EE 362K
Homework 3 Solutions

Q1

14 points
1 point each for calculating damping coefficient, damped frequency, natural frequency, 1 point each for plot of numerical and analytical solution. 1 point for labels, 1 point each for damped frequency and damping coefficient from graph, 2 points for natural frequency from graph, 1 point for comparison, 3 points for explaining change in system response on doubling mass

$$k = 4, m = 3, c = 0.25$$

$$\omega_n = \sqrt{\frac{k}{m}} = 1.1547 rad/s$$

$$\zeta = \frac{c}{2\sqrt{km}} = 0.0361$$

$$\omega_d = \omega_n\sqrt{1-\zeta^2} = 1.1539 rad/s$$

The initial condition used to generate the following graphs was $x(0) = 0, \dot{x}(0) = 1$ as can be seen in the code.

```
%% Q1

clear all
close all

global A;

k = 4; c = 0.25; m = 3;
A = [0 1; -k/m -c/m];

% Numerical solution
[t,z] = ode45(@q1slope,[0,30],[0,1]);

figure;
hold on
plot(t,z(:,1),'b','LineWidth',2)

% Analytical solution
wn = sqrt(k/m);         % natural frequency
zeta = 0.5*c/sqrt(k*m); % damping ration
wd = wn*sqrt(1-zeta^2); % damped frequency

x = exp(-zeta*wn*t).*1/wd.*sin(wd*t);
plot(t,x,'rx--')
xlabel('Time (in s)')
ylabel('Position')
legend('Numerical','Analytical')
```

```
%% function to compute dz/dt for given z, A for Q1
% z is a state (column vector)
% dz/dt = A*z


function f = q1slope(t,z)
     global A
     f = A*z;
end
```

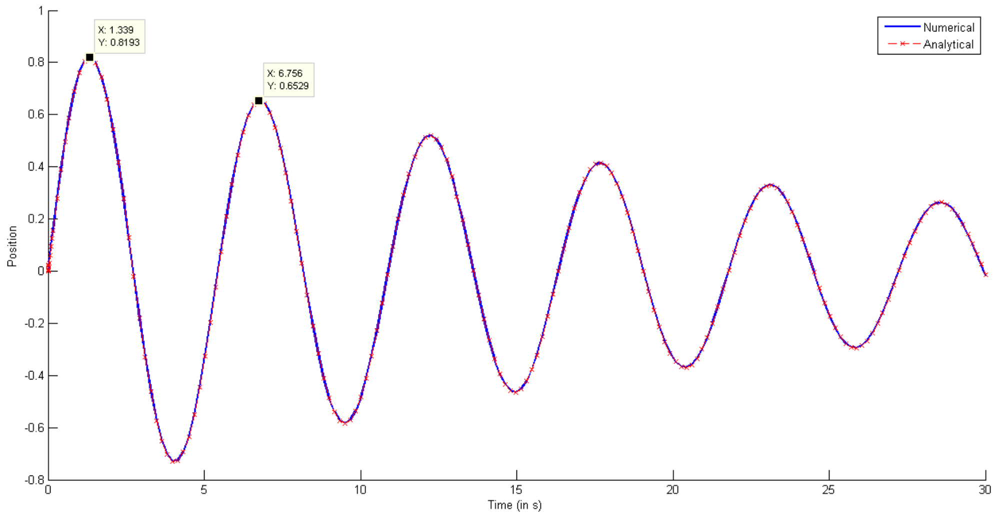


Figure 1: Analytical and numerical solutions for mass spring damper system

From the graph, we can see that the observed frequency, $\omega_d = \frac{2\pi}{6.756-1.339} = 1.16rad/s$. From the decay of the peaks, we can estimate $\zeta\omega_n = \frac{1}{6.756-1.339}\ln\frac{0.8193}{0.6529} = 0.0434$.

$\omega_n = \sqrt{\omega_d^2 + (\zeta\omega_n)^2} = 1.1607rad/s$. This gives $\zeta = 0.0373$. We can see that there is a fairly close agreement between the estimated and calculated values of ω_n, ω_d and ζ.

