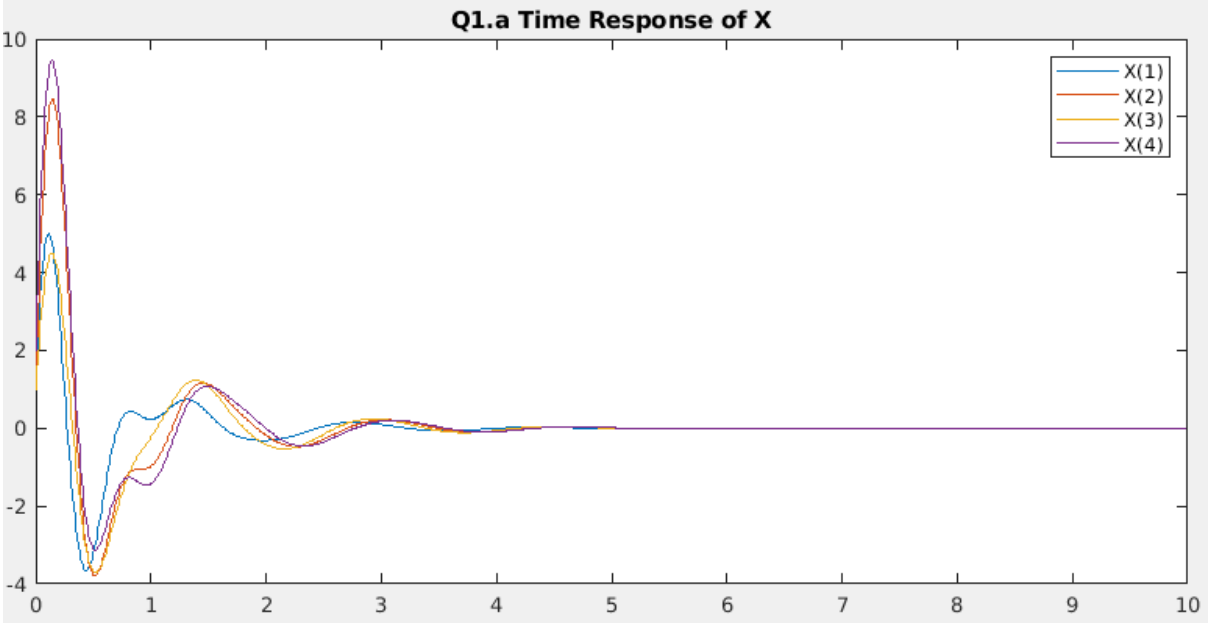
Joshua Benfell 300433229

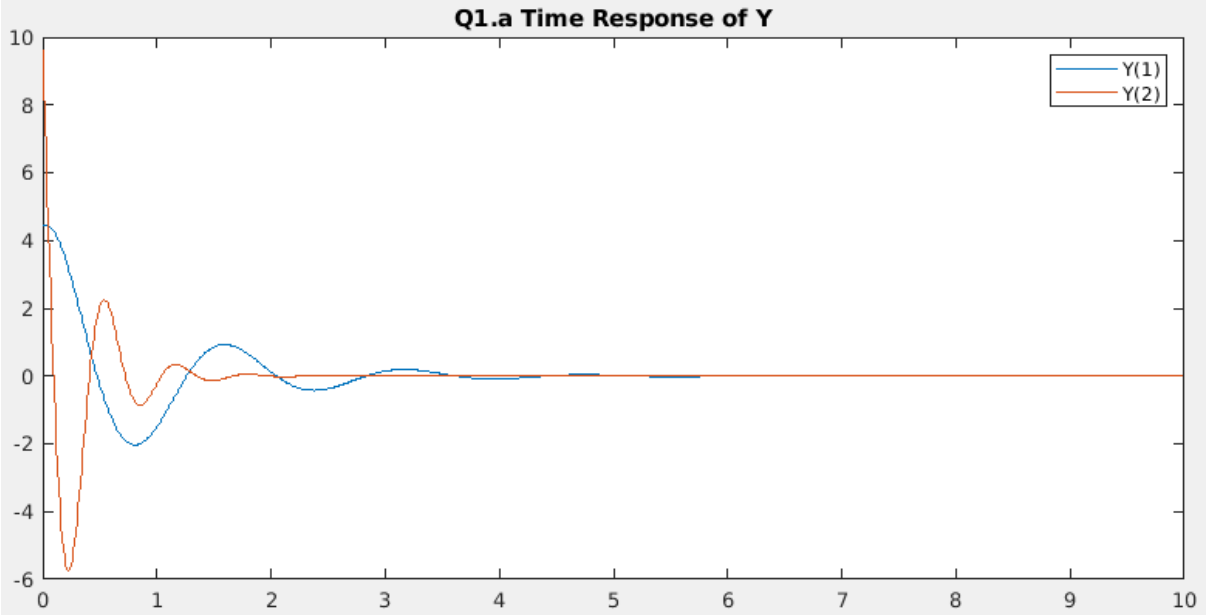
Assignment 2

A. Formative

1.a) Plotting with Lsim

Graphs





Look Graphs do the thing!! They plot with lsim

Joshua Benfell 300433229

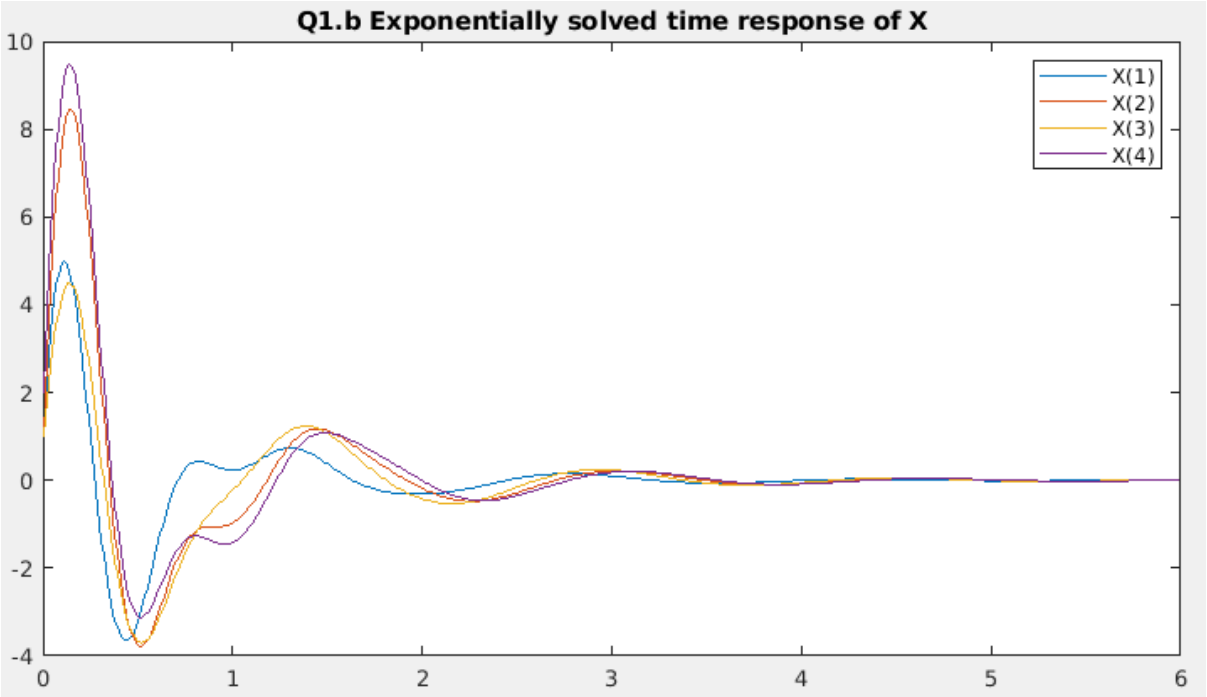
Matlab Code

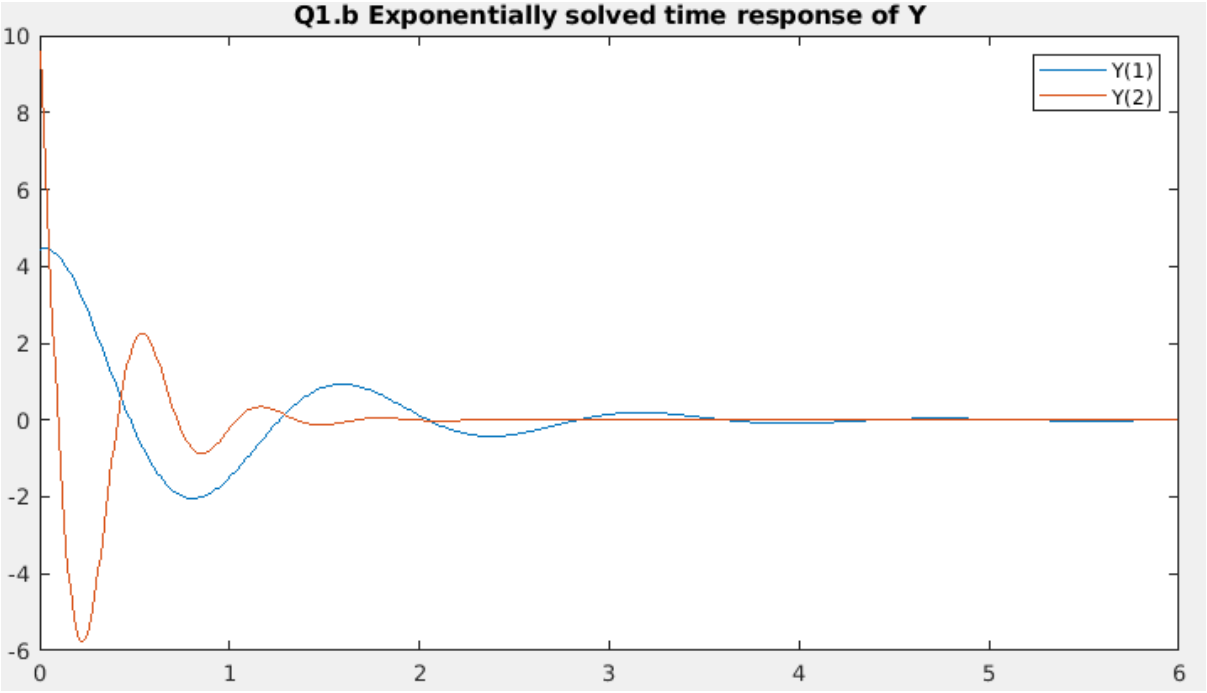
```
close all
clear
% Create System
A = [12.5314 , -91.36 , 28.7129 , 59.9628 ;
    21.316 , -115.6631 , 37.5584 , 75.0519 ;
