Appendices:

A.1. Code for Q1

```
% MEC
% Q1E
clear;

% Parameters
m = 1;
mu = 0.5;
k = 5;
t = 5;

% Time span
tstep = 0.1;
t_vector = 0:tstep:t;

% Initial conditions
x0 = [0; 5];

% Populate A and B matrix
A = zeros(2,2);
A(1,1) = 0;
A(1,2) = 1;
A(2,1) = -k/m;
A(2,2) = -mu/m;

B = [0 ; 1/m];

% Solve for unforced linear system
x = [];
for time = 0:tstep:t
    curr_x = expm(A*time)*x0;
    x = [x, curr_x];
end

% Plot
figure;

plot(t_vector, x(1,:));
hold on
plot(t_vector, x(2,:));
title("Plot of unforced system vs. time for spring-mass-damper system");
legend("Position of mass","Velocity of mass");
xlabel("time");
ylabel("x-value");
hold off

% Q1F
% Desired eigenvalues
p = [complex(-1,1);
    complex(-1,-1)];

% K matrix
K = place(A,B,p);
eigs = eig(A-B*K);

% Q1G
% Parameters
t = 10;
x0 = [1; 1];

% Time span
tstep = 0.1;
t_vector = 0:tstep:t;

% Solve for linear state feedback system
x = [];
for time = 0:tstep:t
    curr_x = expm((A-B*K)*time)*x0;
    x = [x, curr_x];
end

% Plot
figure;

plot(t_vector, x(1,:));
hold on
plot(t_vector, x(2,:));
title("Plot of linear state feedback system vs. time for spring-mass-damper system");
legend("Position of mass","Velocity of mass");
xlabel("time");
ylabel("x-value");
hold off
```

A.2. Code for Q2B

```
% MEC
% Q2B
clear;

syms alpha beta gamma D mu u x3 xcdotdot x2 x4 phidotdot
eq1 = gamma * xcdotdot - beta * phidotdot * cos(x2) + beta * x4 * x4 * sin(x2) + mu * x3 == u;
eq2 = alpha * phidotdot - beta * xcdotdot * cos(x2) - D * sin(x2) == 0;
sol = solve(eq1, eq2, xcdotdot, phidotdot);
disp(sol.xcdotdot);
disp(sol.phidotdot);
```

A.3. Code for Q2C-G

```
% MEC
% Q2C
clear;

% Parameters
gamma = 2;
alpha = 1;
beta = 1;
D = 1;
mu = 3;

% Populate A and B matrix
A = zeros(4,4);
A(1,3) = 1;
A(2,4) = 1;
A(3,2) = 1;
A(3,3) = -3;
A(4,2) = 2;
A(4,3) = -3;

B = zeros(4,1);
B(3,1) = 1;
B(4,1) = 1;

% Eigenvalues of A
eigs = eig(A);

% Q matix for LQR
Qu = 10;

Qx = zeros(4,4);
Qx(1,1) = 1;
Qx(2,2) = 5;
Qx(3,3) = 1;
Qx(4,4) = 5;

