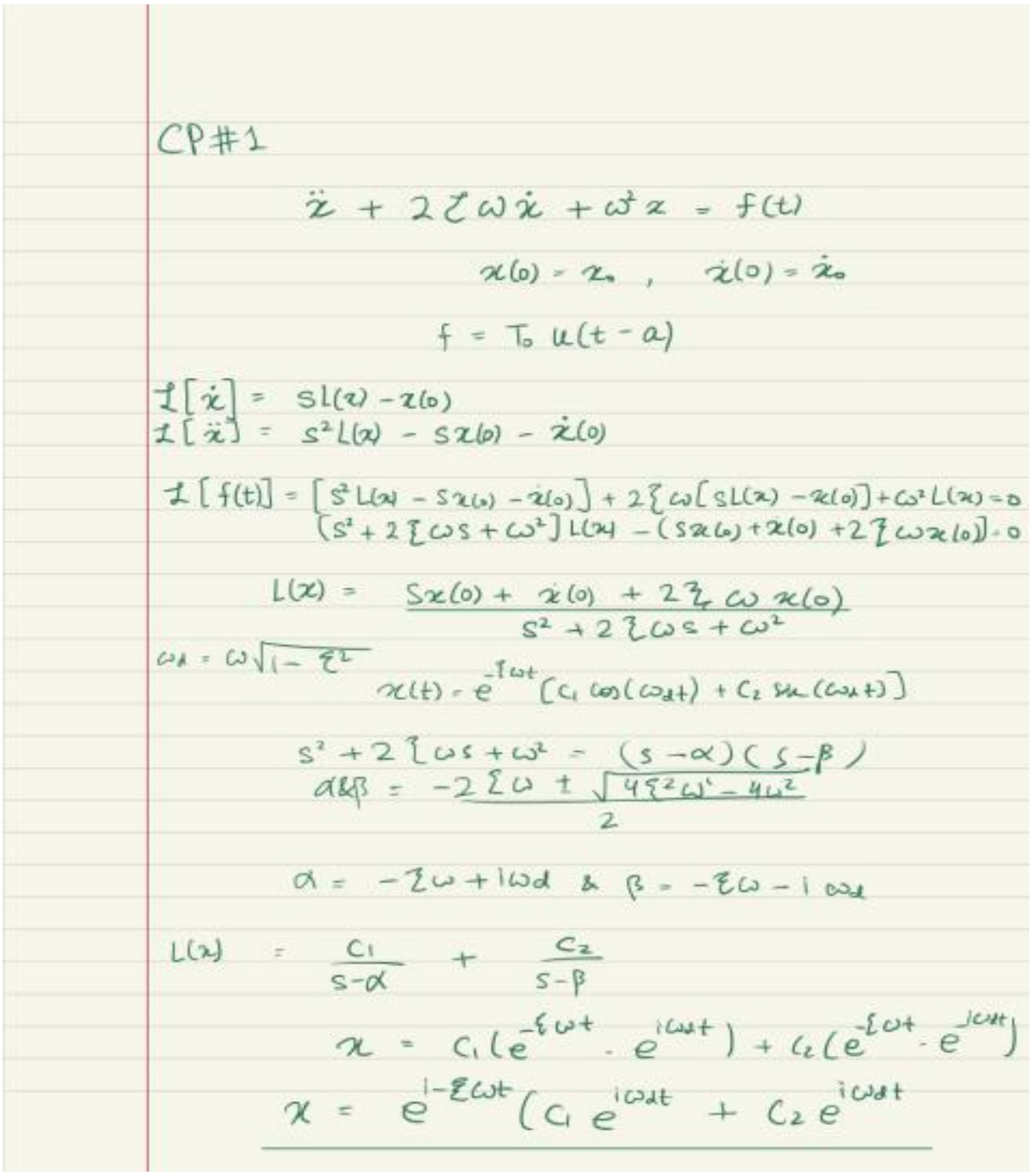
Computer Project 1

09/29/2022

Team members	PSU ID	Contributions
Ankit Gupta	apg5667	Coding
Aspen Stocking	abs6894	Numerical solutions
Siddharth Premnath	sfp5478	Report writing
Nikhil Sharma	nvs5658	Analytical solutions
Kameron Metcalf	kmm8076	Commenting the code

Section 1: Analytical Solutions

1. The scanned paper below presents the solutions to part 1.



CP#1

$$\ddot{x} + 2\zeta\omega\dot{x} + \omega^2 x = f(t)$$

$$x(0) = x_0 \,, \quad \dot{x}(0) = \dot{x}_0$$

$$f = T_0\, u(t-a)$$

$$\mathcal{L}[\dot{x}] = sL(x) - x(0)$$
$$\mathcal{L}[\ddot{x}] = s^2L(x) - sx(0) - \dot{x}(0)$$

$$\mathcal{L}[f(t)] = [s^2L(x) - sx(0) - \dot{x}(0)] + 2\zeta\omega[sL(x) - x(0)] + \omega^2L(x) = 0$$
$$[s^2 + 2\zeta\omega s + \omega^2]L(x) - (sx(0) + \dot{x}(0) + 2\zeta\omega x(0)) = 0$$

$$L(x) = \frac{sx(0) + \dot{x}(0) + 2\zeta\omega x(0)}{s^2 + 2\zeta\omega s + \omega^2}$$

