# AE 5311 - Homework 9

James Grisham

4/18/2013

## Problem 1

### Problem Statement

(**a**). Formulate the mass and stiffness matrices for 6-dof system shown below. (Derive by Newton's Law, Lagrange equation, or use the method given in the notes).

(**b**). Show that each column of the following matrix is an eigenvector of the system for the repeated eigenvalue 100.

$$\mathbf{P} = \begin{pmatrix} -.5 & -1.5 & -1 & -2.5 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

### Solution

#### Part A

$$\mathcal{T} = \sum_{i=1}^{N} \frac{1}{2} m_i \dot{v}_i^2 \tag{1}$$

where $N$ is the number of degrees of freedom.

$$V = \sum_{i=1}^{N} \frac{1}{2} k_i \Delta_i{}^2$$

#### Part B

Defining the displacements,

$$v_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{v} = \mathbf{T}\mathbf{m}_1\mathbf{v}$$

$$v_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{v} = \mathbf{T}\mathbf{m}_2\mathbf{v}$$

$$v_3 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{v} = \mathbf{Tm}_3 \mathbf{v}$$
$$v_4 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{v} = \mathbf{Tm}_4 \mathbf{v}$$
$$v_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{v} = \mathbf{Tm}_5 \mathbf{v}$$
$$v_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{v} = \mathbf{Tm}_6 \mathbf{v}$$

Now, (1) can be represented as

$$\mathcal{T} = \sum_{i=1}^{N} \frac{1}{2} m_i \, \dot{\mathbf{v}}^\top (\mathbf{Tm}_i{}^\top \mathbf{Tm}_i) \dot{\mathbf{v}}$$

and the mass matrix is given by

$$\mathbf{M} = \sum_{i=1}^{N} m_i \, (\mathbf{Tm}_i{}^\top \mathbf{Tm}_i) \tag{2}$$

Now, defining the $\Delta$'s,

$$\Delta_1 = v_1 \qquad \Delta_2 = v_2 - v_1 \qquad \Delta_3 = v_3 - v_2$$
$$\Delta_4 = v_4 - v_2 \qquad \Delta_5 = v_5 - v_2 \qquad \Delta_6 = v_6 - v_2$$

Using an approach similar to before,

$$\Delta_1 = \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}^{\mathbf{Tk}_1} \mathbf{v}$$
$$\Delta_2 = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{v}$$
$$\Delta_3 = \begin{bmatrix} 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{v}$$
$$\Delta_4 = \begin{bmatrix} 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{v}$$
$$\Delta_5 = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{v}$$
$$\Delta_6 = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{v}$$

Now, the stiffness matrix is given by

$$\mathbf{K} = \sum_{i=1}^{N} k_i \, (\mathbf{Tk}_i{}^\top \mathbf{Tk}_i)$$

Implementing this in MATLAB,

```
% Summing and calculating K and M matrices
for n = 1:Ndof
    M = M + (mdof(n).m)*transpose(mdof(n).Tm)*(mdof(n).Tm);
    K = K + (mdof(n).k)*transpose(mdof(n).Tk)*(mdof(n).Tk);
end
```

Using MATLAB to perform the calculations:

```
K =
 400     -200        0        0        0        0
-200     1300     -100     -300     -200     -500
   0     -100      100        0        0        0
   0     -300        0      300        0        0
   0     -200        0        0      200        0
   0     -500        0        0        0      500

M =
   4         0        0        0        0        0
   0        10        0        0        0        0
   0         0        1        0        0        0
   0         0        0        3        0        0
   0         0        0        0        2        0
   0         0        0        0        0        5

Eigenvalues:
4.43
100
100
100
100
226

Eigenvectors (unsorted):
   0         0        0        0        0        0
   0         0        0        0    0.642        0
0.23         0    0.609    0.113        0        0
0.4          0        0        0        0        0
0.126        0    0.399        0    0.024    0.144
0.0949       0     0.19     0.19     0.19     0.19
```