From the formulae used to calculate the above quantities, we can see that doubling the mass would

- reduce the natural frequency by a factor of $\frac{1}{\sqrt{2}}$
- reduce the damping coefficient by a factor of $\frac{1}{\sqrt{2}}$. This means that the sinusoid would decay even more slowly (since $\zeta\omega_n$ reduces by a factor of $\frac{1}{2}$).
- It is not possible to determine the effect on damped frequency simply by observation. However, since the damping ratio is already very small, the damped frequency decreases on doubling the mass (approximately by a factor of $\frac{1}{\sqrt{2}}$). Calculation validates this. This means that the decaying sinusoid that we see will oscillate with a larger time period.
- The initial amplitude is inversely proportional to ω_d, so it would increase.

Q2 (drug administration)

8 points
2 points for appropriate input, 2 points for function to be passed to ode45, 1 point for using ode45

correctly, 3 points for labelled plots

The alcohol intake was assumed to be constant over 8 hours (480 minutes). So, the intravenous alcohol intake would be $\frac{40\times1000}{46\times480} mmol/min$. The oral intake can be similarly calculated. Plugging these values as input into the code gives the graph below.

```matlab
%% Q2

clear all
close all

global Vb Vl q qmax c0 u inp ta

% defining constants
ta = 480;    % assuming that the alcohol doses are administered over 60 minutes
Vb = 48;     % in L
Vl = 0.6;    % in L
q = 1.5;     % L/min
qmax = 2.75;     % mmol/min
c0 = 0.1;    % mmol/L
u = [40; 12]/46/ta*1000;
inp = 1;    % 1 while input is administered

c_init = zeros(2,1);     % initial concentrations - [cb; cl]

[t,c] = ode45(@q2rate,[0 600],c_init);

figure;
plot(t,c);
legend({'$c_b$','$c_l$'},'Interpreter','latex')
xlabel('Time (in min)')
ylabel('Concentration (in mmol/min)')
```

```matlab
% function to compute rate of change in concentrations
% t is time and c is
function dc = q2rate(t,c)

    global Vb Vl q qmax c0 u inp ta

    dc = zeros(2,1);
    if (t>ta)
        inp = 0;
    end
    dc(1) = (q*(c(2)-c(1))+inp*u(1))/Vb;
    dc(2) = (q*(c(1)-c(2)) - qmax*c(2)/(c0+c(2)) + inp*u(2))/Vl;
end
```

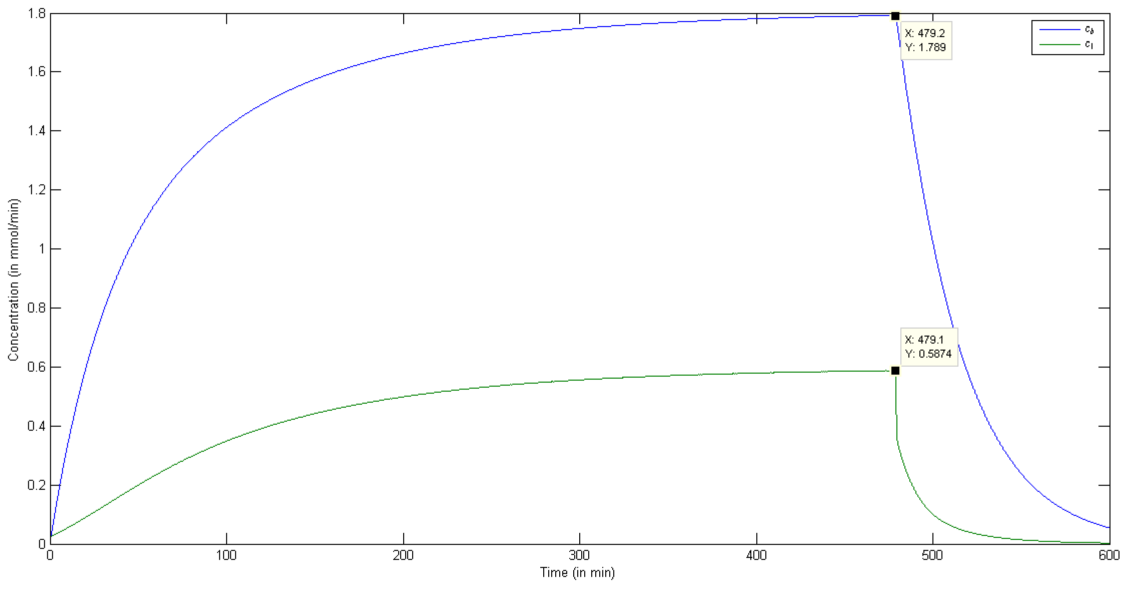


Figure 2: Graphs of alcohol concentrations in blood and liver

Q2 (cruise controller)