$$\omega_d = \omega\sqrt{1-\zeta^2}$$

$$x(t) = e^{-\zeta\omega t}[C_1\cos(\omega_d t) + C_2\sin(\omega_d t)]$$

$$s^2 + 2\zeta\omega s + \omega^2 = (s-\alpha)(s-\beta)$$
$$\alpha \& \beta = \frac{-2\zeta\omega \pm \sqrt{4\zeta^2\omega^2 - 4\omega^2}}{2}$$

$$\alpha = -\zeta\omega + i\omega_d \quad \& \quad \beta = -\zeta\omega - i\omega_d$$

$$L(x) = \frac{C_1}{s-\alpha} + \frac{C_2}{s-\beta}$$

$$x = C_1(e^{-\zeta\omega t} \cdot e^{i\omega_d t}) + C_2(e^{-\zeta\omega t} \cdot e^{-i\omega_d t})$$

$$x = e^{1-\zeta\omega t}(C_1 e^{i\omega_d t} + C_2 e^{i\omega_d t}$$

2. The scanned page below contains the answer to part 2

1.2)

$X(0)=\dot{X}(0)=0$ $\quad \ddot{x}+2\zeta\omega\dot{x}+\omega^2x=T_0u(t-a)$

$$\left[s^2X(s)-s\overset{0}{\cancel{\dot{x}0}}-\overset{0}{\cancel{x(0)}}\right]+2\zeta\omega\left[sX(s)-\overset{0}{\cancel{X(0)}}\right]+\omega^2X(s)=\frac{T_0e^{-as}}{s}$$

$$X(s)\left[s^2+2\zeta\omega s+\omega^2\right]=\frac{T_0e^{-as}}{s}$$

let $-\zeta\omega=\alpha$ and $\beta=\omega\sqrt{1-\zeta^2}$ $\quad \alpha^2+\beta^2=\zeta^2\omega^2+\omega^2(1-\zeta^2)=\omega^2$

$$X(s)\left[s^2-2\alpha s+\alpha^2+\beta^2\right]=\frac{T_0e^{-as}}{s}$$

$$X(s)\left[(s-\alpha)^2+\beta^2\right]=\frac{T_0e^{-as}}{s} \qquad X(s)=T_0\left[\frac{e^{-as}}{s\left[(s-\alpha)^2+\beta^2\right]}\right]$$

$$\Rightarrow X(s)=T_0e^{-as}\left[\frac{As+B}{s}+\frac{C(s-\alpha)+D}{(s-\alpha)^2+\beta}\right]$$

$$(As+B)\left[(s-\alpha)^2+\beta^2\right]+\left[C(s-\alpha)+D\right]s=1$$

$$(As+B)\left[s^2-2\alpha s+\underbrace{\alpha^2+\beta^2}_{\omega^2}\right]+\left[C(s-\alpha)+D\right]s=1$$ if $s=0$, $B\omega^2=1$

$B=\frac{1}{\omega^2}$

$$As^3-2\alpha As+As\omega^2+Bs^2-2\alpha Bs+B\omega^2+Cs^2-C\alpha s+Ds=1$$

matching coefficients

$As^3=0$

$A=0$ $\quad$ now, $Bs^2-2\alpha Bs+B\omega^2+Cs^2-C\alpha s+Ds=1$

3. The scanned page below contains the answer to part 3

$$Bs^2 - 2\alpha Bs + B\omega^2 + Cs^2 + C\alpha s + Ds = 1$$

becomes

$$(B+C)s^2 + [-2\alpha B - C\alpha + D]s + B\omega^2 = 1$$

$B+C=0$ $\quad C=-B \quad$ remember $B=\frac{1}{\omega^2}$

$-2\alpha B - C\alpha + D = 0 \Rightarrow \frac{-2\alpha}{\omega^2} + \frac{\alpha}{\omega^2} + D = 0 \quad D = \frac{\alpha}{\omega^2} \quad C = \frac{-1}{\omega^2}$

SO

$$\Rightarrow X(s) = T_0 e^{-as}\left[\frac{1}{\omega^2 s} - \frac{1(s-\alpha)+\alpha}{\omega^2[(s-\alpha)^2+\beta^2]}\right]$$

$$= \frac{T_0}{\omega^2}\left[\frac{e^{-as}}{s} - \frac{e^{-as}(s-\alpha)}{(s-\alpha)^2+\beta^2} + \frac{e^{-as}(\alpha)}{(s-\alpha)^2+\beta^2}\right]$$

$$\boxed{x(t) = \frac{T_0}{\omega^2}\left[u(t-a) - e^{\alpha(t-a)}\cos(\beta t - \beta a)u(t-a)\right]}$$

when t is large enough the sat. will become stable and its acceleration goes to 0

$\boxed{1.3}$ $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad \dot{y} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ -2\zeta\omega\dot{x} - \omega^2 x + f \end{bmatrix}$

$$\dot{y} = \underline{f}(t, y) = \begin{bmatrix} y_2 \\ -2\zeta\omega y^2 - \omega^2 y_1 + f \end{bmatrix}, \quad y(0) = \begin{bmatrix} x(0) \\ \dot{x}(0) \end{bmatrix} = \begin{bmatrix} x_0 \\ \dot{x}_0 \end{bmatrix}$$

