

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.

$$\mathbf{P} = \begin{pmatrix} -0.5 & -1.5 & -1 & -2.5 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Solution

Part A

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

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,

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

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

$$v_3 = [0 \ 0 \ 1 \ 0 \ 0 \ 0] \mathbf{v} = \mathbf{Tm}_3 \mathbf{v}$$

$$v_4 = [0 \ 0 \ 0 \ 1 \ 0 \ 0] \mathbf{v} = \mathbf{Tm}_4 \mathbf{v}$$

$$v_5 = [0 \ 0 \ 0 \ 0 \ 1 \ 0] \mathbf{v} = \mathbf{Tm}_5 \mathbf{v}$$

$$v_6 = [0 \ 0 \ 0 \ 0 \ 0 \ 1] \mathbf{v} = \mathbf{Tm}_6 \mathbf{v}$$

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

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

Now, defining the Δ 's,

$$\begin{array}{lll} \Delta_1 = v_1 & \Delta_2 = v_2 - v_1 & \Delta_3 = v_3 - v_2 \\ \Delta_4 = v_4 - v_2 & \Delta_5 = v_5 - v_2 & \Delta_6 = v_6 - v_2 \end{array}$$

Using an approach similar to before,

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

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
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
0
For Px = (3.06) * P_1 + (-0.75) * P_3
(K - lambda*M)*Phi =
0
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
0
For Px = (2.47) * P_2 + (-0.69) * P_1
(K - lambda*M)*Phi =
0
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
0
For Px = (-0.31) * P_2 + (0.83) * P_4
(K - lambda*M)*Phi =
0
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
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
0
For Px = (-1.02) * P_4 + (-0.96) * P_1
(K - lambda*M)*Phi =
0
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
0
For Px = (-0.91) * P_4 + (3.73) * P_3
(K - lambda*M)*Phi =
0
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\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\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:

```
% Defining 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

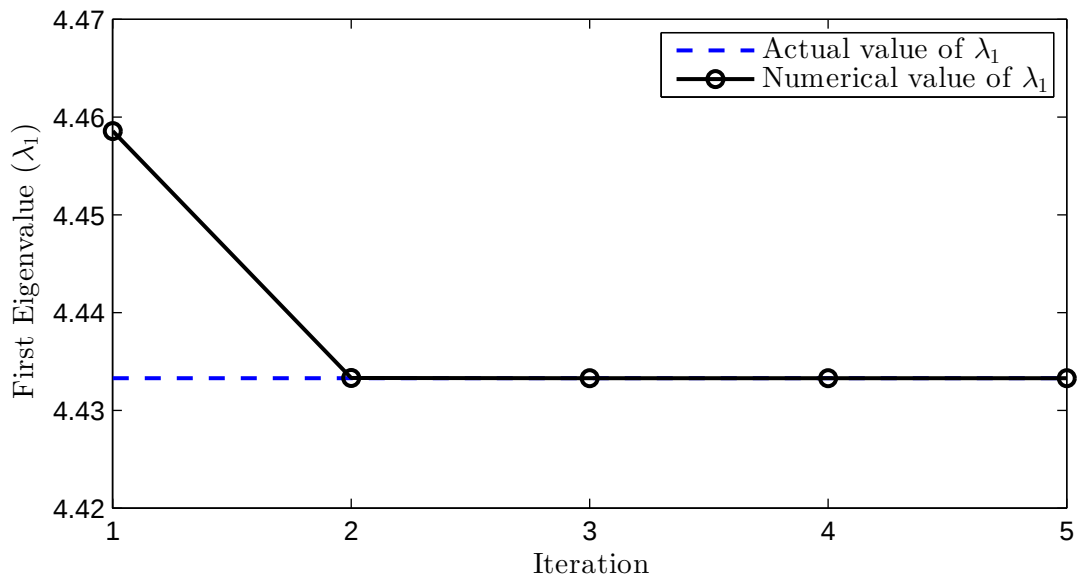


Figure 1. Convergence of inverse iteration procedure.

MATLAB code