```

```matlab
% LQR
[K,S,P] = lqr(A,B,Qx,Qu);

% Timespan
T = 0.01;
tspan = [0 30];
t_vector = 0:T:30;

% Initial conditions
% x0 = transpose([0, 0.1, 0, 0]);
% x0 = transpose([0, 0.5, 0, 0]);
% x0 = transpose([0, 1.0886, 0, 0]);
% x0 = transpose([0, 1.1, 0, 0]);

% Run ode45 for linearized system
% [t, x] = ode45(@(t, x) odefun(t, x, A, B, K), t_vector, x0);

% Run ode45 for original non-linear system
[t, x] = ode45(@(t, x) odefunnl(t, x, gamma, alpha, beta, D, mu, K), t_vector, x0);

% For the [0, 1.1, 0, 0]^T non-linear system initial state, ode45 is not
% able to plot beyond 9.6 secs. Plotting using ode23t instead to 15 secs

% tspan = [0 15];
% [t, x] = ode15s(@(t, x) odefunnl(t, x, gamma, alpha, beta, D, mu, K), tspan, x0);
% t_vector = 0:(30/(length(x)-1)):30;

% Plotting
figure();
plot(t_vector,x(:,1));
hold on
plot(t_vector,x(:,2));
plot(t_vector,x(:,3));
plot(t_vector,x(:,4));
% title("State dynamics of linearized system");
title("State dynamics of original non-linear system");
legend("xc (m)", "phi (rad)", "xcdot (m/sec)", "phidot (rad/sec)");
xlabel("time (sec)");
ylabel("state");
hold off

% Create function for linearized ODE
function dxdt = odefun(t, x, A, B, K)
    dxdt = (A - B * K) * x;
end

% Create function for original non-linear ODE
function dxdt = odefunnl(t, x, gamma, alpha, beta, D, mu, K)
    u = -(K * x);
    x1 = x(1);
    x2 = x(2);
    x3 = x(3);
    x4 = x(4);

    dxdt = zeros(4,1);
    dxdt(1) = x3;
    dxdt(2) = x4;
    dxdt(3) = (-alpha*sin(x2)*beta*x4^2 + alpha*u - alpha*mu*x3 + cos(x2)*sin(x2)*D*beta)/(alpha*gamma - beta^2*cos(x2)^2);
    dxdt(4) = (- cos(x2)*sin(x2)*beta^2*x4^2 + u*cos(x2)*beta + sin(x2)*D*gamma - mu*x3*cos(x2)*beta)/(alpha*gamma - beta^2*cos(x2)^2);
end
```

A.4. Code for Q2H-I

```matlab
% MEC
% Q2H
clear;

% Parameters
```

```
gamma = 2;
alpha = 1;
beta = 1;
D = 1;
mu = 3;

% Populate A, B, and C matrix
A = zeros(4,4);
A(1,3) = 1;
A(2,4) = 1;
A(3,2) = 1;
A(3,3) = -3;
A(4,2) = 2;
A(4,3) = -3;

B = zeros(4,1);
B(3,1) = 1;
B(4,1) = 1;

C = [39.37 0 0 0];

% Eigenvalues of A
eigs = eig(A);

% Q matix for LQR
Qu = 1;

Qx = zeros(4,4);
Qx(1,1) = 50;
Qx(2,2) = 1;
Qx(3,3) = 5;
Qx(4,4) = 5;

% LQR
[K,S,P] = lqr(A,B,Qx,Qu);

% Timespan
T = 0.01;
tspan = [0 200];
t_vector = 0:T:200;

% Initial conditions
x0 = transpose([0, 0, 0, 0]);

% Run ode45 for original non-linear system
[t, x] = ode45(@(t, x) odefunnl(t, x, gamma, alpha, beta, D, mu, K, A, B, C), t_vector, x0);

% Plotting
figure();
plot(t_vector,x(:,1));
hold on
plot(t_vector,x(:,2));
plot(t_vector,x(:,3));
plot(t_vector,x(:,4));
title("State dynamics of non-linear system");
legend("xc (m)", "phi (rad)", "xcdot (m/sec)", "phidot (rad/sec)");
xlabel("time (sec)");
ylabel("state");
hold off

yd = 20 * square(0.02 * pi * t_vector);
y = C * transpose(x);
figure();
plot(t_vector,y);
hold on
plot(t_vector,yd);
title("Desired and actual outputs");
legend("Actual xc", "Desired xc");
xlabel("time (sec)");
ylabel("output (in)");
hold off

% Create function for original non-linear ODE
```

```matlab
function dxdt = odefunnl(t, x, gamma, alpha, beta, D, mu, K, A, B, C)
    % Find feedback control from linearized ODE
    yd = 20 * square(0.02 * pi * t);
    v = -inv((C * inv(A - B * K) * B)) * yd;
    u = -(K * x) + v;

    x1 = x(1);
    x2 = x(2);
    x3 = x(3);
    x4 = x(4);

    dxdt = zeros(4,1);
    dxdt(1) = x3;
    dxdt(2) = x4;
    dxdt(3) = (-alpha*sin(x2)*beta*x4^2 + alpha*u - alpha*mu*x3 + cos(x2)*sin(x2)*D*beta)/(alpha*gamma - beta^2*cos(x2)^2);
    dxdt(4) = (- cos(x2)*sin(x2)*beta^2*x4^2 + u*cos(x2)*beta + sin(x2)*D*gamma - mu*x3*cos(x2)*beta)/(alpha*gamma - beta^2*cos(x2)^2);
end
```