pts

Question 1

Q1

6 / 10 pts

+ 0 pts Incorrect

✔ + 3 pts Part A

+ 2 pts Part A : Minor error

+ 1 pt Part A : Right direction

+ 3 pts Part B

+ 2 pts Part B : Show $AQ_n x = \lambda Q_n x$ but not why $Q_n x \neq 0$

✔ + 1 pt Part B : Mention $AQ_n = Q_n H_n$.

+ 4 pts Part C

+ 3 pts Part C : Minor error (e.g., no justification for $\det(A) \neq 0 \implies \det(H_n) \neq 0$.

✔ + 2 pts Part C : Correct direction

b : Q_n Q_n* is not Identity. So A is not equal to Q_n H_n Q_n*.

c : det(Q_n) isn't even defined.

Question 2

Q2

5 / 10 pts

+ 0 pts Incorrect

✔ + 5 pts Part A : Correct

+ 3 pts Part A : Partially correct

+ 5 pts Part B : Correct

+ 3 pts Part B : Partially correct

b : Your claim is wrong (consider identity 2x2

Question 3

Q3 **15** / 15 pts

- ✔ **+ 2.5 pts** A. Max part
- ✔ **+ 2.5 pts** A. Min part
- ✔ **+ 5 pts** B. Leq part
- ✔ **+ 5 pts** B. geq part

+ 0 pts Incorrect

− 1.5 pts Late

Question 4

Q4 **7** / 10 pts

- ✔ **+ 4 pts** (i) correct

− 1 pt (i) minor errors

+ 1.5 pts (i) correct idea but incomplete proof/errors in proof

+ 0 pts (i) incorrect/unattempted

+ 3 pts (ii) correct

− 0.5 pts (ii) minor issues

− 1.5 pts (ii) Proof uses convexity of W(A) but missing/incorrect proof of convexity of W(A)

+ 1.5 pts (ii) Correct idea, mistakes in proof

- ✔ **+ 0 pts** (ii) incorrect/unattempted
- ✔ **+ 3 pts** (iii) correct

− 0.5 pts (iii) minor issues

+ 1 pt (iii) idea is correct, major mistakes in proof/details missing

+ 0 pts (iii) incorrect/unattempted

Question 5

Q5 **13** / 13 pts

- ✔ **+ 5 pts** Newton Code
- ✔ **+ 5 pts** Broyden Code
- ✔ **+ 3 pts** Comparision

+ 0 pts Incorrect

− 1.3 pts Late

Question 6

Q6

10 / 10 pts

- ✔ **+ 3 pts** Code Part A
- ✔ **+ 3 pts** Code Part B
- ✔ **+ 4 pts** Results Part C

+ 0 pts Incorrect

+ 10 pts Correct

− 1 pt Late

Question 7

Q7

4 / 10 pts

- ✔ **+ 4 pts** Correct code for lanczos

+ 2 pts Correct Code for Ritz values

+ 4 pts Correct Plots for Ritz values

− 2 pts Missing code for plot generation

− 2 pts Matrix A used is different

+ 0 pts Incorrect/Unattempted

1 Where is the code for ritz values? Where is the code for plotting them?

Question assigned to the following page: 1

Q1

(a) $K_n = \langle b, Ab, \ldots A^{n-1}b \rangle$

$= \langle q_1, q_2, \ldots, q_n \rangle$

$K_{n+1} = \langle b, Ab, \ldots, A^{n-1}b, A^n b \rangle$

$= \langle q_1, q_2, \ldots q_n, q_{n+1} \rangle$

we can see that

$K_n \subseteq K_{n+1}$

any vector $v_1 \in K_n$, $v_2 \in K_{n+1}$

$$v_1 = \sum_{i=1}^{n} a_i A^{i-1} b$$

$$v_2 = \sum_{i=1}^{n+1} a_i A^{i-1} b = \underbrace{\sum_{i=1}^{n} a_i A^{i-1} b}_{\text{vector in } K_n} + \underbrace{a_{n+1} A^n b}_{\text{projection along } A^n b}$$

to show that $K_{n+1} = K_n$

it is sufficient to show that $A^n b \in K_n$

$$A^n b = A \cdot \underbrace{\left(A^{n-1} \cdot b\right)}_{\text{in } K_n} = A \cdot \left(\sum_{i=1}^{n} a_i q_i\right) \in A \cdot K_n \quad \text{---(1)}$$