14 points
2 points for saturating input, 3 points for vehicle dynamics, 3 points for designing PI controller, 2 points for varied initial conditions, 1 point for identifying axes for phase portrait, 3 points for phase portrait

The file *cruisedyn.m* given for the previous homework has been reused here. The throttle input is a function of the error, $e = v_r - v$ with $u = k_p e + \int_o^t edt$. The integration was approximated by a summation.

```
% Function to compute u = kp(e) + ki(integral of error)
% vref is the desired speed, v is an array of speeds that the
% vehicle has taken, timestep is the time difference between
% successive speed "measurements"

function thr = throttle(vref,v,timestep)
    global kp ki

    thr = kp*(vref-v(end))+ki*(sum(timestep*(vref-v)));
    thr = min(thr,1); thr = max(thr,0);

end
```

acc.m computes the acceleration of the vehicle for the current speed and throttle input.

```
% Function to compute accelaration at current speed (v)
function dv = acc(~,v)

    global m vehgr theta u

    dv = cruisedyn(v, u, vehgr, theta, m);

end
```

Finally, ode45 was used to find the speed for the next time step.

```
%% Cruise controller

 close all
 clear all

 global m vehgr theta u

m = 1000;        % mass of vehicle
 vref = 20;     % nominal speed, m/s
 vehgr = 3;                 % gear
 theta = 0;       % flat ground

% Control law parameters
 global kp ki
 kp = 0.5;
 ki = 0.1;


 hold on

 v_init = linspace(10,30,20);     % initial speeds
jm = length(v_init);

h = waitbar(0, 'Please wait');

 for j = 1:length(v_init)
     vi = v_init(j);          % initial speed for this phase plane curve
     timestep = 0.01;
     time = 0:0.1:100;        % each phase plane curve computed for 100s
     vcurr = zeros(1,length(time));
     acc_curr = zeros(1,length(time));
     vcurr(1) = vi;  % initialising speed


     for i = 1:length(time)-1
         u= throttle(vref,vcurr(1:i),0.1);    % output of PI controller
         acc_curr(i) = acc(1,vcurr(i));  % computing acceleration
         [~,v] = ode45(@acc,time(i:i+1),vcurr(i));    % finding velocity at next time step
         vcurr(i+1) = v(end);
     end

    % updating final values of arrays
     u= throttle(vref,vcurr,0.1);
     acc_curr(end) = acc(1,vcurr(end));
     plot(vcurr,acc_curr, '-s', 'Color',[j/jm,4*j/jm*(1-j/jm),1-j/jm], 'MarkerFaceColor', [0.9*j/jm
         ,0.5*j/jm,1-0.9*j/jm], 'MarkerSize',2);
     plot(vcurr(1),acc_curr(1),'kx','MarkerSize',10)
     xlim([10 30])
     ylim([-1 3])

     waitbar(j/jm);
end

 close(h);
 xlabel('Speed')
 ylabel('Acceleration')
```

The phase portrait thus obtained is shown below. The crosses indicate the initial points for which the phase plane curves have been drawn.