```
1  %-----
2  %% Homework 9
3  %-----
4  % James Grisham
5  % AE - 5311
6
7  %-----
8  %% Clearing workspace and setting up output files
9  %-----
10 clc,clear,close all
11
12 % Setting defaults
13 set(0,'defaulttextinterpreter','LaTeX')
14
15 % Setting up directory for output files
16 dataPath = ['C:\Users\James\Desktop\School\Courses\UTA\AE 5311',...
17            '- Structural Dynamics\Data Files\'];
18
19 % Setting up path for images
20 ImgPath = ['C:\Users\James\Desktop\School\Courses\UTA\',...
21           'AE 5311 - Structural Dynamics\Images\'];
22
23 % Setting up file names for each question and subpart
24 fid_q1A = 'Question_1_Part_A.txt';
25 fid_q1B = 'Question_1_Part_B.txt';
26 fid_q2 = 'Question_2.txt';
27 fid_q3 = 'Question_3.txt';
28 fid_q4 = 'Question_4.txt';
29 fid_q5 = 'Question_5.txt';
30
31 % Opening first file
32 fid = fopen([dataPath,fid_q1A],'w');
33
34 %-----
35 %% Problem 1 Part A
36 %-----
37
38 % Inputs
39 m_1 = 4; % kg
40 m_2 = 10; % kg
41 m_3 = 1; % kg
42 m_4 = 3; % kg
43 m_5 = 2; % kg
44 m_6 = 5; % kg
45
46 k_1 = 200; % N/m
47 k_2 = 200; % N/m
48 k_3 = 100; % N/m
49 k_4 = 300; % N/m
50 k_5 = 200; % N/m
51 k_6 = 500; % N/m
52
53 % Calculations
54
55 % Creating data structure
```

```

56 Ndof = 6;
57 mdof = struct('m',[],'k',[],'Tm',[],'Tk',[]);
58
59 % Defining Tm_n
60 mdof(1).Tm = [1 0 0 0 0 0];
61 mdof(2).Tm = [0 1 0 0 0 0];
62 mdof(3).Tm = [0 0 1 0 0 0];
63 mdof(4).Tm = [0 0 0 1 0 0];
64 mdof(5).Tm = [0 0 0 0 1 0];
65 mdof(6).Tm = [0 0 0 0 0 1];
66
67 % Defining the masses
68 mdof(1).m = m_1;
69 mdof(2).m = m_2;
70 mdof(3).m = m_3;
71 mdof(4).m = m_4;
72 mdof(5).m = m_5;
73 mdof(6).m = m_6;
74
75 % Defining stiffnesses
76 mdof(1).k = k_1;
77 mdof(2).k = k_2;
78 mdof(3).k = k_3;
79 mdof(4).k = k_4;
80 mdof(5).k = k_5;
81 mdof(6).k = k_6;
82
83 % Defining Tk_n
84 mdof(1).Tk = [1 0 0 0 0 0];
85 mdof(2).Tk = [-1 1 0 0 0 0];
86 mdof(3).Tk = [0 -1 1 0 0 0];
87 mdof(4).Tk = [0 -1 0 1 0 0];
88 mdof(5).Tk = [0 -1 0 0 1 0];
89 mdof(6).Tk = [0 -1 0 0 0 1];
90
91 % Preallocating K and M matrices
92 M = zeros(Ndof,Ndof);
93 K = M;
94
95 % Summing and calculating K and M matrices
96 for n = 1:Ndof
97     M = M + (mdof(n).m)*transpose(mdof(n).Tm)*(mdof(n).Tm);
98     K = K + (mdof(n).k)*transpose(mdof(n).Tk)*(mdof(n).Tk);
99 end
100
101 % Finding eigenvectors and eigenvalues
102 [Phi Lambda] = eig(K,M);
103
104 % Sorting eigenvalues
105 Lambda = sort(diag(Lambda));
106
107 % Setting elements less than 1e-10 equal to zero
108 Phi(Phi<1e-10) = 0;
109
110 % Printing results
111 fprintf(fid,'K = \n');
112 fprintf(fid,'%4.0f \t %4.0f \t %4.0f \t %4.0f \t %4.0f \t %4.0f\n',K);
113 fprintf(fid,'\nM = \n');
114 fprintf(fid,'%3.3g \t %3.3g \t %3.3g \t %3.3g \t %3.3g \t %3.3g\n',M);

```