Section 2: Numerical Solutions

Part 1- Refer to the Appendix

Part 2 of Section 2

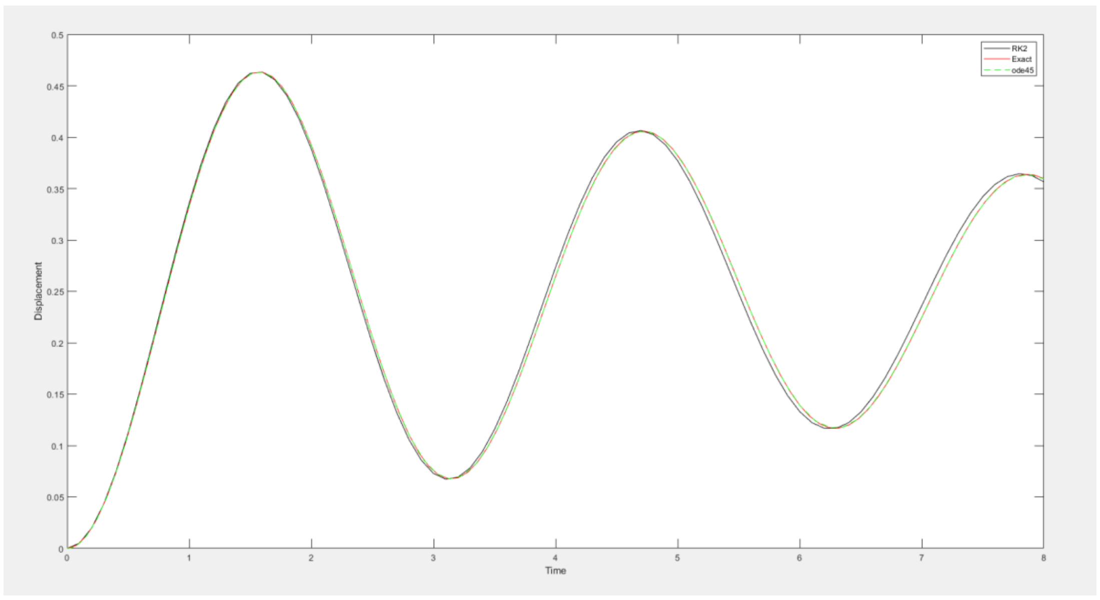


Figure 1 Displacement vs Time graph with all RK2, Exact and ODE45 solutions

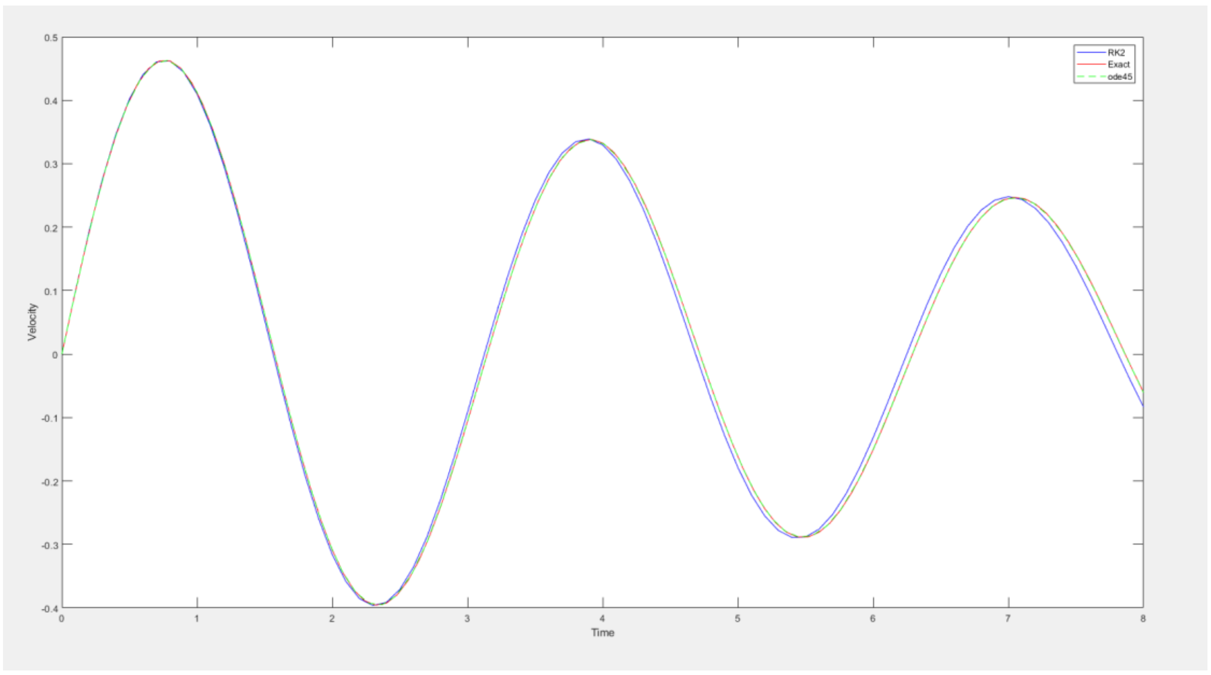


Figure 2 Velocity vs Time graph with RK2, EXACT and ODE45 Solutions

- From the above graphs we can confirm that the all the solutions are very similar. In the displacement and velocity graph the RK2 and Ode solver stick together as time increases while the exact solution continues to break away.