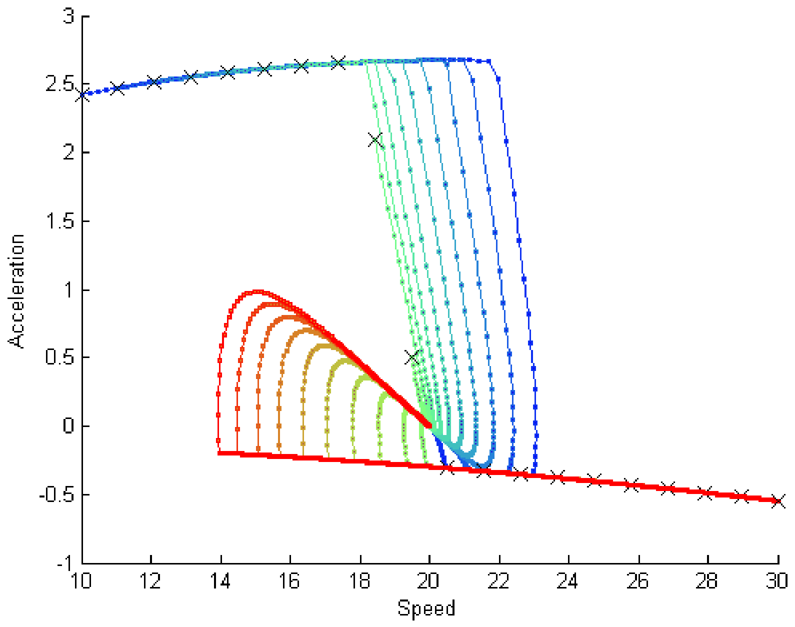


Figure 3: Phase portrait for cruise controller

Q3

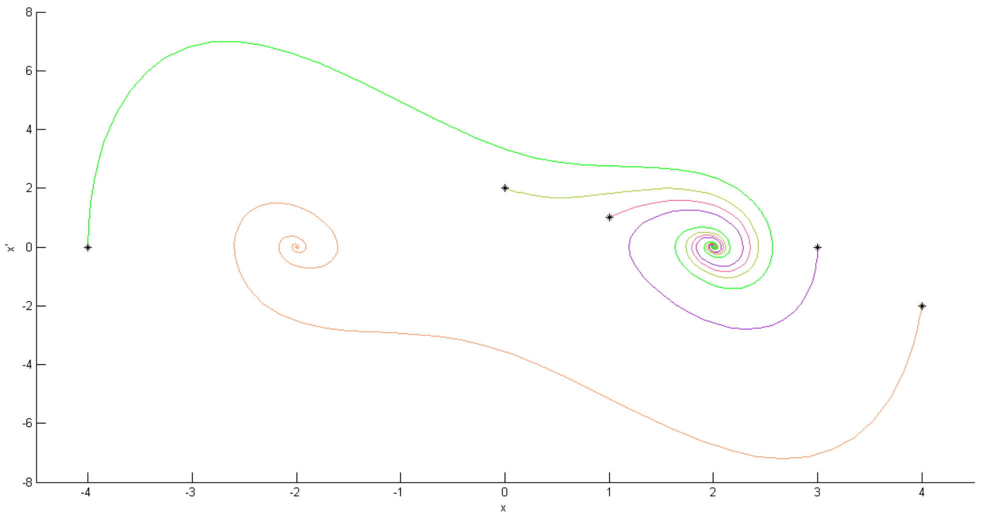
15 points
2 points for finding equilibrium points, 1 point for method to determine stability, 2 points each for stability of equilibrium point, 1 point for each plane phase curve, 1 point for labels

Setting $z_1 = x$, $z_2 = \dot{x}$, we get that $\dot{z_1} = z_2$ and $\dot{z_2} = 4z_1 - z_1^3 - 1.2z_2$.
For equilibrium points, we set the state derivatives to zero. This gives us the three equilibrium points $(0, 0), (-2, 0)$ and $(2, 0)$.

To determine the stability of these points, we shall linearize the system. This gives us that $\Delta z_1 = \Delta z_2$ and $\Delta z_2 = (4 - 3z_1^2)\Delta z_1 - 1.2\Delta z_2$. So, in $\dot{(z)} = Az$, we get that $A = \begin{bmatrix} 0 & 1 \\ 4 - 3z_1^2 & -1.2 \end{bmatrix}$

- For $(0, 0)$, $A = \begin{bmatrix} 0 & 1 \\ 4 & -1.2 \end{bmatrix}$. A has a real positive and negative eigenvalue. So, this is an unstable equilibrium point.
- For $(-2, 0)$, $A = \begin{bmatrix} 0 & 1 \\ -8 & -1.2 \end{bmatrix}$. A has complex conjugate eigenvalues with a negative real part . So, this is a stable equilibrium point.
- For $(-2, 0)$, A is the same as the previous case. So, it is also a stable equilibrium point.

The initial conditions used to generate the phase portrait below are $(3, 0), (1, 1), (4, -2), (0, 2)$ and $(-4, 0)$. A quiver/velocity plot has also been shown to get a sense of direction for the phase-plane curves. This plot plots an arrow indicating the direction of the state derivative at each point.



