Appendix - III

MATLAB Program for Examples of Chapter 5

Example 5.1 (Case 1), Single Support Excitation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Response of MDOF System with Single-Support Earthquake Excitation
%% Solution obtained by State Space Method
%% Earthquake ground motion considered is 'El Centro, 1940 (N-S component)'

clear
clc
close all
format short g

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nd = 2;                    % Enter nos. of superstructure degrees of freedom
m = 10;                          % Enter mass of building in 'kg'
k = 1000;                        % Enter stiffness of frame in 'N/m'
M = [2*m 0;0 m]                  % Generate the mass matrix
K = [12*k -4*k;-4*k 4*k]         % Generate the stiffness matrix
[V,D] = eig(K,M);                % Eigen values and Eigen vectors
w = sqrt([D]);
phi1 = [V(1)/V(2);V(2)/V(1)];    % Mode shape
phi2 = [V(3)/V(4);V(4)/V(4)];    % Mode shape
w1 = sqrt(D(1));                 % Natural frequency
w2 = sqrt(D(4));                 % Natural frequency
zhy = 5;                         % Enter damping (in percentage)
w = [1/(2*w1) w1/2;1/(2*w2) w2/2];
Zhy = [zhy/100;zhy/100];
const_matrx = w\Zhy;
alpha = const_matrx(1);
beta = const_matrx(2);
C = alpha*[M]+beta*[K]           % Rayleigh's Damping matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load El_centro_unit_g_dt_0.02.mat  % Load the earthquake ground motion file
l = length(a);
a = a*9.81;
dt1 = 0.02;          % Enter existing time step, as in original .mat file
t_t = (l-1)*dt1;
t=0:dt1:t_t;
time = t';
n = a;
div = 1;            % Enter the nos. of division for required interpolation
dt = dt1/div;
ti = 0:dt:t_t;
time1 = ti';
ni = interp1(t,n,ti);
a = ni';                           % Ground acceleration vector
l1= length(a);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% State Space method matrices
A = [zeros(nd,nd) eye(nd);-inv(M)*K -inv(M)*C];
Ad = expm([[A]*dt]);
D = inv(A)*[Ad-eye(2*nd)];
E = [zeros(nd,1);-1;-1];
Ed = [D]*[E];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initial conditions
z = zeros(2*nd,1);
d1 = 0;
d2 = 0;
v1 = 0;
v2 = 0;
v11 = 0;
v21 = 0;
a1 = -a(1);
a2 = -a(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

j = 1;                % To store the values at certain intervals, to reduce
the program execution time

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:l1-1

z = [Ad]*[z]+[Ed]*[a(i)];    % State vector for displacement and velocity
zdot = A*z+E*a(i+1);         % State vector for velocity and acceleration

if i == j              % To store the values at certain intervals, to
reduce the program execution time

d1 = [d1;z(1)];        % Relative displacement corresponding to DOF#1
d2 = [d2;z(2)];        % Relative displacement corresponding to DOF#2
v1 = [v1;z(3)];        % Relative velocity corresponding to DOF#1
(calculated from state vector 'z')
v2 = [v2;z(4)];        % Relative velocity corresponding to DOF#2
(calculated from state vector 'z')
v11 = [v11;zdot(1)];   % Relative velocity corresponding to DOF#1
(calculated from state vector 'zdot' for check)
v21 = [v21;zdot(2)];   % Relative velocity corresponding to DOF#2
(calculated from state vector 'zdot' for check)
a1 = [a1;zdot(3)];     % Relative acceleration corresponding to DOF#1
a2 = [a2;zdot(4)];     % Relative acceleration corresponding to DOF#2

j = j+div;
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a_abs_1 = [a1+a];      % Absolute acceleration corresponding to DOF#1
a_abs_2 = [a2+a];      % Absolute acceleration corresponding to DOF#2
```