### Part B

Numerically proving that

$$\mathbf{P}_i{}^{\top}\mathbf{M}\,\mathbf{P}_j \neq 0$$

Implementing in MATLAB

```
% Testing orthogonality properties for each eigenvector
for I = 1:numel(P(1,:))
    for J = 1:numel(P(1,:))
        if I~=J
            o_prop(n) = P(:,I)'*M*P(:,J);
            fprintf(fid,'P_%d^T * M * P_%d = %d\n',I,J,o_prop(n));
            n = n + 1;
        end
    end
end
```

Output from MATLAB:

```
P_1^T * M * P_2 = 3
P_1^T * M * P_3 = 2
P_1^T * M * P_4 = 5
P_2^T * M * P_1 = 3
P_2^T * M * P_3 = 6
P_2^T * M * P_4 = 15
P_3^T * M * P_1 = 2
P_3^T * M * P_2 = 6
P_3^T * M * P_4 = 10
P_4^T * M * P_1 = 5
P_4^T * M * P_2 = 15
P_4^T * M * P_3 = 10

Clearly, given eigenvectors are not M-orthogonal.
```

To show that combinations of linearly independent eigenvalues are also eigenvalues, pseudo-random numbers are generated and multiplied against the eigenvectors. Then, the resulting linear combination is inserting into the eigenvalue problem to show that the eigenvalue problem is satisfied. Implementing in MATLAB,

$$(\mathbf{K} - \lambda\,\mathbf{M})\mathbf{p}_x = \{0\}$$

```
% Showing that any linear combination of given eigenvectors is also an
% eigenvector
Nel = 6;
Px = zeros(1,Nel);

for I = 1:numel(P(1,:))
    for J = 1:numel(P(1,:))
        if I~=J

            % Generating psuedo-random number
            rand1 = 2*randn;
            rand2 = 3*randn;

            % Computing linear combination of eigenvectors
            Px = rand1*P(:,I) + P(:,J);

            % Computing result to eigenvalue problem
            EigValProb = (K - 100*M)*Px;

            % Checking for "approximately" zero elements
            EigValProb(EigValProb<1e-10) = 0;

            % Printing
            fprintf(fid,'For Px = (%1.2f) * P_%d + (%1.2f) * P_%d\n',...
                rand1,I,rand2,J);
            fprintf(fid,'(K - lambda*M)*Phi =\n');
            fprintf(fid,'%3.3g\n',EigValProb);

        end
    end
end
```

```
For Px = (-0.38) * P_1 + (-0.82) * P_2
(K - lambda*M)*Phi =
  0
  0
  0
  0
  0
  0
For Px = (3.06) * P_1 + (-0.75) * P_3
(K - lambda*M)*Phi =
  0
  0
  0
  0
  0
  0
For Px = (-2.13) * P_1 + (4.81) * P_4
(K - lambda*M)*Phi =
  0
  0
  0
  0
  0
  0
For Px = (2.47) * P_2 + (-0.69) * P_1
(K - lambda*M)*Phi =
  0
  0
  0
  0
  0
  0
For Px = (-3.01) * P_2 + (-1.33) * P_3
(K - lambda*M)*Phi =
  0
  0
  0
  0
  0
  0
For Px = (-0.31) * P_2 + (0.83) * P_4
(K - lambda*M)*Phi =
  0
  0
  0
  0
  0
  0
For Px = (-0.52) * P_3 + (1.33) * P_1
(K - lambda*M)*Phi =
  0
  0
  0
  0
  0
  0
For Px = (0.78) * P_3 + (-3.75) * P_2
(K - lambda*M)*Phi =
```

```
     0
     0
     0
     0
     0
     0
  For Px = (-1.90) * P_3 + (-2.22) * P_4
  (K - lambda*M)*Phi =
     0
     0
     0
     0
     0
     0
  For Px = (-1.02) * P_4 + (-0.96) * P_1
  (K - lambda*M)*Phi =
     0
     0
     0
     0
     0
     0
  For Px = (0.02) * P_4 + (-9.09) * P_2
  (K - lambda*M)*Phi =
     0
     0
     0
     0
     0
     0
  For Px = (-0.91) * P_4 + (3.73) * P_3
  (K - lambda*M)*Phi =
     0
     0
     0
     0
     0
     0
```