It is sufficient to show $\forall x \in A.K_n$, $x \in K_n$

$\forall x \in A \cdot K_n$

$x = A \cdot \langle q_1, q_2 \ldots q_n \rangle$

$= \langle Aq_1, Aq_2, \ldots Aq_n \rangle$

Now,

$$A\left[q_1 | q_2 | \ldots | q_n\right] = \left[q_1 | q_2 \ldots | q_{n+1}\right] \begin{bmatrix} h_{11} & \cdots & \cdots & h_{1n} \\ h_{21} & \cdots & \cdots & h_{2n} \\ & \ddots & & \vdots \\ & & h_{n-1,n} & h_{nn} \\ & & & 0 \end{bmatrix}$$

$$= \left[q_1 | q_2 | \ldots q_n\right] \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ h_{21} & \ddots & \vdots \\ & \ddots & h_{nn} \end{bmatrix}$$

$$AQ_n = Q_n H_n$$

Question assigned to the following page: 1

this means that each Aq_i can be expressed as a linear combination of $q_1 \ldots q_n$
ie each $Aq_i \in K_n$ ——— (2)

$\Rightarrow \quad x = \langle Aq_1, Aq_2, \ldots Aq_n \rangle$

$\Rightarrow \quad x \in K_n$

$\Rightarrow \quad A^n b \in K_n$

$\Rightarrow \quad K_n = K_{n+1}$

Lets consider K_{n+1} and K_{n+2}

$K_{n+1} = \langle q_1, q_2 \ldots q_n \rangle = K_n$

$K_{n+2} = \langle q_1, q_2, \ldots q_{n+2} \rangle$

$= \langle b, Ab, \ldots A^{n+1} b \rangle$

vector in K_{n+2} = vector in K_{n+1} + $c \cdot A^{n+1} b$

again it is sufficient to prove that $A^{n+1} b \in K_{n+1}$

(to prove $K_{n+1} = K_{n+2}$)

$A^{n+1} b = A\left(\sum_{i=1}^{n} a_i q_i\right) \in A \cdot K_n = K_n$ from (2)

(similar calculation as above)

$\Rightarrow \quad K_{n+1} = K_{n+2} = K_n$

$\Rightarrow$ By induction

$K_n = K_{n+1} = K_{n+2} = \ldots$

Base case & induction step

Question assigned to the following page: 1

Q1 (b) $H_n = Q_n^* A Q_n$

eigen values of $H_n \approx$ eigen values of A

let λ be eigen value of H_n

$\Rightarrow \det(\lambda I - H_n) = 0$

$= \det(Q_n) \cdot \det(\lambda I - H_n) \cdot \det(Q_n^*)$

$= \det(Q_n (\lambda I - H_n) Q_n^*)$

$= \det(\lambda I - Q_n H_n Q_n^*)$

$= \det(\lambda I - A)$

$\Rightarrow \lambda$ is eigenvalue of A

Question assigned to the following page: 1

Q1 (c) A is invertible $\Rightarrow$ $|A| \neq 0$

$\Rightarrow$ $Ax = b \iff x = A^{-1}b$

we have to prove that $x = A^{-1}b$ lies in K_n

$A Q_n = Q_n H_n$ $\Rightarrow \det(H_n) \neq 0$

$\det(A) \neq 0$ $\quad$ $\det(Q_n) \neq 0$

So $Q_n = A Q_n H_n^{-1}$

$Q_n = \left[Aq_1 \mid Aq_2 \mid \cdots \mid Aq_n \right] H_n^{-1}$

$= \left[q_1 \mid q_2 \mid \cdots \mid q_n \right]$

$\Rightarrow$ each q_i is a linear combination of $(Aq_i)_{i=1}^{n}$

$\Rightarrow$ $K_n \in AK_n$

also $AK_n \in K_n$ (from (a))

$\Rightarrow$ $K_n = AK_n$

$b \in K_n$

$\Rightarrow b \in A \cdot K_n$

$\Rightarrow$ b is linear comb^n of $\{Aq_i\}_{i=1}^{n}$

$\Rightarrow$ $A^{-1}b$ is linear combination of $\{q_i\}_{i=1}^{n}$

$\Rightarrow$ $A^{-1}b \in K_n$

Proved !!