```

115 fprintf(fid, '\nEigenvalues:\n');
116 fprintf(fid, '%3.3g\n', Lambda);
117 fprintf(fid, '\nEigenvectors (unsorted):\n');
118 fprintf(fid, '%3.3g \t %3.3g \t %3.3g \t %3.3g \t %3.3g \t %3.3g\n', Phi);
119
120 % Closing first file
121 fclose(fid);
122
123 %-----
124 %% Problem 1 Part B
125 %-----
126
127 % Opening next file
128 fid = fopen([dataPath, fid_q1B], 'w');
129
130 % Given P matrix
131 P = [-.5 -1.5 -1 -2.5;
132      0 0 0 0;
133      1 0 0 0;
134      0 1 0 0;
135      0 0 1 0;
136      0 0 0 1];
137
138 % Initializing counter variable
139 n = 1;
140
141 % Testing orthogonality properties for each eigenvector
142 for I = 1:numel(P(1,:))
143     for J = 1:numel(P(1,:))
144         if I~=J
145             o_prop(n) = P(:,I)'*M*P(:,J);
146             fprintf(fid, 'P_%d^T * M * P_%d = %d\n', I, J, o_prop(n));
147             n = n + 1;
148         end
149     end
150 end
151
152 fprintf(fid, '\nClearly, given eigenvectors are not M-orthogonal.\n\n');
153
154 % Showing that any linear combination of given eigenvectors is also an
155 % eigenvector
156 Nel = 6;
157 Px = zeros(1, Nel);
158
159 for I = 1:numel(P(1,:))
160     for J = 1:numel(P(1,:))
161         if I~=J
162
163             % Generating psuedo-random number
164             rand1 = 2*randn;
165             rand2 = 3*randn;
166
167             % Computing linear combination of eigenvectors
168             Px = rand1*P(:,I) + P(:,J);
169
170             % Computing result to eigenvalue problem
171             EigValProb = (K - 100*M)*Px;
172
173             % Checking for "approximately" zero elements

```

```

174         EigValProb(EigValProb<1e-10) = 0;
175
176         % Printing
177         fprintf(fid, 'For Px = (%1.2f) * P_%d + (%1.2f) * P_%d\n', ...
178             rand1,I,rand2,J);
179         fprintf(fid, '(K - lambda*M)*Phi =\n');
180         fprintf(fid, '%3.3g\n',EigValProb);
181
182     end
183 end
184 end
185
186 fclose(fid);
187
188 %-----
189 %% Problem 2
190 %-----
191
192 fid = fopen([dataPath,fid_q2],'w');
193
194 % Defining A matrix
195 A = K\M;
196 LamLower = 1/trace(A);
197
198 % Printing details
199 fprintf(fid, 'Lambda_1 = %2.2f\n\n', Lambda(1));
200 fprintf(fid, 'Lower Bound: 1/tr(A) = %2.2f\n\n', LamLower);
201
202 % Logic to determine if condition is met
203 if Lambda(1) > LamLower
204     fprintf(fid, '%2.2f > %2.2f\n\n', Lambda(1), LamLower);
205     fprintf(fid, 'Therefore, Lambda_1 > LamLower\n');
206 else
207     fprintf(fid, '%2.2f <= %2.2f\n', Lambda(1), LamLower);
208     fprintf(fid, 'Condition not met, Lambda_1 <= LamLower\n');
209 end
210
211 % Closing file
212 fclose(fid);
213
214 %-----
215 %% Problem 3
216 %-----
217
218 % Opening new file
219 fid = fopen([dataPath,fid_q3],'w');
220
221 % Preallocating V
222 V = zeros(1,Ndof);
223
224 % Determining n-th elements of V
225 for n = 1:Ndof
226     V(n) = K(n,n)/M(n,n);
227 end
228
229 % Finding upper bound
230 LamUpper = min(V);
231
232 % Printing details

```