Part 3 of Section 2

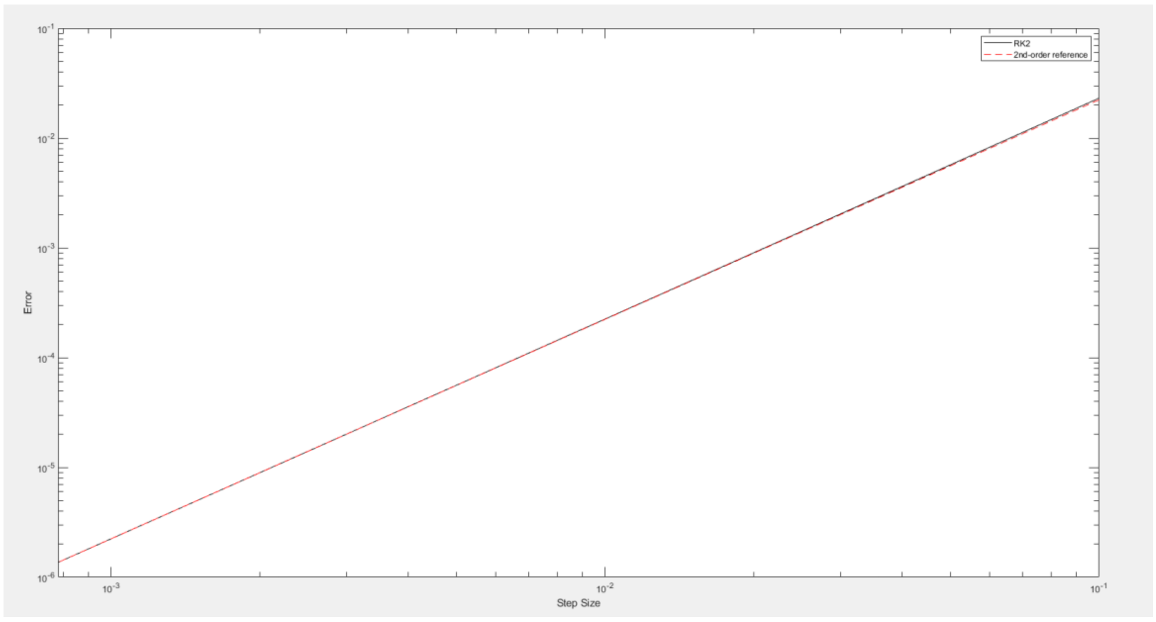


Figure 3 Step Size vs Error graph in LOG-LOG configuration

- From the above graph we can observe that the error of the rk2 function increases as the step size increases in an exponential manner. The second order reference also follows the RK2 function extremely closely.

Section 3: Application of the Computer Program

Part 1 of section 3– refer to appendix

Part 2 of section 3-

First Half– refer to appendix

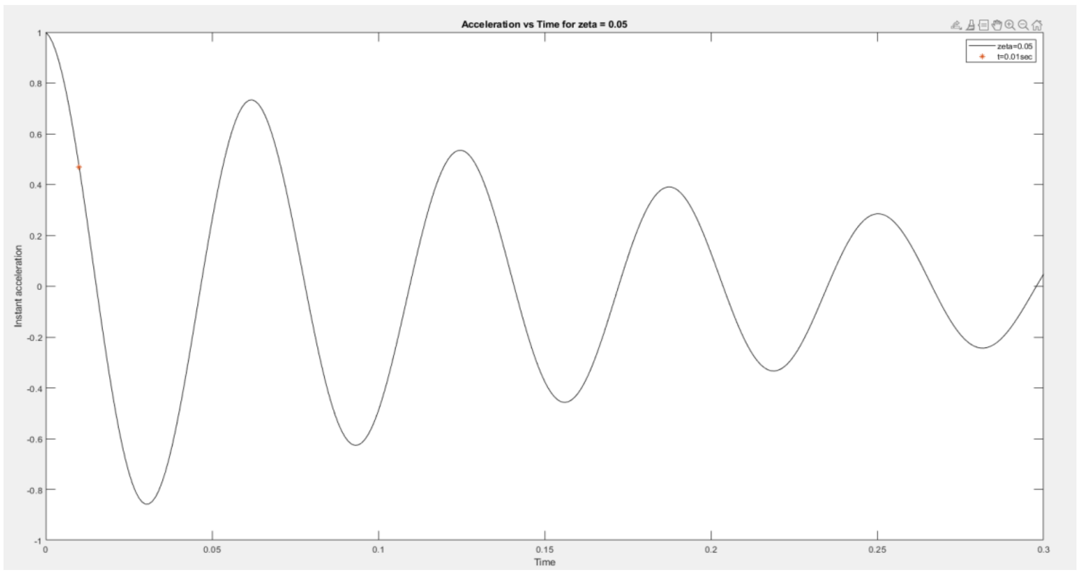


Figure 4 Acceleration vs Time Graph for Zeta=0.05

Second Half-

- The trend we observe is a decrease in amplitude between acceleration and time graph for multiple dampening values. This is due to dampening force removing energy from the system. This causes the satellite to have lower accelerations over time. Also, after the t=0.3, the acceleration is completely dampened.