Question assigned to the following page: 2

Q2 (a) A, B = symmetric, real

$AB = BA$

By Schur's Theorem

$A = UT_1U^*$

$B = VT_2V^*$ where $T_1, T_2 = [\nabla]$

But A & B are symmetric $T_1 = U^*AU$ $T_2 = V^*BV$

$\Rightarrow$ T_1 and T_2 are also symmetric

$\Rightarrow$ T_1 and T_2 are diagonal $\begin{bmatrix} \lambda & & 0 \\ & \ddots & \\ 0 & & \lambda \end{bmatrix}$

Now we want to prove that $U = V$

let λ is a eigenvalue of A

$\lambda x = Ax$

$A(Bx) = B(Ax) = B(\lambda x) = \lambda(Bx)$

$\Rightarrow$ Bx is an eigenvector of A (with same λ for both x & Bx)

similarly B^2x, B^2x .. are eigenvectors of λ

A has n distinct eigen values (there can be atmost n linearly independent eigenvectors)

$\Rightarrow$ B must be linear map from x to a multiple of x

$\Rightarrow$ $Bx = cx$

$\Rightarrow$ x is eigenvector of B (eigenvalue $= c$)

$\Rightarrow$ x is common eigenvector of A & B (different eigenvalue)

$A = UT_1U^*$

$B = VT_2V^*$

in some Basic U & V, there is one common vector $\frac{x}{\|x\|}$

By Induction,

all eigenvectors of A are also eigenvectors of B

(with different eigenvalue)

$\Rightarrow$ all vectors in Basis U = vectors in Basis V

$\Rightarrow$ $U = V$

But $(T_1 \neq T_2)$

(since A is real symmetric its eigenvectors will be orthogonal)

$\Rightarrow$ $A = UT_1U^*$

$B = VT_2V^* = UT_2U^*$

where T_1 and T_2 are diagonal

$\Rightarrow$ $A = UD_1U^*$

$B = UD_2U^*$ where D_1 and D_2 are diagonal

Question assigned to the following page: 2

Q2 (b) claim – If A is real-symmetric, then A has n distinct eigenvalues.

If this claim holds, then, the required proof is done without the assumption that A has n distinct eigenvalues. (using part (a) for further proof)

Proof The characteristic polynomial of A

$\det(A-\lambda I)=0$ has n roots (real or complex) (distinct or repeated)

each root is an eigenvalue

let $\lambda \in \mathbb{C}$ be any eigenvalue

$Au = \lambda u$

transpose → $u^T A^T = \lambda u^T = u^T A$ (as $A = A^T$)

complex conjugate → $\bar{u}^T A = \bar{\lambda}\bar{u}^T \Rightarrow u^* A = \bar{\lambda} u^*$

$u^* A u = \bar{\lambda} u^* u$

$u^* A u = \lambda u^* u$

$\Rightarrow \boxed{\lambda = \bar{\lambda}}$

$\rightarrow \lambda$ is real

If λ is real, λ cannot be a repeated root of char. poly. (as repeated roots are conjugate of each other)

$\Rightarrow$ eigenvalues of A are distinct

Further proof is same as part(a)

Question assigned to the following page: 3

Q3 (a) $M(T) = \max_{\substack{u \in T \\ |u|=1}} u^T A u$ $\quad N(S) = \min_{\substack{u \in S \\ |u|=1}} u^T A u$

Claim : $\max N(S) = \min M(T) = \lambda_k$

$\max N(S)$: $\dim\ n-k+1$ $\quad$ $\min M(T)$: $\dim\ k$

(i) $\min M(T) = \lambda_k$

$\min \max_{u \in S, |u|=1} u^T A u = \lambda_k$

$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{bmatrix}$

let $x = \sum_{i=1}^{k} c_i u_i$

s.t. $|x| = 1$

$\Rightarrow \sum c_i^2 = 1$

$$x^T A x = \sum_{j=1}^{k} A \lambda_j c_j^2 \leq \lambda_k \sum c_j^2 = \lambda_k$$ (as $\lambda_1 \geq \lambda_2 \geq \dots$)

$$\min \max (u^T A u) \leq \max u^T A u \leq \lambda_k \quad \text{— (1)}$$

Now let $x = \sum_{i=k}^{n} c_i u_i$ s.t $|x| = 1$

$$\Rightarrow \sum_{i=k}^{n} c_i^2 = 1$$

$$x^T A x = \sum_{i=k}^{n} \lambda_i A c_i^2 \geq \lambda_k$$

$$\min\max (u^T A u) \geq \lambda_k \quad \text{— (2)}$$

From (1) and (2)

$$\min\max (u^T A u) = \lambda_k$$

Question assigned to the following page: 3

Q3 (b)