    13.967 , -53.5817 , 15.4161 , 33.4391 ;
    20.5222 , -123.3107 , 42.267 , 79.7156]; %4x4
B = [1.7649 , 1.7649;
    2.8318 , 2.8318;
    2.2353 , 2.2353;
    2.7294 , 2.7294]; %2x4
C = [-0.46166 , -4.4635 , 1.9151 , 3.7274 ;
     1.0853 , -12.82 , 4.5753 , 8.3759]; %4x2
 D = [0 , 0 ;
    0 , 0 ]; %2x2
sys = ss(A,B,C,D);
% End Create System
T = 0:0.01:6;
X0 = [1, 1, 1, 2].';  % Column vector of initial state
U = zeros(2, length(T));  % Set up with no inputs 2x1001
[Y, Tsim, X] = lsim(sys, U, T, X0);
figure(1)
plot(Tsim, Y);
title("Q1.a Time Response of Y");
legend("Y(1)", "Y(2)")
figure(2)
plot(Tsim, X);
title("Q1.a Time Response of X");
legend("X(1)", "X(2)", "X(3)", "X(4)")
```

Joshua Benfell 300433229

1.b) Plotting with Matrix Exponential

Graphs





Look Graphs do the thing again but plotted through the matrix exponential. It looks the same as the lsim one, fancy that considering it's autonomous.