```
Max_d1 = max(abs(d1));       % Maximum displacement corresponding to DOF#1
Max_d2 = max(abs(d2));       % Maximum displacement corresponding to DOF#2
Max_a1 = max(abs(a1));       % Maximum acceleration corresponding to DOF#1
Max_a2 = max(abs(a2));       % Maximum acceleration corresponding to DOF#1
Max_a_abs_1 = max(abs(a_abs_1)); % Maximum absolute acceleration
corresponding to DOF#1
Max_a_abs_2 = max(abs(a_abs_2)); % Maximum absolute acceleration
corresponding to DOF#2

RMS_d1 = sqrt(sum(d1.*conj(d1))/size(d1,1));    % RMS displacement
corresponding to DOF#1
RMS_d2 = sqrt(sum(d2.*conj(d1))/size(d2,1));    % RMS displacement
corresponding to DOF#2
RMS_a1 = sqrt(sum(a1.*conj(a1))/size(a1,1));    % RMS acceleration
corresponding to DOF#1
RMS_a2 = sqrt(sum(a2.*conj(a2))/size(a2,1));    % RMS acceleration
corresponding to DOF#2
RMS_a_abs_1 = sqrt(sum(a_abs_1.*conj(a_abs_1))/size(a_abs_1,1)); % RMS
absolute acceleration corresponding to DOF#1
RMS_a_abs_2 = sqrt(sum(a_abs_2.*conj(a_abs_2))/size(a_abs_2,1)); % RMS
absolute acceleration corresponding to DOF#2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Result_Table =
[Max_d1;Max_d2;Max_a_abs_1;Max_a_abs_2;RMS_d1;RMS_d2;RMS_a_abs_1;RMS_a_abs_
2]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Time history Plots
figure(1)
plot(time,d1)
xlabel ('Time(s)')
ylabel ('Relative Displacement x1 (m)')
title('Time History for Relative Displacement corresponding to DOF#1')
grid on
figure(2)
plot(time,d2)
xlabel ('Time(s)')
ylabel ('Relative Displacement x2 (m)')
title('Time History for Relative Displacement corresponding to DOF#2')
grid on
figure(3)
plot(time,a_abs_1)
xlabel ('Time(s)')
ylabel ('Absolute Acceleration x1 (m/s2)')
title('Time History for Absolute Acceleration corresponding to DOF#1')
grid on
figure(4)
plot(time,a_abs_2)
xlabel ('Time(s)')
ylabel ('Absolute Acceleration x2 (m/s2)')
title('Time History for Absolute Acceleration corresponding to DOF#2')
grid on

% Program Ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Example 5.1 (Case 2), Multi Support Excitation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Response of MDOF System with Multi-Support Earthquake Excitation
%% Solution obtained by State Space Method
%% Earthquake ground motion considered is 'El Centro, 1940 (N-S component)'

clear
clc
close all
format short g

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nd = 2;                    % Enter nos. of superstructure degrees of freedom
m = 10;                          % Enter mass of building in 'kg'
k = 1000;                        % Enter stiffness of frame in 'N/m'
M = [2*m 0;0 m]                  % Generate the mass matrix
K = [12*k -4*k;-4*k 4*k]         % Generate the stiffness matrix
[V,D] = eig(K,M);                % Eigen values and Eigen vectors
w = sqrt([D]);
phi1 = [V(1)/V(2);V(2)/V(1)];    % Mode shape
phi2 = [V(3)/V(4);V(4)/V(4)];    % Mode shape
w1 = sqrt(D(1));                 % Natural frequency
w2 = sqrt(D(4));                 % Natural frequency
zhy = 5;                         % Enter damping (in percentage)
w = [1/(2*w1) w1/2;1/(2*w2) w2/2];
Zhy = [zhy/100;zhy/100];
const_matrx = w\Zhy;
alpha = const_matrx(1);
beta = const_matrx(2);
C = alpha*[M]+beta*[K]           % Rayleigh's Damping matrix

Ksg = [-2*k -2*k -2*k -2*k;0 0 0 0]
r = -inv(K)*Ksg                  % Influence coefficient matrix