---

# Problem 2

## Problem Statement

For a vibration system described by $\mathbf{K}$ and $\mathbf{M}$ matrices, it can be shown that the lowest eigenvalue is greater than $\lambda_L = 1/\text{tr}(\mathbf{A})$, where $\text{tr}(\mathbf{A})$ is the trace of the matrix $\mathbf{A} = \mathbf{K}^{-1}\mathbf{M}$. Demonstrate this bound using the data from Problem 1 (i.e., $\lambda_1 > \lambda_L$).

### Solution

MATLAB implementation:

```
% Defining A matrix
A = K\M;
LamLower = 1/trace(A);

% Printing details
fprintf(fid,'Lambda_1 = %2.2f\n\n',Lambda(1));
fprintf(fid,'Lower Bound: 1/tr(A) = %2.2f\n\n',LamLower);

% Logic to determine if condition is met
if Lambda(1) > LamLower
    fprintf(fid,'%2.2f > %2.2f\n\n',Lambda(1),LamLower);
    fprintf(fid,'Therefore, Lambda_1 > LamLower\n');
else
    fprintf(fid,'%2.2f <= %2.2f\n',Lambda(1),LamLower);
    fprintf(fid,'Condition not met, Lambda_1 <= LamLower\n');
end
```

MATLAB output:

```
Lambda_1 = 4.43

Lower Bound: 1/tr(A) = 3.70

4.43 > 3.70

Therefore, Lambda_1 > LamLower
```

---

## Problem 3

### Problem Statement

For a vibration system described by $\mathbf{K}$ and $\mathbf{M}$ matrices, it can be shown that the lowest eigenvalue is less than $\lambda_U = \min(\mathbf{v}$ where $\mathbf{v}$ is a vector with $i$-th component equal to $\mathbf{K}_{ii}/\mathbf{M}_{ii}$. Demonstrate this bound using the data of Problem 1 (i.e., $\lambda_1 < \lambda_U$).

### Solution

MATLAB implementation:

```
% Determining n-th elements of V
for n = 1:Ndof
    V(n) = K(n,n)/M(n,n);
end

% Finding upper bound
LamUpper = min(V);

% Printing details
fprintf(fid,'Lambda_1 = %2.2f\n\n',Lambda(1));
fprintf(fid,'Upper bound: min(V) = %2.2f\n\n',LamUpper);

% Logic to determine if condition is met
if Lambda(1)<LamUpper
    fprintf(fid,'%2.2f < %2.2f\n\n',Lambda(1),LamUpper);
    fprintf(fid,'Therefore, Lambda_1 < LamUpper\n');
else
    fprintf(fid,'%2.2f >= %2.2f\n',Lambda(1),LamUpper);
    fprintf(fid,'Condition not met, Lambda_1 >= LamUpper\n');
end
```

MATLAB output:

```
Lambda_1 = 4.43

Upper bound: min(V) = 100.00

4.43 < 100.00

Therefore, Lambda_1 < LamUpper
```

---

# Problem 4

## Problem Statement

For a vibration system described by $\mathbf{K}$ and $\mathbf{M}$ matrices, it can be shown that the following quotient

$$RQ = \frac{\mathbf{y}^\top \mathbf{K}\mathbf{y}}{\mathbf{y}^\top \mathbf{M}\mathbf{y}}$$

where

$$\mathbf{y} = \begin{Bmatrix} 1 \\ 2 \\ \vdots \\ N \end{Bmatrix}$$

and where $N$ is the number of degrees of freedom, is an upper bound of the lowest eigenvalue. Demonstrate this bound using the data from Problem 1 (i.e., $\lambda_1 < RQ$).