```

233 fprintf(fid, 'Lambda_1 = %2.2f\n\n', Lambda(1));
234 fprintf(fid, 'Upper bound: min(V) = %2.2f\n\n', LamUpper);
235
236 % Logic to determine if condition is met
237 if Lambda(1) < LamUpper
238     fprintf(fid, '%2.2f < %2.2f\n\n', Lambda(1), LamUpper);
239     fprintf(fid, 'Therefore, Lambda_1 < LamUpper\n');
240 else
241     fprintf(fid, '%2.2f >= %2.2f\n\n', Lambda(1), LamUpper);
242     fprintf(fid, 'Condition not met, Lambda_1 >= LamUpper\n');
243 end
244
245 % Closing file
246 fclose(fid);
247
248 %-----
249 %% Problem 4
250 %-----
251
252 % Opening new file
253 fid = fopen([dataPath, fid_q4], 'w');
254
255 y = (1:Ndof)';
256 RQ = y'*K*y/(y'*M*y);
257
258 % Printing details
259 fprintf(fid, 'Lambda_1 = %2.2f\n\n', Lambda(1));
260 fprintf(fid, 'Upper bound: RQ = %2.2f\n\n', RQ);
261
262 % Logic to determine if condition is met
263 if Lambda(1) < RQ
264     fprintf(fid, '%2.2f < %2.2f\n\n', Lambda(1), RQ);
265     fprintf(fid, 'Therefore, Lambda_1 < RQ\n');
266 else
267     fprintf(fid, '%2.2f >= %2.2f\n\n', Lambda(1), RQ);
268     fprintf(fid, 'Condition not met, Lambda_1 >= RQ\n');
269 end
270
271 % Closing file
272 fclose(fid);
273
274 %-----
275 %% Problem 5
276 %-----
277
278 fid = fopen([dataPath, fid_q5], 'w');
279
280 Y_initial = y;
281
282 % Initializing counter variable
283 n = 1;
284
285 % Initializing convergence variable
286 eps = 1;
287
288 % Setting up figure
289 figure('Position', [900 50 650 300])
290 hold on
291

```

```

292 % Definining number of iterations:
293 Niterations = 5;
294
295 for n = 1:Niterations
296     if n == 1
297         Y(:,1) = K\ (M*Y_initial);
298         RQ(1) = Y(:,1)'*K*Y(:,1)/(Y(:,1)'*M*Y(:,1));
299         fprintf(fid,'Iteration: %4d \t Lambda_1: \t %1.8f \n',n,RQ(n));
300     else
301         Y(:,n) = K\ (M*Y(:,n-1));
302         RQ(n) = Y(:,n)'*K*Y(:,n)/(Y(:,n)'*M*Y(:,n));
303         eps = RQ(n) - RQ(n-1);
304         fprintf(fid,'Iteration: %4d \t Lambda_1: \t %1.8f \n',n,RQ(n));
305         n = n + 1;
306     end
307 end
308
309 % Plotting convergence history
310 plot(1:Niterations,Lambda(1)*ones(1,Niterations),'--b','LineWidth',1.5)
311 plot(1:Niterations,RQ,'-ok','LineWidth',1.5)
312
313 % Legend and labels
314 lh = legend('Actual value of $\lambda_1$',...
315     'Numerical value of $\lambda_1$');
316 set(lh,'FontSize',11)
317 set(lh,'Interpreter','LaTeX')
318 xlabel('Iteration','FontSize',11)
319 ylabel('First Eigenvalue ($\lambda_1$)','FontSize',11)
320 set(gca,'box','on')
321 set(gca,'XTick',1:Niterations)
322 set(gca,'XLim',[0.999,5])
323
324 % Saving plot
325 set(gcf,'PaperPositionMode','auto')
326 print(gcf, '-depsc', [ImgPath,'Hw9Prob5.eps'])
327
328 fclose(fid);

```