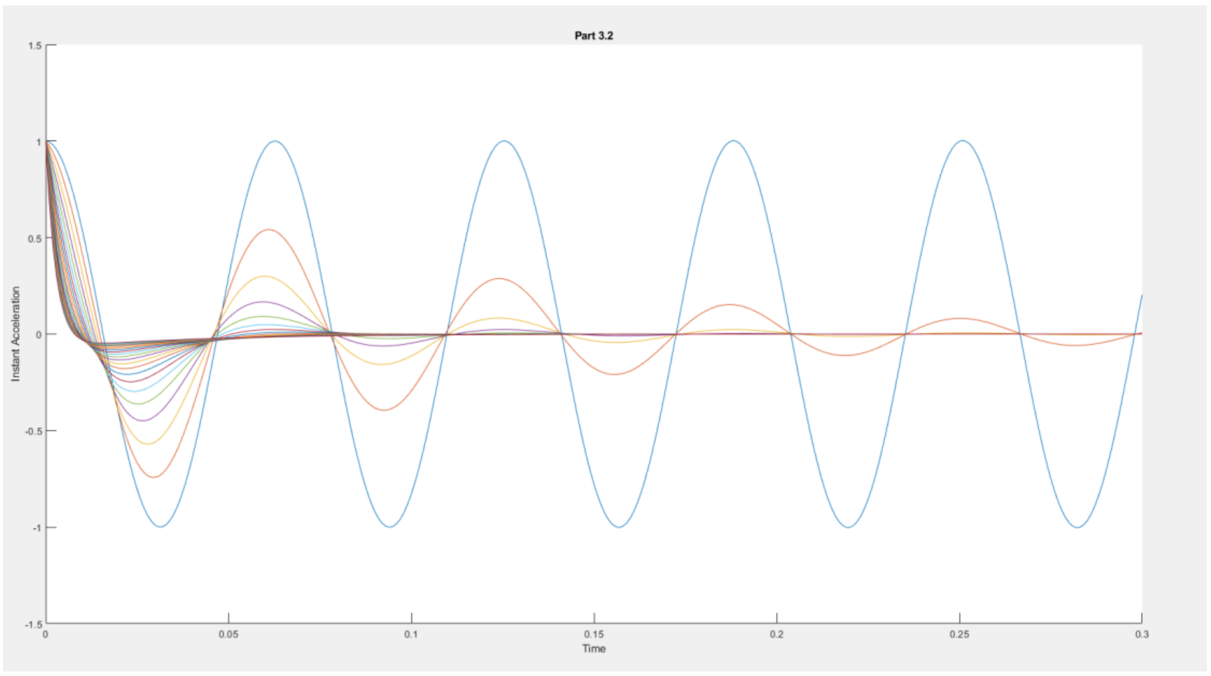


Figure 5 Acceleration vs Time graph for multiple Zeta numbers

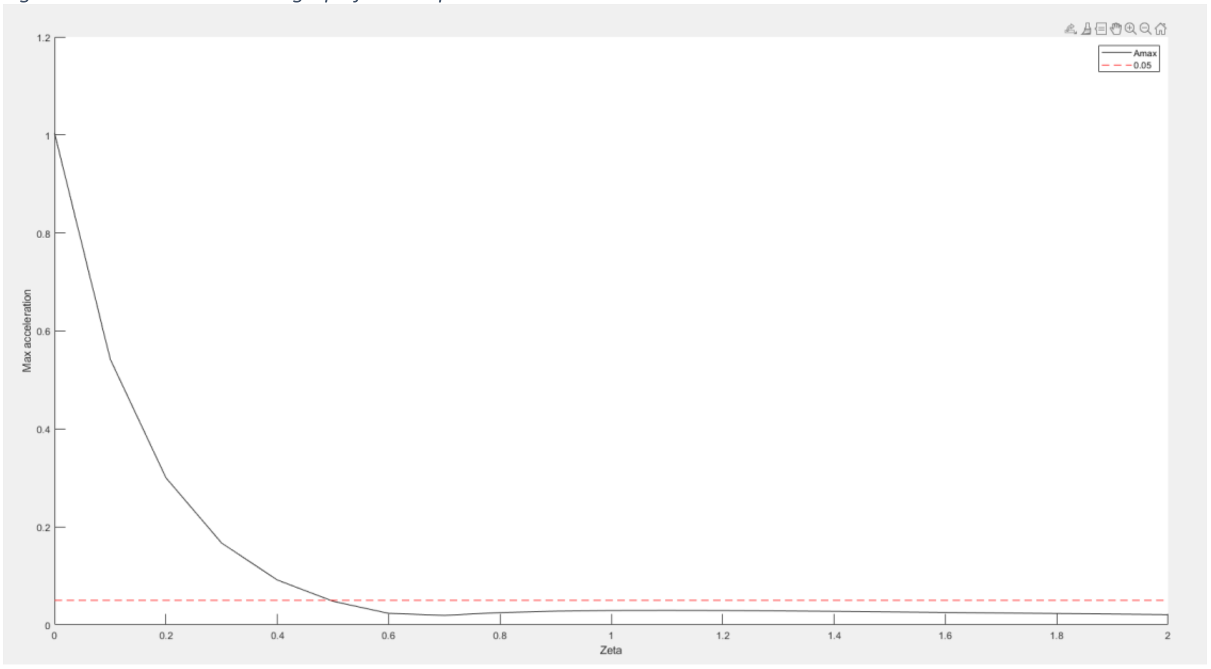


Figure 6 Maximum Acceleration vs Zeta values

Part 3 of section 3