### Solution

MATLAB implementation:

```
y = (1:Ndof)';
RQ = y'*K*y/(y'*M*y);

% Printing details
fprintf(fid,'Lambda_1 = %2.2f\n\n',Lambda(1));
fprintf(fid,'Upper bound: RQ = %2.2f\n\n',RQ);

% Logic to determine if condition is met
if Lambda(1)<RQ
    fprintf(fid,'%2.2f < %2.2f\n\n',Lambda(1),RQ);
    fprintf(fid,'Therefore, Lambda_1 < RQ\n');
else
    fprintf(fid,'%2.2f >= %2.2f\n',Lambda(1),RQ);
    fprintf(fid,'Condition not met, Lambda_1 >= RQ\n');
end
```

MATLAB output:

```
Lambda_1 = 4.43

Upper bound: RQ = 34.74

4.43 < 34.74

Therefore, Lambda_1 < RQ
```

## Problem 5

### Problem Statement

Use the $\mathbf{K}$ and $\mathbf{M}$ matrices from Problem 1 and the vector

$$\mathbf{x}_1 = \begin{Bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{Bmatrix}$$

For $i = 1$,

$$\mathbf{y}_i = \mathbf{K}^{-1}\mathbf{M}\mathbf{x}_1$$

$$RQ_i = \frac{\mathbf{y}_i^\top \mathbf{K}\mathbf{y}_i}{\mathbf{y}_i^\top \mathbf{M}\mathbf{y}_i}$$

Else, $i > 1$,

$$\mathbf{y}_i = \mathbf{K}^{-1}\mathbf{M}\mathbf{y}_{i-1}$$

$$RQ_i = \frac{\mathbf{y}_i^\top \mathbf{K}\mathbf{y}_i}{\mathbf{y}_i^\top \mathbf{M}\mathbf{y}_i}$$

## Solution

MATLAB implementation:

```
% Defininning number of iterations:
Niterations = 5;

for n = 1:Niterations
    if n == 1
        Y(:,1) = K\(M*Y_initial);
        RQ(1) = Y(:,1)'*K*Y(:,1)/(Y(:,1)'*M*Y(:,1));
        fprintf(fid,'Iteration: %4d \t Lambda_1: \t %1.8f \n',n,RQ(n));
    else
        Y(:,n) = K\(M*Y(:,n-1));
        RQ(n) = Y(:,n)'*K*Y(:,n)/(Y(:,n)'*M*Y(:,n));
        eps = RQ(n) - RQ(n-1);
        fprintf(fid,'Iteration: %4d \t Lambda_1: \t %1.8f \n',n,RQ(n));
        n = n + 1;
    end
end
```

MATLAB output:

```
Iteration:    1      Lambda_1:   4.45855764
Iteration:    2      Lambda_1:   4.43330974
Iteration:    3      Lambda_1:   4.43327812
Iteration:    4      Lambda_1:   4.43327806
Iteration:    5      Lambda_1:   4.43327806
```