(ii) Similar proof as part (i)

$$\max N(S) = \max_{u \in S, |u|=1} \min u^T A u = \lambda_{n-k+1}$$

$\dim = k$

let $x = \sum_{i=n-k+1}^{n} c_i u_i$

$$x^T A x = \sum_{j=n-k+1}^{n} \lambda_j A c_j^2 \leq \lambda_{n-k+1} \sum c_j^2 \quad (\text{as } \lambda_1 \geq \lambda_2 \geq \cdots)$$

$$\max \min x^T A x \leq \lambda_{n-k+1}$$

similarly

$$\max \min u^T A u \geq \lambda_{n-k+1}$$

$$\Rightarrow \boxed{\max \min u^T A u \geq \lambda_{n-k+1}}$$

From (i) and (ii)

$$\boxed{\lambda_k = \max_S N(S) = \min_T M(T)}$$

Question assigned to the following page: 3

Q3 (b) $A = \begin{bmatrix} H & b^T \\ b & u \end{bmatrix}$ $H: (n-1)\times(n-1)$ symmetric

$\mu_1 \geq \mu_2 \geq \dots \mu_{n-1}$: eigenvalues

Claim $\mu_i \leq \lambda_i$

$\mu_i \geq \lambda_{i+1} \quad \forall i = 1, 2, \dots n-1$

Proof

A is also symmetric

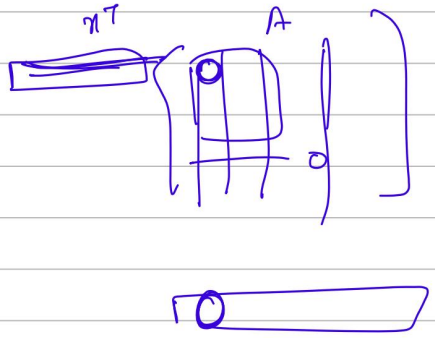
lets calculate eigenvalue of A

$$\lambda_K = \min_{|u|=1} \max x^T A x$$

H is symmetric → its eigen values are real

let $H = V^b D V$

$$A = \begin{bmatrix} V^* D V & b^T \\ b & u \end{bmatrix}$$



Question assigned to the following page: 4

[Q4] (a) $AA^* = A^*A$

$\Rightarrow A$ is normal matrix

By Schur's Theorem

$A = UTU^*$ (T is triangular) //

$A^* = UT^*U^*$

$AA^* = UTU^*\,UT^*U^*$

$= U(TT^*)U^*$

$A^*A = UT^*U^*\,UTV^*$

$= U(T^*T)U^*$

$A^*A = AA^*$

$\Rightarrow \boxed{T^*T = TT^*}$ T is normal //

We need to prove that T is diagonal

Proof By Induction

T is a normal matrix of size $n \times n$

Base case $n=1$ holds true

Induction let $T = n \times n$ be upper triangular and normal and diagonal

let $T' = (n+1) \times (n+1)$ be upper triangular and normal

$$T' = \begin{bmatrix} t & z \\ 0 & T \end{bmatrix} \qquad T'^* = \begin{bmatrix} t^* & 0 \\ z^* & T^* \end{bmatrix} \qquad \begin{matrix} t\ z \\ 0\ T \end{matrix}$$

$$T'T'^* = \begin{bmatrix} |t|^2 + |z|^2 & zT^* \\ Tz^* & TT^* \end{bmatrix}$$

$$T'^*T' = \begin{bmatrix} |t|^2 & t^*z \\ z^*t & T^*T \end{bmatrix}$$

T' is normal $\Rightarrow T'T'^* = T'^*T'$

Question assigned to the following page: 4

$\Rightarrow |z| = 0 \Rightarrow \boxed{z = 0}$

then T' is of the form $\begin{bmatrix} t & 0 \\ 0 & T \end{bmatrix}$

$\Rightarrow$ where T is diagonal

$\Rightarrow$ T' is also diagonal

Hence proved

Q4 (b) $W(A) = \dfrac{u^T A u}{u^T u}$

Rayleigh coefficient $R(q_i) = \dfrac{q_i^T A q_i}{q_i^T q_i} = \lambda_i$

to show that $W(A)$ contains the convex hull of λ_i

it is enough to show

$\alpha \lambda_i + (1-\alpha)\lambda_j \in W(A) \quad \forall i, j$

$= \alpha \dfrac{q_1^T A q_1}{q_1^T q_1} + (1-\alpha) \dfrac{q_2^T A q_2}{q_2^T q_2}$ $\rightarrow$ resolve this to form of $\left(\dfrac{u^T A u}{|u|^2}\right)$

for simplicity, let $|q_1| = 1$, $|q_2| = 1$

$= \alpha(q_1^T A q_1) + (1-\alpha) q_2^T A q_2$

here $q_1 = \dfrac{A^k q_2}{|A^k q_2|}$ (Arnoldi iteration) for some $k > 0$

$= \dfrac{A^k q_2}{c}$

$= \dfrac{\alpha(q_2^T (A^k)^T) \cdot A \cdot (A^k q_2)}{c^2} + (1-\alpha) q_2^T A q_2$