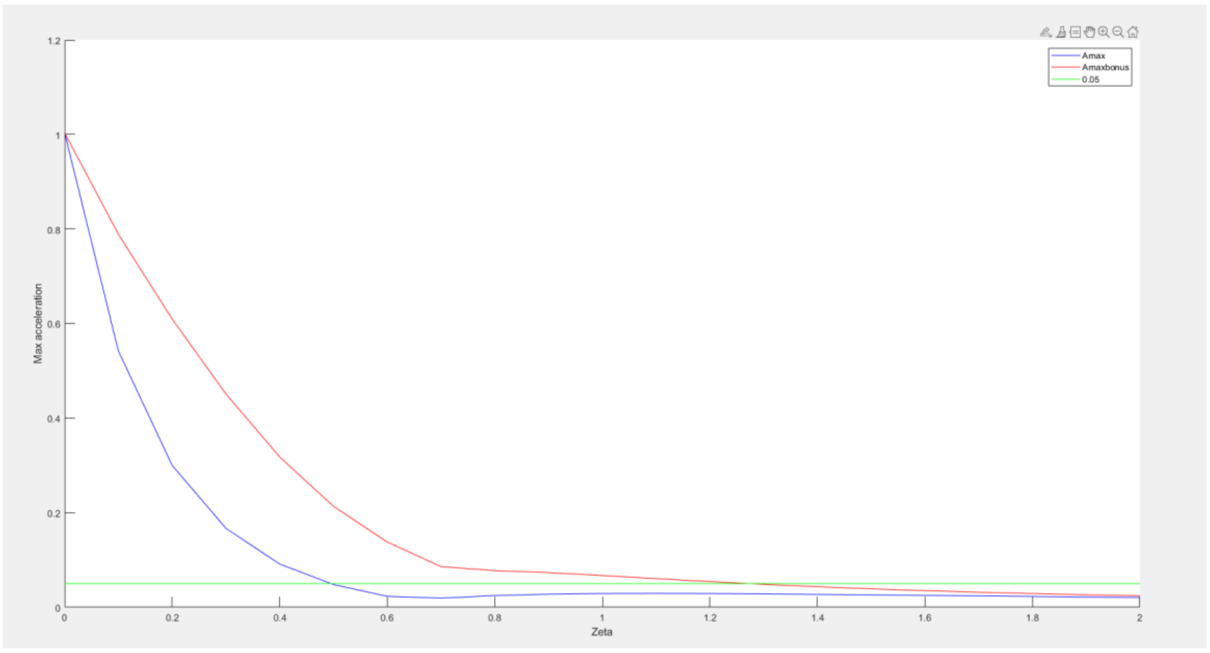


Figure 7 Maximum acceleration Vs Zeta with RK2 function

- From the graph above, we can observe that the range of zeta functions has reduced and starts from 1.3 onwards.