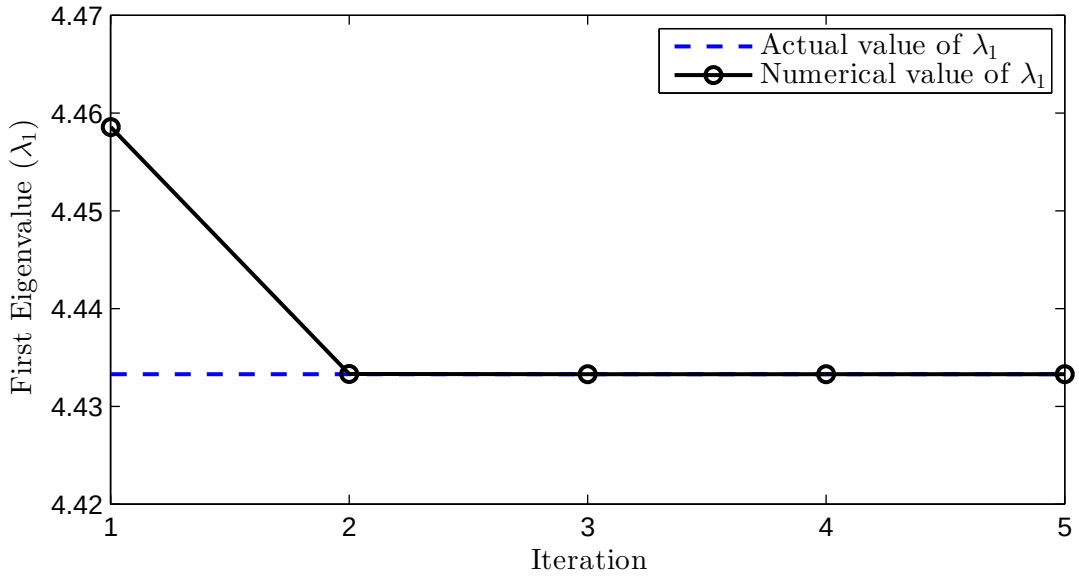


**Figure 1.** Convergence of inverse iteration procedure.

## MATLAB code

```
%--------------------------------------------------------------------------
%% Homework 9
%--------------------------------------------------------------------------
% James Grisham
% AE - 5311

%--------------------------------------------------------------------------
%% Clearing workspace and setting up output files
%--------------------------------------------------------------------------
clc,clear,close all

% Setting defaults
set(0,'defaulttextinterpreter','LaTeX')

% Setting up directory for output files
dataPath = ['C:\Users\James\Desktop\School\Courses\UTA\AE 5311',...
    ' - Structural Dynamics\Data Files\'];

% Setting up path for images
ImgPath = ['C:\Users\James\Desktop\School\Courses\UTA\',...
    'AE 5311 - Structural Dynamics\Images\'];

% Setting up file names for each question and subpart
fid_q1A = 'Question_1_Part_A.txt';
fid_q1B = 'Question_1_Part_B.txt';
fid_q2 = 'Question_2.txt';
fid_q3 = 'Question_3.txt';
fid_q4 = 'Question_4.txt';
fid_q5 = 'Question_5.txt';

% Opening first file
fid = fopen([dataPath,fid_q1A],'w');

%--------------------------------------------------------------------------
%% Problem 1 Part A
%--------------------------------------------------------------------------

% Inputs
m_1 = 4;                                % kg
m_2 = 10;                               % kg
m_3 = 1;                                % kg
m_4 = 3;                                % kg
m_5 = 2;                                % kg
m_6 = 5;                                % kg

k_1 = 200;                              % N/m
k_2 = 200;                              % N/m
k_3 = 100;                              % N/m
k_4 = 300;                              % N/m
k_5 = 200;                              % N/m
k_6 = 500;                              % N/m

% Calculations

% Creating data structure
```

```
Ndof = 6;
mdof = struct('m',[],'k',[],'Tm',[],'Tk',[]);

% Defining Tm_n
mdof(1).Tm = [1 0 0 0 0 0];
mdof(2).Tm = [0 1 0 0 0 0];
mdof(3).Tm = [0 0 1 0 0 0];
mdof(4).Tm = [0 0 0 1 0 0];
mdof(5).Tm = [0 0 0 0 1 0];
mdof(6).Tm = [0 0 0 0 0 1];

% Defining the masses
mdof(1).m = m_1;
mdof(2).m = m_2;
mdof(3).m = m_3;
mdof(4).m = m_4;
mdof(5).m = m_5;
mdof(6).m = m_6;

% Defining stiffnesses
mdof(1).k = k_1;
mdof(2).k = k_2;
mdof(3).k = k_3;
mdof(4).k = k_4;
mdof(5).k = k_5;
mdof(6).k = k_6;

% Defining Tk_n
mdof(1).Tk = [1 0 0 0 0 0];
mdof(2).Tk = [-1 1 0 0 0 0];
mdof(3).Tk = [0 -1 1 0 0 0];
mdof(4).Tk = [0 -1 0 1 0 0];
mdof(5).Tk = [0 -1 0 0 1 0];
mdof(6).Tk = [0 -1 0 0 0 1];

% Preallocating K and M matrices
M = zeros(Ndof,Ndof);
K = M;

% Summing and calculating K and M matrices
for n = 1:Ndof
    M = M + (mdof(n).m)*transpose(mdof(n).Tm)*(mdof(n).Tm);
    K = K + (mdof(n).k)*transpose(mdof(n).Tk)*(mdof(n).Tk);
end

% Finding eigenvectors and eigenvalues
[Phi Lambda] = eig(K,M);

% Sorting eigenvalues
Lambda = sort(diag(Lambda));

% Setting elements less that 1e-10 equal to zero
Phi(Phi<1e-10) = 0;

% Printing results
fprintf(fid,'K = \n');
fprintf(fid,'%4.0f \t %4.0f \t %4.0f \t %4.0f \t %4.0f \t %4.0f\n',K);
fprintf(fid,'\nM = \n');
fprintf(fid,'%3.3g \t %3.3g \t %3.3g \t %3.3g \t %3.3g \t %3.3g\n',M);
```

```
fprintf(fid,'\nEigenvalues:\n');
fprintf(fid,'%3.3g\n',Lambda);
fprintf(fid,'\nEigenvectors (unsorted):\n');
fprintf(fid,'%3.3g \t %3.3g \t %3.3g \t %3.3g \t %3.3g \t %3.3g\n',Phi);

% Closing first file
fclose(fid);

%-------------------------------------------------------------------------
%% Problem 1 Part B
%-------------------------------------------------------------------------

% Opening next file
fid = fopen([dataPath,fid_q1B],'w');

% Given P matrix
P = [-.5 -1.5 -1 -2.5;
    0 0 0 0;
    1 0 0 0;
    0 1 0 0;
    0 0 1 0;
    0 0 0 1];