Joshua Benfell 300433229

Matlab Code

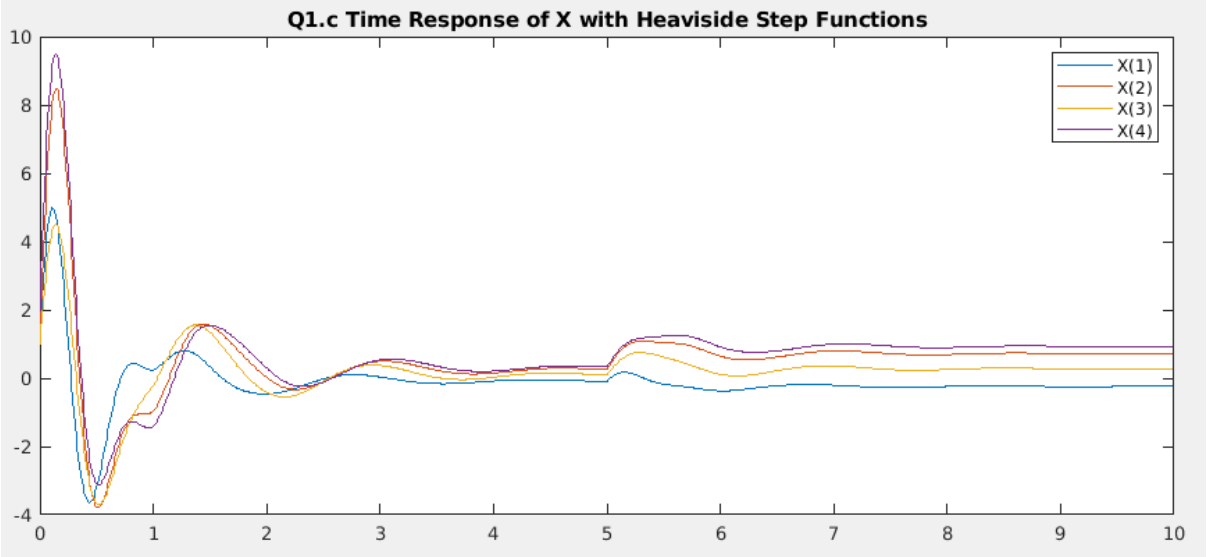
```
A = [12.5314 , -91.36 , 28.7129 , 59.9628 ;
    21.316 , -115.6631 , 37.5584 , 75.0519 ;
    13.967 , -53.5817 , 15.4161 , 33.4391 ;
    20.5222 , -123.3107 , 42.267 , 79.7156]; %4x4
B = [1.7649 , 1.7649;
    2.8318 , 2.8318;
    2.2353 , 2.2353;
    2.7294 , 2.7294]; %2x4
C = [-0.46166 , -4.4635 , 1.9151 , 3.7274 ;
     1.0853 , -12.82 , 4.5753 , 8.3759]; %4x2
 D = [0 , 0 ;
    0 , 0 ]; %2x2
% End Create System
T = 0:0.01:6;
T_idx = 2:1:length(T);
X0 = [1, 1, 1, 2].';  % Column vector of initial state
X = zeros(4,length(T));
X(:,1) = X0;
% As there is no input, it is autonomous, so we can ignore B and D
for t = T_idx
   X(:,t) = expm((T(t) - T(t-1))*A)*X(:,(t-1));
end
figure(3)
plot(T,X)
title("Q1.b Exponentially solved time response of X")
legend("X(1)", "X(2)", "X(3)", "X(4)")
Y = C * X;
figure(4)
plot(T,Y)
title("Q1.b Exponentially solved time response of Y")
legend("Y(1)", "Y(2)")
```