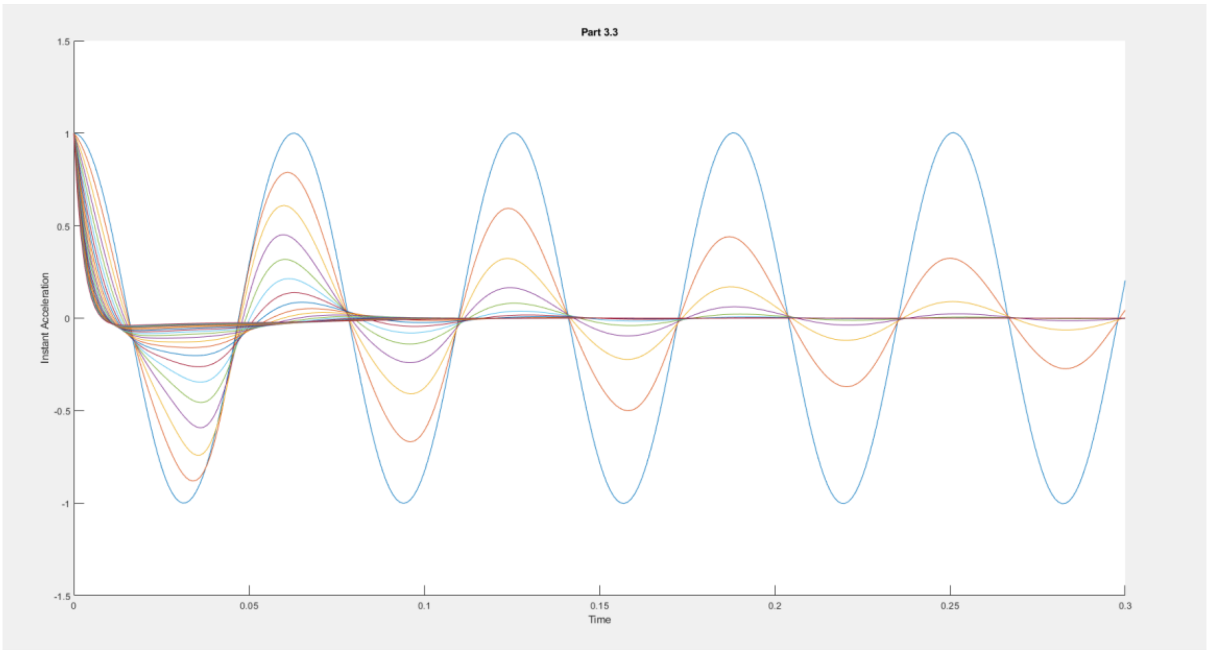


Figure 8 Acceleration vs Time using RK2 function

Section 4: Summary

In this project we analysed the second order differential equations pertaining to the dampening function of a satellite. The project was split into three parts: Analytical solutions, Numerical solutions and the Satellite mount design. The analytical solution part required substantial team collaboration due to difficulty of problems. Part 2 and 3 caused the most trouble due to the ambiguity of the questions and required multiple opinions on what the question meant in our discussions. This was a recurring theme in the numerical solutions. The ambiguity of the questions would leave our group dumbfounded on where we should start at. Thus, the coding aspect of this part required multiple discussions and needed to go to office hours to clear our doubts. We set small goals to offset this problem to see our progression with this project. The design aspect of the project was straightforward after we had figured out the solutions to the numerical problems. The code for the Runge Kutta method was modified for this part with additional code for inputting the values for zeta and omega. This project helped us understand how to tackle a large coding project with a team and how to implement MATLAB to solve numerical problems.

Appendix: The Computer Program

Part 2

```
clc; % Clear command line
clear; % Clear workspace variables
close all % Closes open figures
%Part II
z1 = 0.05; % Zeta for part 2
o1 = 2; % Omega for part 2
T0 = 1; % T0 for part 2
%{
 Define the f(t, y) function for Part 2.
 Input:
 t: Time step, a scalar
 y: Numerical solution of the current step, 2-element vector,
 [displacement, velocity]
 Output:
 A 2-element vector as derivative of y
%}
f1 = @(t,y) [y(2); -2*z1*o1*y(2) - o1^2*y(1) + T0*1]; % Function to be
 %integrated for part 2
%----------------------------------------------------------------------
% Set up for Part 2
%----------------------------------------------------------------------
tf = 8; % Final time for part 2
y0 = [0.,0.]; % Initial condition
wd = o1*sqrt(1-(z1)^2); % Damped frequency constant
time = linspace(0,tf,101); % Array of 101 time stations with a time step size of 0.08
b = -z1*o1; % Coefficient (to decrease length of expressions for a1 and a2)
C1 = T0/(b^2 + wd^2); % Another coefficient
% Exact solution for displacement
a1 = C1.*(1+b/wd.*exp(b.*time).*sin(wd.*time) - exp(b.*time).*cos(wd.*time));
% Exact solution for velocity
a2 = exp(b.*time)/wd.*sin(wd.*time);
%----------------------------------------------------------------------
% Part 2(1) & (2)
%----------------------------------------------------------------------
```

```
% This generates the numerical solution using RK2
[t2,y2] = rk2(f1,y0,tf,0.1);
% This generates the numerical solution using ode45
[tode45,yode45] = ode45(f1, [0 8], [0 0]);
%-----------------------------------------------------------------------
%Process and plot the results
%-----------------------------------------------------------------------

g1 = figure(1);
graph = plot(t2, y2(:,1), 'black',time, a1, '-red',tode45,yode45(:,1),'--green');
legend([graph(1) graph(2) graph(3)],{'RK2', 'Exact','ode45'});
ylabel('Displacement')
xlabel('Time')
g12 = figure(2);
graph = plot(t2, y2(:,2), 'blue',time, a2, '-red',tode45,yode45(:,2),'--green');
legend([graph(1) graph(2) graph(3)],{'RK2','Exact','ode45'});
ylabel('Velocity');
xlabel('Time');
%-----------------------------------------------------------------------
% Part 2(3)
%-----------------------------------------------------------------------
N = 8; % The number of step sizes to check
yf = [ a1(end); a2(end) ]; % Exact solution at last time step
h = zeros(N,1); % Allocate the array for step sizes
err = zeros(N,1); % Allocate the array for errors.
for i = 1:N
 graph = 0.1/2^(i-1); % Halving the step size each time
 [~, y] = rk2(f1, y0, tf, graph); % Ignores the first input argument in rk2
 h(i) = graph;
 err(i) = norm( y(end,:)' - yf); % Distance between points y and yf
end
error = err(end)*(h/h(end)).^2; % Computing error
g3 = figure(3);
graph = loglog(h, err, 'black-',h, error, '--red'); % loglog will plot the data using a base 10 logarithmic
scale for both x and y axis.
legend([graph(1), graph(2)], {'RK2', '2nd-order reference'})
xlabel('Step Size')
ylabel('Error')
```