Question assigned to the following page: 4

Q4 © $AA^* = A^*A$

$D = X^*AX$ is diagonal

$W(A) = \{u^*Au \mid |u|=1\}$ ↷ same

$W(D) = W(A)$

$W(D) = \{u^*Du \mid |u|=1\}$ let $u = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

$$[\bar{x}_1\ \bar{x}_2 \cdots \bar{x}_n] \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \sum_{i=1}^{n} \lambda_i (\bar{x}_i)(x_i)$$

$$= \sum_{i=1}^{n} \lambda_i |x_i|^2 \qquad \text{since } |u|=1,\ \sum |x_i|^2 = 1$$

$\Rightarrow$ linear combination of λ_i

$=$ convex hull of λ_i

Question assigned to the following page: 5

COL726 - Homework 3

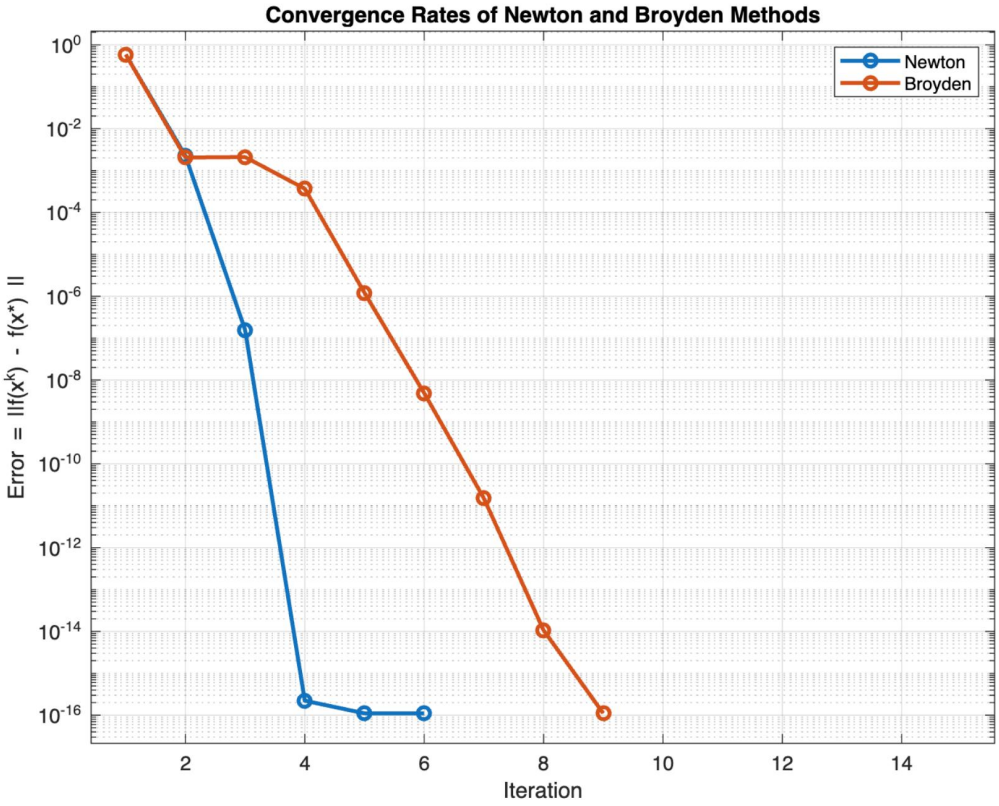
yymanshi

April 2024

1 Q5

```
% Define the functions f and its Jacobian Jf
f = @(x) [(x(1) + 3)*(x(2)^3 - 7)+18; sin(x(2)*exp(x(1)) -1)];
Jf = @(x) [x(2)^3 - 7, 3*(x(1)+3)*x(2)^2; x(2)*exp(x(1))*cos(x(2)*exp(x(1)) - 1), cos(x(2)*exp(x(1

% Initial guess and exact solution
x_init = [-0.5; 1.4];
x_final = [0; 1];

num_iterations = 15;  % Set the number of iterations based on how many you actually perform

% Newton's method
newton_x = x_init;
newton_err = zeros(1, num_iterations);

for i = 1:num_iterations
    delta = -Jf(newton_x)\f(newton_x);
    newton_x = newton_x + delta;
    newton_err(i) = norm(f(newton_x) - f(x_final));
end

% Broyden's method
x = x_init;
B = Jf(x);
b_err = zeros(1, num_iterations);

for i =1:num_iterations
    delta = -B\f(x);
    x_new = x + delta;
    y = f(x_new) -f(x);
    B = B + ((y-B*delta)*delta')/(delta'*delta);
    x = x_new;
    b_err(i) = norm(f(x) - f(x_final));
end
disp(b_err);
disp(newton_err);
% Plot the errors in a graph
figure;
semilogy(1:num_iterations, newton_err(1:num_iterations), 'o-', 'LineWidth', 2); % Ensure this only
hold on;
semilogy(1:num_iterations, b_err(1:num_iterations), 'o-', 'LineWidth', 2);
xlabel('Iteration');
ylabel('Error = ||f(x^k) - f(x*) ||');
title('Convergence Rates of Newton and Broyden Methods');
legend('Newton', 'Broyden');
grid on;  % Adds a grid for better visibility
axis([1 num_iterations min([newton_err b_err]) max([newton_err b_err])]);
```