% Initializing counter variable
n = 1;

% Testing orthogonality properties for each eigenvector
for I = 1:numel(P(1,:))
   for J = 1:numel(P(1,:))
       if I~=J
           o_prop(n) = P(:,I)'*M*P(:,J);
           fprintf(fid,'P_%d^T * M * P_%d = %d\n',I,J,o_prop(n));
           n = n + 1;
       end
   end
end

fprintf(fid,'\nClearly, given eigenvectors are not M-orthogonal.\n\n');

% Showing that any linear combination of given eigenvectors is also an
% eigenvector
Nel = 6;
Px = zeros(1,Nel);

for I = 1:numel(P(1,:))
    for J = 1:numel(P(1,:))
        if I~=J

            % Generating psuedo-random number
            rand1 = 2*randn;
            rand2 = 3*randn;

            % Computing linear combination of eigenvectors
            Px = rand1*P(:,I) + P(:,J);

            % Computing result to eigenvalue problem
            EigValProb = (K - 100*M)*Px;

            % Checking for "approximately" zero elements
```

```
            EigValProb(EigValProb<1e-10) = 0;

            % Printing
            fprintf(fid,'For Px = (%1.2f) * P_%d + (%1.2f) * P_%d\n',...
                rand1,I,rand2,J);
            fprintf(fid,'(K - lambda*M)*Phi =\n');
            fprintf(fid,'%3.3g\n',EigValProb);

        end
    end
end

fclose(fid);

%-------------------------------------------------------------------------
%% Problem 2
%-------------------------------------------------------------------------

fid = fopen([dataPath,fid_q2],'w');

% Defining A matrix
A = K\M;
LamLower = 1/trace(A);

% Printing details
fprintf(fid,'Lambda_1 = %2.2f\n\n',Lambda(1));
fprintf(fid,'Lower Bound: 1/tr(A) = %2.2f\n\n',LamLower);

% Logic to determine if condition is met
if Lambda(1) > LamLower
    fprintf(fid,'%2.2f > %2.2f\n\n',Lambda(1),LamLower);
    fprintf(fid,'Therefore, Lambda_1 > LamLower\n');
else
    fprintf(fid,'%2.2f <= %2.2f\n',Lambda(1),LamLower);
    fprintf(fid,'Condition not met, Lambda_1 <= LamLower\n');
end

% Closing file
fclose(fid);

%-------------------------------------------------------------------------
%% Problem 3
%-------------------------------------------------------------------------

% Opening new file
fid = fopen([dataPath,fid_q3],'w');

% Preallocating V
V = zeros(1,Ndof);

% Determining n-th elements of V
for n = 1:Ndof
    V(n) = K(n,n)/M(n,n);
end

% Finding upper bound
LamUpper = min(V);

% Printing details
```

```
fprintf(fid,'Lambda_1 = %2.2f\n\n',Lambda(1));
fprintf(fid,'Upper bound: min(V) = %2.2f\n\n',LamUpper);

% Logic to determine if condition is met
if Lambda(1)<LamUpper
    fprintf(fid,'%2.2f < %2.2f\n\n',Lambda(1),LamUpper);
    fprintf(fid,'Therefore, Lambda_1 < LamUpper\n');
else
    fprintf(fid,'%2.2f >= %2.2f\n',Lambda(1),LamUpper);
    fprintf(fid,'Condition not met, Lambda_1 >= LamUpper\n');
end

% Closing file
fclose(fid);

%-------------------------------------------------------------------------
%% Problem 4
%-------------------------------------------------------------------------

% Opening new file
fid = fopen([dataPath,fid_q4],'w');

y = (1:Ndof)';
RQ = y'*K*y/(y'*M*y);

% Printing details
fprintf(fid,'Lambda_1 = %2.2f\n\n',Lambda(1));
fprintf(fid,'Upper bound: RQ = %2.2f\n\n',RQ);

% Logic to determine if condition is met
if Lambda(1)<RQ
    fprintf(fid,'%2.2f < %2.2f\n\n',Lambda(1),RQ);
    fprintf(fid,'Therefore, Lambda_1 < RQ\n');
else
    fprintf(fid,'%2.2f >= %2.2f\n',Lambda(1),RQ);
    fprintf(fid,'Condition not met, Lambda_1 >= RQ\n');
end

% Closing file
fclose(fid);

%-------------------------------------------------------------------------
%% Problem 5
%-------------------------------------------------------------------------

fid = fopen([dataPath,fid_q5],'w');

Y_initial = y;

% Initializing counter variable
n = 1;

% Initializing convergence variable
eps = 1;

% Setting up figure
figure('Position',[900 50 650 300])
hold on