Part 3

```
z3 = 0.05; % Initial zeta for part 3
o3 = 100; % Omega for part 3
h = 0.001; % Time step for part 3
f3 = @(t,y) [y(2); -2*z3*o3*y(2) - o3^2*y(1) + T0*1]; % Function to be integrated for part 3
%--------------------------------------------------------------------------
% Setup for Part 3
%--------------------------------------------------------------------------
y0 = [0.,0.]; % Time step for Part 3
T0 = 1; % T0 for part 3
[t3,A3] = rk23(y0,tf3,h,z3);
% Plots
g4 = figure(4);
graph = plot(t3, A3,'black',0.01,0.47,'*');
ylabel('Instant acceleration');
xlabel('Time');
title('Acceleration vs Time for zeta = 0.05');
legend('zeta=0.05','t=0.01sec');
%--------------------------------------------------------------------------
% 3.2
%--------------------------------------------------------------------------
zmax = 2; % Max zeta value
dz = 0.1; % Time steps for zeta values
N = int32(ceil(zmax/dz))+1; % Number of points for zeta
zeta = linspace(0,zmax,N); % Zeta values
tf3 = 0.3; % Final time span (based on my intuition)
y0 = [0.,0.]; % Initial conditions
Amax = zeros(1,N);
for i = 1:N
 [t32,A32] = rk23(y0,tf3,h,zeta(i));
 % Plot
 g5 = figure(5);
 hold on
 graph(i) = plot(t32, A32);
 ylabel('Instant Acceleration');
 xlabel('Time');
 title('Part 3.2');
 Amax(i) = max(abs(A32(50:length(A32))));
end
g6 = figure(6);
hold on
graph = plot(zeta,Amax,'black',zeta,0.05*ones(length(Amax)),'--red');
xlabel('Zeta');
ylabel('Max acceleration')
legend('Amax','0.05')

% 3.3
%--------------------------------------------------------------------------
Amax2 = zeros(1,N);
for i = 1:N
 % numerical solution using RK23
 [t33,A33] = rk2bonus(y0,tf3,h,zeta(i));
 % Plot
 g7 = figure(7);
 hold on
 graph(i) = plot(t33, A33);
 ylabel('Instant Acceleration');
 xlabel('Time');
 title('Part 3.3');
 Amax2(i) = max(abs(A33(50:length(A33))));
end
g8 = figure(8);
hold on
plot(zeta,Amax,'-blue',zeta,Amax2,'-red',zeta,0.05*ones(length(Amax)),'-green')
xlabel('Zeta');
ylabel('Max acceleration')
legend('A{max}','A{max}bonus','0.05')
```

RK2

```
function [t, y] = rk2(f, y0, tf, h)

    n = int32(ceil(tf/h))+1;  % determine the number of steps
    t = linspace(0, tf, n);   % generate the time step vector
    y = zeros(n,2);           % allocate the array for numerical solutions
    y(1,:) = y0';             % The first row of y is the initial condition

    for ii = 1:(n-1)                              % Loop over the time steps
        k1 = f( t(ii), y(ii,:)' );                % The first slope at the current step
        k2 = f(t(ii) + h/2, y(ii,:)'+h*k1/2);          % The second slope at the middle point
        y(ii+1,:) = y(ii,:) + h*k2';                % Compute the next step
    end
end
```

RK23

```
function [t, A] = rk23(y0, tf, h, zeta)
n = ceil(tf/h) + 1;   % determine the number of steps
t = linspace(0,tf,n); % generate the time step vector
y = zeros(n,2);       % allocate the array for numerical solutions
A = zeros(n,1);       % allocate the array for acceleration values
y(1,:) = y0';         % The first row of y is the initial condition

o3 = 100;             % Omega for part 3
T0 = 1;               % T0 for part 3
A(1) = T0*1.0;        % initial value for acceleration

f = @(t,y) [y(2); -2*zeta*o3*y(2) - o3^2*y(1) + T0*1];   % Function to be integrated

for i = 1:(n-1)
    k1 = f(t(i),y(i,:)');
    k2 = f(t(i) + 0.5.*h, y(i,:)' + 0.5.*h.*k1);
    y(i+1,:) = transpose(y(i,:)' + h.*k2);.
    tp = f(t(i+1),y(i+1,:)');
    A(i+1) = tp(2);
end
end
```

RK2 Bonus

```
function [t, A] = rk2bonus(y0, tf, h, zeta)

n = ceil(tf/h) + 1;   % determine the number of steps
t = linspace(0,tf,n); % generate the time step vector
y = zeros(n,2);       % allocate the array for numerical solutions
A = zeros(n,1);       % allocate the array for acceleration values
y(1,:) = y0';         % The first row of y is the initial condition

o3 = 100;             % Omega for part 3
T0 = 1;               % T0 for part 3
k = 5e7;              % k for part 3.3

A(1) = T0*1.0;            % initial condition for acceleration
f = @(t,y) [y(2); -2*zeta*o3*(1-k*y(1)^2)*y(2) - o3^2*y(1) + T0*1];
for i = 1:(n-1)
    k1 = f(t(i),y(i,:)');
    k2 = f(t(i) + 0.5.*h, y(i,:)' + 0.5.*h.*k1);
    y(i+1,:) = transpose(y(i,:)' + h.*k2);
    tp = f(t(i+1),y(i+1,:)');
    A(i+1) = tp(2);
end
end
```