Questions assigned to the following page: 6 and 5



RESULT:
Both the methods converge. Newton's method converges faster than Broyden's, at around 6 iterations. Broydens converges at 9 iterations. Rate of convergence of Newton's, is faster, as visible from the plot also.
In Newton, we use the exact Jacobian for all calculations, so the results are more accurate and convergence is achieved in fewer iterations.
In Broyden, it is faster as it does not calculate Jacobian for every iteration, and instead approximates it. But due to approximation, accuracy decreases and convergence is delayed.
Newton's method's fast convergence can be because it is highly sensitive to the choice of the initial guess. A good initial guess can lead to very fast convergence.
Broyden's method is good in case of poorer initial guesses but here the initial guess is not that bad, so there no such benefit of Bryiden but there is a slower rate of convergence due to the approximations in the Jacobian approximation.

2 Q6

```
function ass3q6

A1 = [2, -1; 3, 2];
A2 = [-4, 2; -3, 1];

disp('Method (a), infinite series:');
EA1 = matexp_series(A1), EA2 = matexp_series(A2)

disp('Method (b), eigenvalue/eigenvector decomposition:');
EA1 = matexp_ev(A1), EA2 = matexp_ev(A2)

disp('MATLAB expm function:');
```

Question assigned to the following page: 6

```
EA1 = expm(A1), EA2 = expm(A2)

end

function EA = matexp_series(A)
% Compute matrix exponential using series
EA = eye(size(A)); % Start with identity matrix
Ak = eye(size(A)); % A^0
k = 0; % Power of A
kf = 1; % Factorial (k!)
delA = eye(size(A)); % Change added in each step, initialized as identity
tol = 1e-15; % Tolerance

while norm(delA) > tol
    k = k + 1;
    Ak = A * Ak; % Compute A^k
    kf = kf * k; % Compute k!
    delA = Ak / kf; % Change to add in this step
    EA = EA + delA; % Add change to result
end
end

function EA = matexp_ev(A)
% Compute matrix exponential using eigenvalue/eigenvector decomposition
[U, Lambda] = eig(A); % Decompose A
EA = U * diag(exp(diag(Lambda))) / U; % Compute exponential
end
```

OUTPUT:

```
Method (a), infinite series:
EA1 =
   -1.1864   -4.2107
   12.6322   -1.1864

EA2 =
   -0.3298    0.4651
   -0.6976    0.8330

Method (b), eigenvalue/eigenvector decomposition:

EA1 =
  -1.1864 + 0.0000i  -4.2107 + 0.0000i
  12.6322 + 0.0000i  -1.1864 - 0.0000i

EA2 =
   -0.3298    0.4651
   -0.6976    0.8330

MATLAB expm function:
EA1 =
   -1.1864   -4.2107
   12.6322   -1.1864

EA2 =
   -0.3298    0.4651
   -0.6976    0.8330
```