% for ground motion # 1
load El_centro_unit_g_dt_0.02_gm1.mat   % Load the earthquake ground motion
file
l = length(agm_1);
agm_1 = agm_1*9.81;
dt1 = 0.02;            % Enter existing time step, as in original .mat file
t_t = (l-1)*dt1;
t = 0:dt1:t_t;
time = t';
n = agm_1;
div = 1;            % Enter the nos. of division for required interpolation
dt = dt1/div;
ti = 0:dt:t_t;
time1 = ti';
ni = interp1(t,n,ti);
agm_1 = ni';                     % Ground acceleration vector for support#1
l1 = length(agm_1);

% for ground motion # 2 (with time delay of 5 sec)
load El_centro_unit_g_dt_0.02_gm2.mat   % Load the earthquake ground motion
file
l = length(agm_2);
agm_2 = agm_2*9.81;
```

```
dt1 = 0.02;                         % Enter existing time step, as in original
.mat file
t_t = (l-1)*dt1;
t = 0:dt1:t_t;
time = t';
n = agm_2;
div = 1;                            % Enter the nos. of division for required
interpolation
dt = dt1/div;
ti = 0:dt:t_t;
time1 = ti';
ni = interp1(t,n,ti);
agm_2 = ni';                        % Ground acceleration vector for support#2
l1 = length(agm_2);

% for ground motion # 3 (with time delay of 10 sec)
load El_centro_unit_g_dt_0.02_gm3.mat   % Load the earthquake ground motion
file
l = length(agm_3);
agm_3 = agm_3*9.81;
dt1 = 0.02;                         % Enter existing time step, as in original
.mat file
t_t = (l-1)*dt1;
t = 0:dt1:t_t;
time = t';
n = agm_3;
div = 1;                            % Enter the nos. of division for required
interpolation
dt = dt1/div;
ti = 0:dt:t_t;
time1 = ti';
ni = interp1(t,n,ti);
agm_3 = ni';                        % Ground acceleration vector for support#3
l1 = length(agm_3);

% for ground motion # 4 (with time delay of 15 sec)
load El_centro_unit_g_dt_0.02_gm4.mat   % Load the earthquake ground motion
file
l = length(agm_4);
agm_4 = agm_4*9.81;
dt1 = 0.02;                         % Enter existing time step, as in original
.mat file
t_t = (l-1)*dt1;
t = 0:dt1:t_t;
time = t';
n = agm_4;
div = 1;                            % Enter the nos. of division for required
interpolation
dt = dt1/div;
ti = 0:dt:t_t;
time1 = ti';
ni = interp1(t,n,ti);
agm_4 = ni';                        % Ground acceleration vector for support#4
l1 = length(agm_4);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% State Space method matrices
A = [zeros(nd,nd) eye(nd);-inv(M)*K -inv(M)*C];
Ad = expm([[A]*dt]);
```

```
D = inv(A)*[Ad-eye(2*nd)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initial conditions
z = zeros(2*nd,1);
d1 = 0;
d2 = 0;
v1 = 0;
v2 = 0;
v11 = 0;
v21 = 0;
a1 = -
[(r(1,1)*agm_1(1))+(r(1,2)*agm_2(1))+(r(1,3)*agm_3(1))+(r(1,4)*agm_4(1))];
a2 = -
[(r(2,1)*agm_1(1))+(r(2,2)*agm_2(1))+(r(2,3)*agm_3(1))+(r(2,4)*agm_4(1))];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

j = 1; % To store the values at certain intervals, to reduce the program
execution time

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:l1-1

GAV_z = [agm_1(i);agm_2(i);agm_3(i);agm_4(i)];
Ez = r*GAV_z;
E_z = [zeros(nd,1);-Ez(1);-Ez(2)];
Ed = [D]*[E_z];
z = [Ad]*[z]+[Ed];             % State vector for displacement and velocity