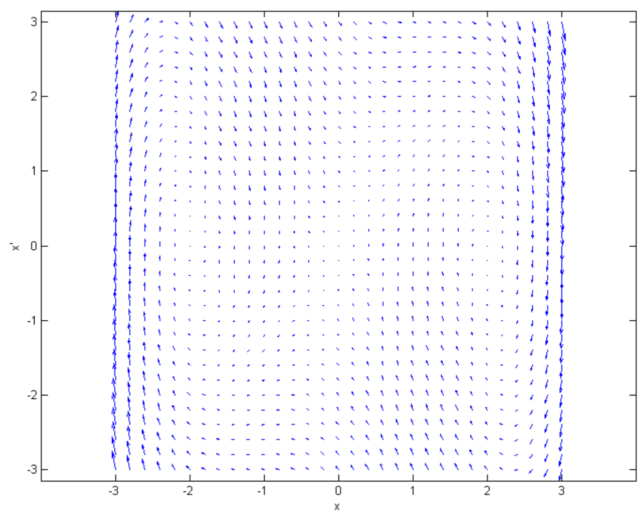


Figure 4: Phase portrait and quiver plot for given system

Q4

10 points
1 point for characteristic equation, 1 point for eigenvalues, 1 point for each eigenvector, 1 point for explanation of discrepancy with MATLAB

(a)

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

So, the characteristic equation, given by $det(\lambda I - A) = 0$, is $(\lambda - 1)^2 - 1 = 0 \implies \lambda^2 - 2\lambda = 0$. Thus, the eigenvalues of A are 0 and 2. By observation, the eigenvector for the eigenvalue 0, is $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and for that for the eigenvalue 2 is $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

(b)

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 0 \\ 2 & -4 & 2 \end{bmatrix}$$

The characteristic equation is $(\lambda-1)(\lambda-2)(\lambda-3)=0$. So, the eigenvalues are 1,2 and 3. For the eigenvector for eigenvalue 1, we have

$$(A-I)v_1 = 0$$
$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 2 & 0 \\ 2 & -4 & 1 \end{bmatrix} v_1 = 0$$

We can see that $v_1 = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix}$.
For the eigenvector for eigenvalue 2, we have

$$(A-2I)v_2 = 0$$
$$\begin{bmatrix} -1 & 2 & 0 \\ 0 & 1 & 0 \\ 2 & -4 & 0 \end{bmatrix} v_2 = 0$$

So, $v_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.
For the eigenvector for eigenvalue 3, we have

$$(A-3I)v_3 = 0$$
$$\begin{bmatrix} -2 & 2 & 0 \\ 0 & 0 & 0 \\ 2 & -4 & -1 \end{bmatrix} v_3 = 0$$

Careful observation gives us $v_3 = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}$.

The eigenvectors returned by MATLAB are normalized and any sign mismatch is due to the fact that if v is an eigenvector of A, then so is $-v$.

Q5

12 points
3 points for linearization, 1 point for characteristic equation, 1 point for finding eigenvalues by

hand,1 point each for (a),(c),(d) and (e), 3 points for (b)

The linearization of the system, as explained in the solution to Q3, gives us $A = \begin{bmatrix} 0 & 1 \\ 1 & -1.2 \end{bmatrix}$. This gives us the characteristic equation $\lambda^2 + 1.2\lambda - 1 = 0$. So, the eigenvalues are $\frac{-1.2 \pm \sqrt{5.44}}{2} = 0.5662, -1.7662$.

(a)

Using the *eig* command in MATLAB, we get the same eigenvalues.

(b)

The condition number is $|\frac{-1.7662}{-0.5662}| = 3.1194$. This implies that A is a well-conditioned matrix. Such a system is robust to numerical errors and it is invertible, i.e., we can derive the present state from the state velocities.

(c)

$det(A) = (0)(-1.2) - (1)(1) = -1$

(d)

$det(A^T) = det(A) = -1$

(e)

From the characteristic equation, the product of the eigenvalues = -1.