Questions assigned to the following page: 6 and 7

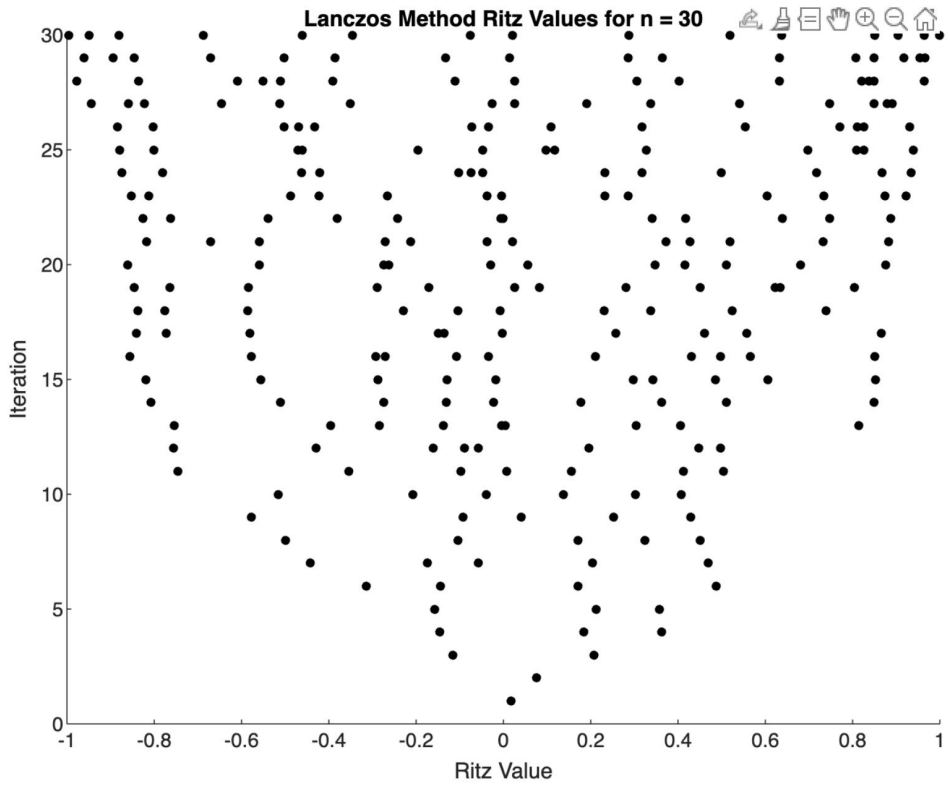
IN This example, both the methods produce similar and accurate results.
In general, For matrices that are easily diagonalizable (e.g., matrices with distinct eigenvalues), method (b) can be computationally efficient.
Not all matrices can be easily diagonalized. Method (A) can be useful here. This method can be applied to any square matrix and is derived directly from the definition of the exponential function. Its is computationally expensive.

3 Q7

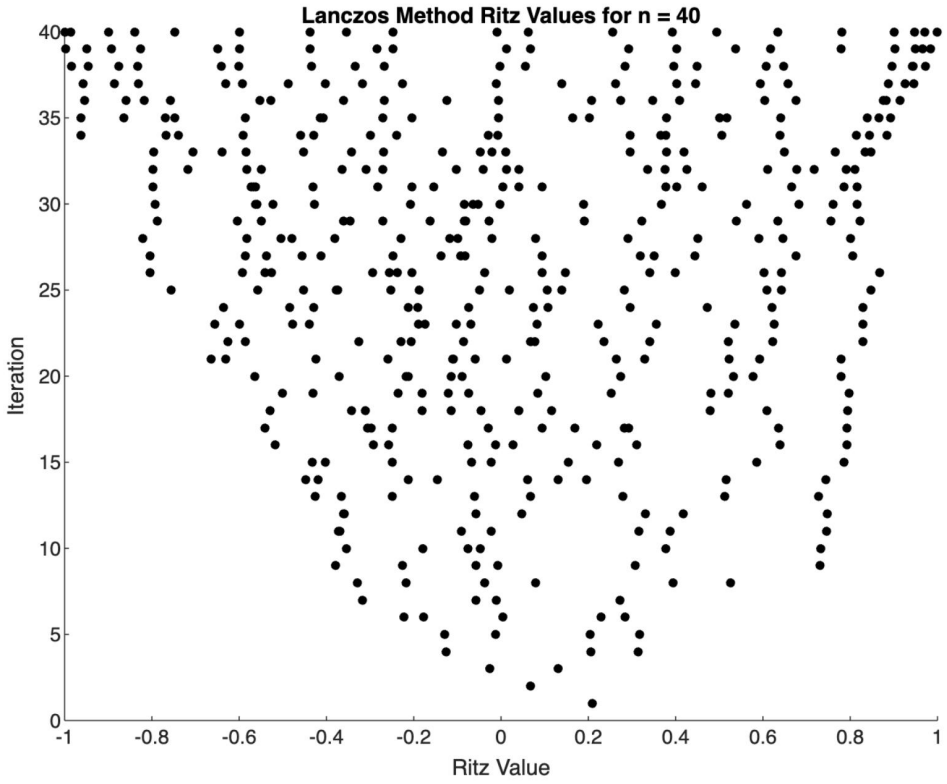
```
function lanczos_test(n)
    % Generate a random n x n matrix B
    B = rand(n);

    % Perform QR factorization of B
    [Q, ~] = qr(B);

    % Construct the matrix A with known eigenvalues
    D = diag(1:n);
    A = Q * D * Q';
    disp('A:');
    disp(A);

    % Lanczos algorithm
    [T, ritzValues] = lanczos(A, n);

    \begin{figure}
        \centering
        \includegraphics[width=0.5\linewidth]{imagen50.png}
        \caption{Enter Caption}
        \label{fig:enter-label}
    \end{figure}
end

function [Q, T] = lanczos(A, n)
    m = length(A);
    Q = zeros(m, n); % Orthogonal vectors
    T = zeros(n, n); % Tridiagonal matrix
    b = rand(m, 1); % Initialize b to a random vector
    q0 = zeros(m, 1); % q0 is initialized to zero vector
    q1 = b / norm(b); % Normalize the initial vector to get q1
    beta = 0;
    for k = 1:n
        v = A * q1;          % Apply matrix A to q1
        alpha = q1' * v;     % Compute the coefficient alpha
        v = v - beta * q0 - alpha * q1;  % Orthogonalize v against q1 and q0
        beta = norm(v);      % Compute the new beta

        if beta < eps       % If beta is close to zero (machine precision)
            break;          % Break if the vector v is nearly zero
        end

        Q(:, k) = q1;       % Store the orthogonalized vector q1 in Q
        q0 = q1;            % Update q0 to be the last q1
        q1 = v / beta;      % Update q1 to the new orthogonalized vector

        % Build the tridiagonal matrix T
        T(k, k) = alpha;
        if k > 1
            T(k, k-1) = beta;  % off-diagonal elements
```