GAV_zdot = [agm_1(i+1);agm_2(i+1);agm_3(i+1);agm_4(i+1)];
Ezdot = r*GAV_zdot;
E_zdot = [zeros(nd,1);-Ezdot(1);-Ezdot(2)];
zdot = [A]*[z]+[E_zdot];      % State vector for velocity and acceleration

if i == j              % To store the values at certain intervals, to
reduce the program execution time

d1 = [d1;z(1)];        % Relative displacement corresponding to DOF#1
d2 = [d2;z(2)];        % Relative displacement corresponding to DOF#2
v1 = [v1;z(3)];        % Relative velocity corresponding to DOF#1
(calculated from state vector 'z')
v2 = [v2;z(4)];        % Relative velocity corresponding to DOF#2
(calculated from state vector 'z')
v11 = [v11;zdot(1)];   % Relative velocity corresponding to DOF#1
(calculated from state vector 'zdot' for check)
v21 = [v21;zdot(2)];   % Relative velocity corresponding to DOF#2
(calculated from state vector 'zdot' for check)
a1 = [a1;zdot(3)];     % Relative acceleration corresponding to DOF#1
a2 = [a2;zdot(4)];     % Relative acceleration corresponding to DOF#2

j = j+div;
end
end

a_abs_1 = [a1+(r(1,1)*agm_1)+(r(1,2)*agm_2)+(r(1,3)*agm_3)+(r(1,4)*agm_4)];
% Absolute acceleration corresponding to DOF#1
```

```
a_abs_2 = [a2+(r(1,1)*agm_1)+(r(1,2)*agm_2)+(r(1,3)*agm_3)+(r(1,4)*agm_4)];
% Absolute acceleration corresponding to DOF#2

Max_d1 = max(abs(d1));        % Maximum displacement corresponding to DOF#1
Max_d2 = max(abs(d2));        % Maximum displacement corresponding to DOF#2
Max_a1 = max(abs(a1));        % Maximum acceleration corresponding to DOF#1
Max_a2 = max(abs(a2));        % Maximum acceleration corresponding to DOF#1
Max_a_abs_1 = max(abs(a_abs_1)); % Maximum absolute acceleration
corresponding to DOF#1
Max_a_abs_2 = max(abs(a_abs_2)); % Maximum absolute acceleration
corresponding to DOF#2

RMS_d1 = sqrt(sum(d1.*conj(d1))/size(d1,1));    % RMS displacement
corresponding to DOF#1
RMS_d2 = sqrt(sum(d2.*conj(d1))/size(d2,1));    % RMS displacement
corresponding to DOF#2
RMS_a1 = sqrt(sum(a1.*conj(a1))/size(a1,1));    % RMS acceleration
corresponding to DOF#1
RMS_a2 = sqrt(sum(a2.*conj(a2))/size(a2,1));    % RMS acceleration
corresponding to DOF#2
RMS_a_abs_1 = sqrt(sum(a_abs_1.*conj(a_abs_1))/size(a_abs_1,1)); % RMS
absolute acceleration corresponding to DOF#1
RMS_a_abs_2 = sqrt(sum(a_abs_2.*conj(a_abs_2))/size(a_abs_2,1)); % RMS
absolute acceleration corresponding to DOF#2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Result_Table =
[Max_d1;Max_d2;Max_a_abs_1;Max_a_abs_2;RMS_d1;RMS_d2;RMS_a_abs_1;RMS_a_abs_
2]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Time history Plots
figure(1)
plot(time,d1)
xlabel ('Time(s)')
ylabel ('Relative Displacement x1 (m)')
title('Time History for Relative Displacement corresponding to DOF#1')
grid on
figure(2)
plot(time,d2)
xlabel ('Time(s)')
ylabel ('Relative Displacement x2 (m)')
title('Time History for Relative Displacement corresponding to DOF#2')
grid on
figure(3)
plot(time,a_abs_1)
xlabel ('Time(s)')
ylabel ('Absolute Acceleration x1 (m/s2)')
title('Time History for Absolute Acceleration corresponding to DOF#1')
grid on
figure(4)
plot(time,a_abs_2)
xlabel ('Time(s)')
ylabel ('Absolute Acceleration x2 (m/s2)')
title('Time History for Absolute Acceleration corresponding to DOF#2')
grid on
% Program Ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```