Q6

18 points
(a) 1 point for characteristic equation, 1 point for eigenvalues, 1 point for each eigenvector, 3 points for general solution, 2 points for determining constants (b) 5 points for phase plot, 1 point for labels, 3 points for explanation of relationship with eigenvectors

(a)

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}$$

The characteristic equation is $(\lambda - 1)(\lambda - 2) - 6 = 0 \implies \lambda^2 - 3\lambda - 4 = 0 \implies \lambda = -1, 4$.
Solving for the eigenvectors, we get $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ for $\lambda = -1$ and $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ for $\lambda = 4$.

So, the analytical solution is of the form $z(t) = k_1 e^{-t} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + k_2 e^{4t} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$. We are given that $z(0) = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$. This gives $k_1 = 1.6, k_2 = -0.8$. So, $z(t) = 1.6e^{-t} \begin{bmatrix} 1 \\ -1 \end{bmatrix} - 0.8e^{4t} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$.

(b)

The phase plot below has been drawn for 81 initial points, i.e., every possible pair of initial conditions from [-4:1:4] has been used. The code for the phase portrait and the state derivative function passed to ODE45 is included.

```
%% Q6

clear all
close all

global A;
A = [1 2; 3 2];

init = -4:1:4;
[z1,z2] = meshgrid(init,init);  % all initial conditions

figure;
hold on

for i = 1:numel(z1)
    [t,z] = ode45(@q6deriv,[0 1],[z1(i) z2(i)]);
    plot(z(:,1),z(:,2));
    plot(z1(i),z2(i),'kx','MarkerSize',10); % Marking initial points
%    za = zeros(size(z));
%    za(:,1) = 1.6*exp(-t) - 1.6*exp(4*t);
%    za(:,2) = -1.6*exp(-t) -2.4*exp(4*t);
%    plot(t,za,'r--');
end

axmin = -10; axmax = 10;
xlim([axmin axmax])
ylim([axmin,axmax])
xlabel('$z_1$','Interpreter','latex')
ylabel('$z_2$','Interpreter','latex')
x = linspace(axmin,axmax,40);   % Plotting eigenvectors
plot(x,1.5*x,'--r', 'Linewidth', 2)
plot(x,-x,'--r', 'Linewidth', 2)
```

```
% state derivative for Q6
function dz = q6deriv(t,z)
    global A;
    dz = A*z;
end
```

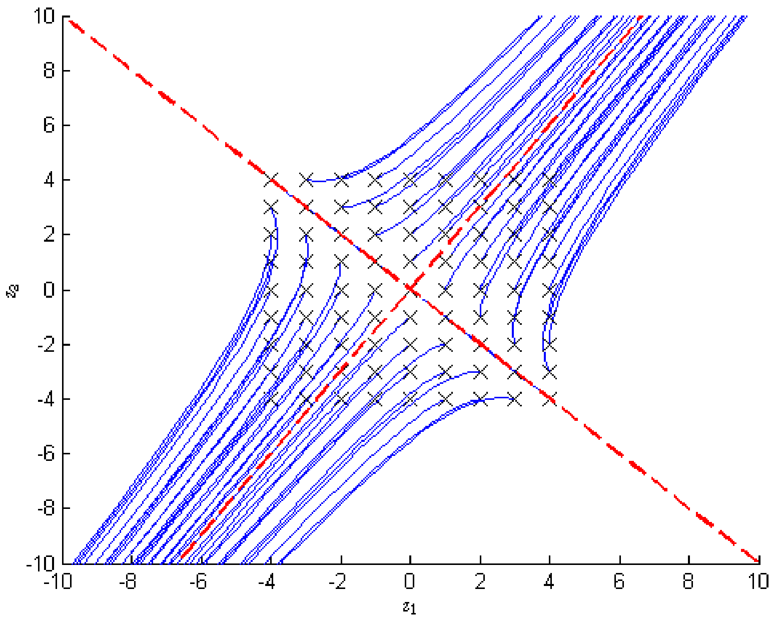


Figure 5: Phase portrait. The dashed lines in red indicate the directions of the eigenvectors.