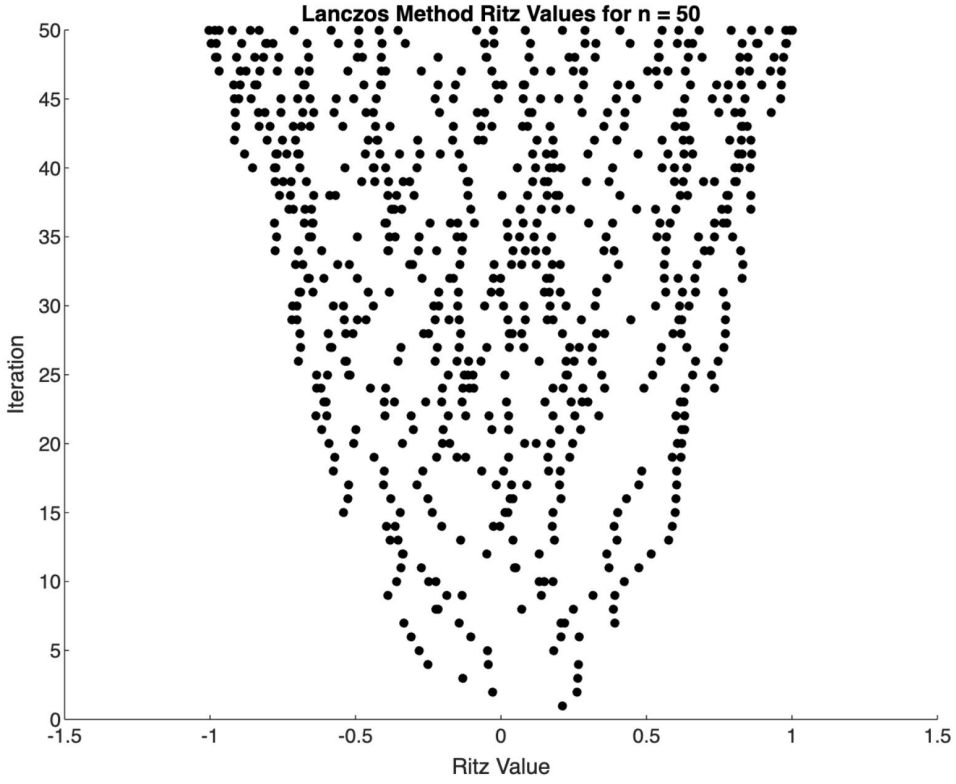
Question assigned to the following page: 7

```
            T(k-1, k) = beta;
        end
    end
end
```



Question assigned to the following page: 7


Lanczos Method Ritz Values for n = 40
Iteration
Ritz Value
0
5
10
15
20
25
30
35
40
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Lanczos Method Ritz Values for n = 50
Iteration
Ritz Value
0
5
10
15
20
25
30
35
40
45
50
-1.5
-1
-0.5
0
0.5
1
1.5