```

```
% Defininining number of iterations:
Niterations = 5;

for n = 1:Niterations
    if n == 1
        Y(:,1) = K\(M*Y_initial);
        RQ(1) = Y(:,1)'*K*Y(:,1)/(Y(:,1)'*M*Y(:,1));
        fprintf(fid,'Iteration: %4d \t Lambda_1: \t %1.8f \n',n,RQ(n));
    else
        Y(:,n) = K\(M*Y(:,n-1));
        RQ(n) = Y(:,n)'*K*Y(:,n)/(Y(:,n)'*M*Y(:,n));
        eps = RQ(n) - RQ(n-1);
        fprintf(fid,'Iteration: %4d \t Lambda_1: \t %1.8f \n',n,RQ(n));
        n = n + 1;
    end
end

% Plotting convergence history
plot(1:Niterations,Lambda(1)*ones(1,Niterations),'--b','LineWidth',1.5)
plot(1:Niterations,RQ,'-ok','LineWidth',1.5)

% Legend and labels
lh = legend('Actual value of $\lambda_1$',...
    'Numerical value of $\lambda_1$');
set(lh,'FontSize',11)
set(lh,'Interpreter','LaTeX')
xlabel('Iteration','FontSize',11)
ylabel('First Eigenvalue ($\lambda_1$)','FontSize',11)
set(gca,'box','on')
set(gca,'XTick',1:Niterations)
set(gca,'XLim',[0.999,5])

% Saving plot
set(gcf,'PaperPositionMode','auto')
print(gcf, '-depsc', [ImgPath,'Hw9Prob5.eps'])

fclose(fid);
```