We can see that the phase-plane curves move away from the eigenvector ($\begin{bmatrix} 1 \\ -1 \end{bmatrix}$) associated with the stable eigenvalue (-1) and align themselves along the eigenvector ($\begin{bmatrix} 2 \\ 3 \end{bmatrix}$) associated with the unstable eigenvalue (4). This is logical as with the progression of time, the terms with e^{4t} dominate over the terms with e^{-t}, so the ratio, $\frac{z_2(t)}{z_1(t)}$ will tend towards the associated eigenvector. For a greater variety of examples, please refer to the uploaded document on phase portraits.

Q7

9 points
3 points for calculating performance metrics, 3 points for estimating them from graph, 1 point for identifying final value, 1 point for validation, 1 point for using step input (initial condition)

Code:

```
%% Q7

clear all
close all

global A B u;

k = 4; c = 0.25; m = 3;
A = [0 1; -k/m -c/m];
B = [0; 1];  % step input
u = 1;

% Numerical solution
[t,z] = ode45(@q7slope,[0,100],[0,0]);
figure;
plot(t,z(:,1),'b')
xlabel('Time (in s)')
ylabel('Position')
```

```

% for settling time
hold on
zl = 0.95*0.75*ones(length(t),1);    % lower bound
zu = 1.05*0.75*ones(length(t),1);    % upper bound
plot(t,zl,'r--')
plot(t,zu,'r--')
```

```
%% function to compute dz/dt for given z, A, B for Q7
% z is a state (column vector)
% dz/dt = A*z


function f = q7slope(t,z)
    global A B u
    f = A*z + B*u;
end
```

The calculate quantities are

$$\begin{aligned}
\beta &= \arctan(\frac{\sqrt{1-\zeta^2}}{\zeta}) = 1.5347 \\
t_r &= \frac{\pi - \beta}{\omega_d} = 1.3925s \\
M_p &= e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} = 0.89276 \\
t_s &= \frac{3}{\zeta\omega_n} = 72s
\end{aligned}$$

Notice that for the step input, the final value would be $\frac{1}{\omega_n^2} = 0.75$. You get this by setting the derivatives $\dot{x}$ and $\ddot{x}$ to zero in the original ODE.

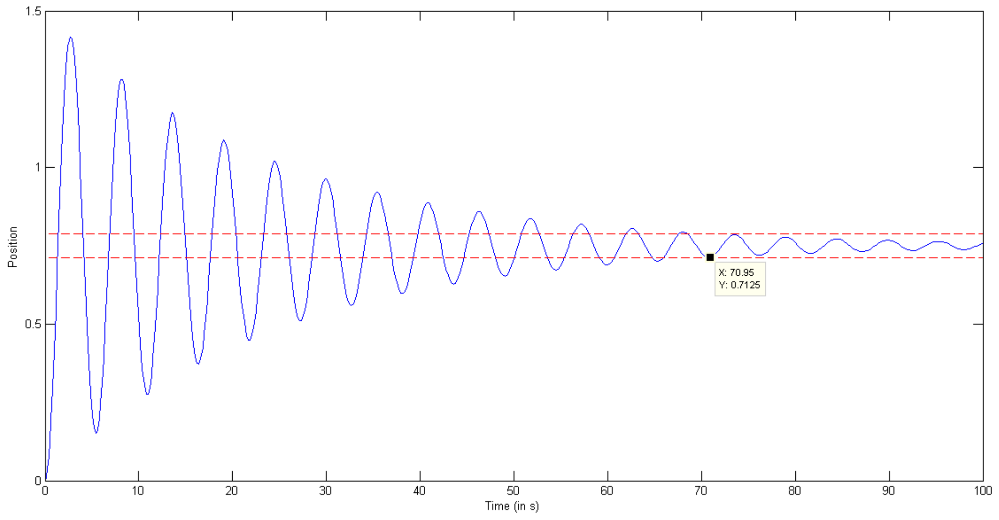


Figure 6: Step response of given spring mass damper system

The rise time in the graph is the time taken for the step response to reach 0.75 for the first time. This is approximately 1.4s. The peak value is 1.417, which is an overshoot of $\frac{1.417-0.75}{0.75} \times 100\% = 88.9\%$. The settling time is the time after which the step response remains between $0.95 * 0.75 = 0.7125$

and $1.05 * 0.75 = 0.7875$. From the graph, this time seems to be 71s.

We can see that the values of the performance metrics estimated from the graph are quite close to those given by the equations. So, these empirical equations serve as good estimates of these performance metrics.