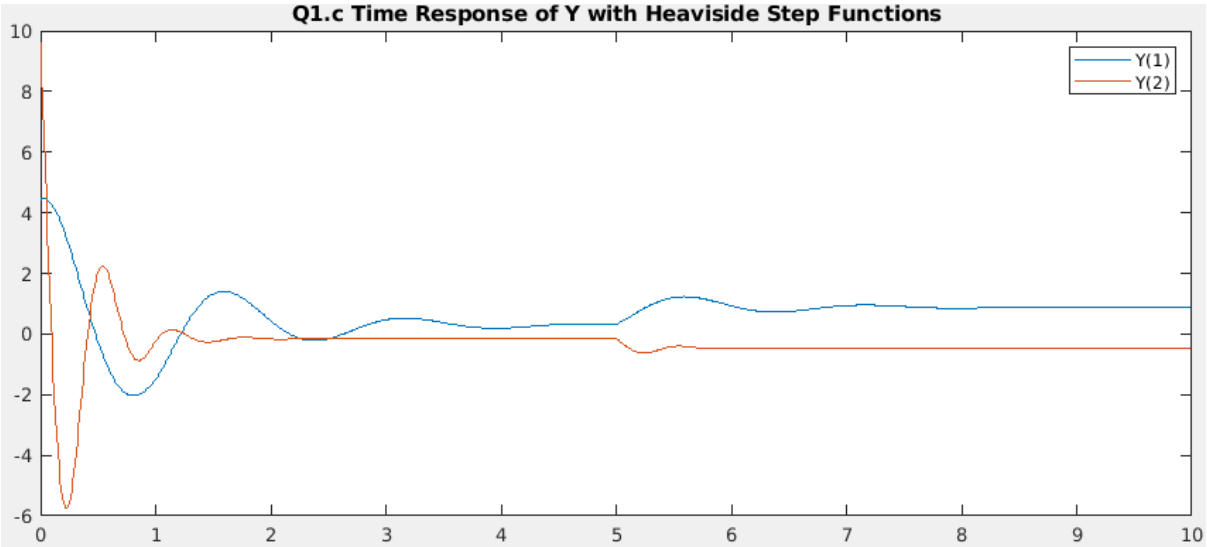
Joshua Benfell 300433229

1.c) Plotting response to input with lsim

Graphs

With the previously defined initial conditions

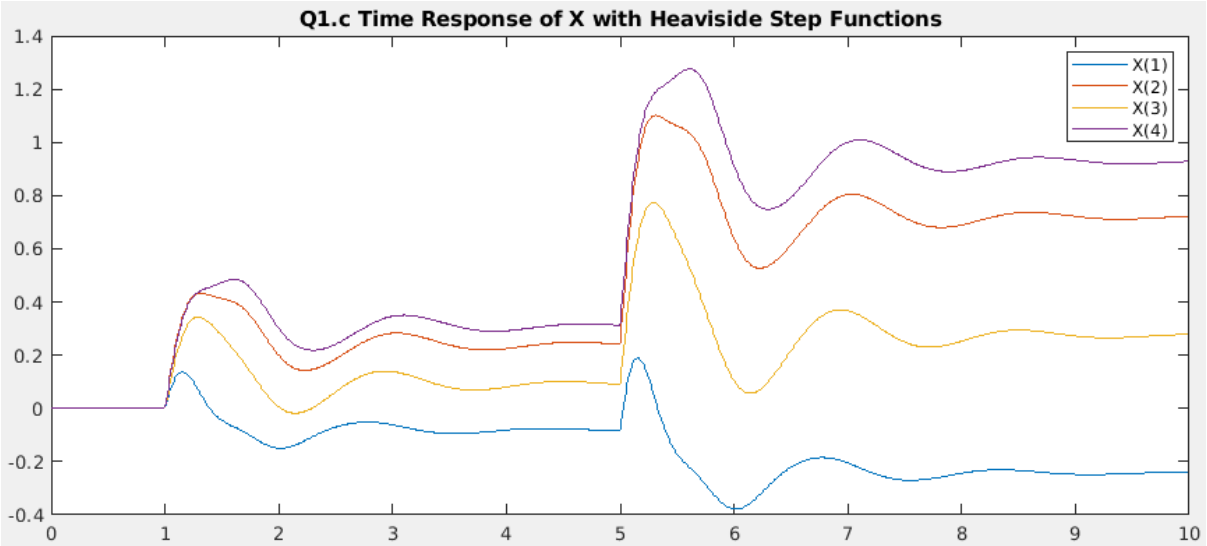


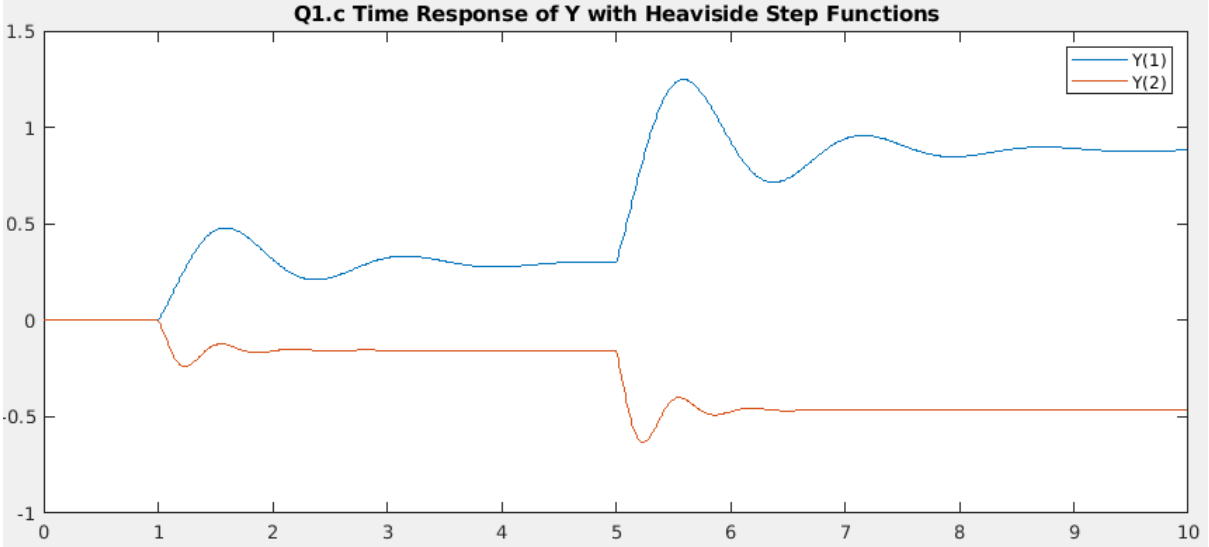


From this

Joshua Benfell 300433229

No Initial Conditions





Matlab Code

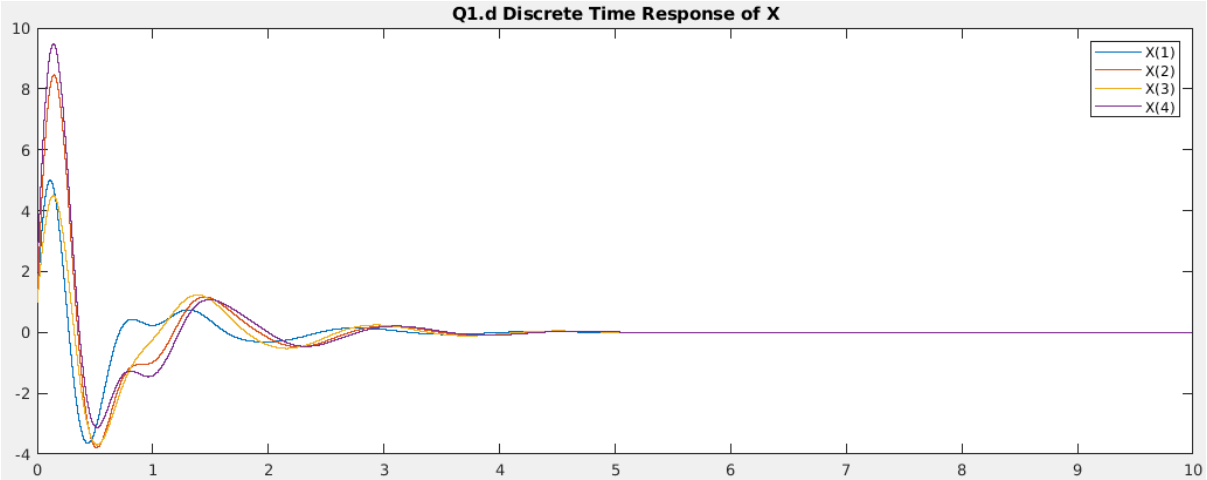
```
A = [12.5314 , -91.36 , 28.7129 , 59.9628 ;
    21.316 , -115.6631 , 37.5584 , 75.0519 ;
    13.967 , -53.5817 , 15.4161 , 33.4391 ;
    20.5222 , -123.3107 , 42.267 , 79.7156]; %4x4
B = [1.7649 , 1.7649;
    2.8318 , 2.8318;
    2.2353 , 2.2353;
    2.7294 , 2.7294]; %2x4
C = [-0.46166 , -4.4635 , 1.9151 , 3.7274 ;
     1.0853 , -12.82 , 4.5753 , 8.3759]; %4x2
```

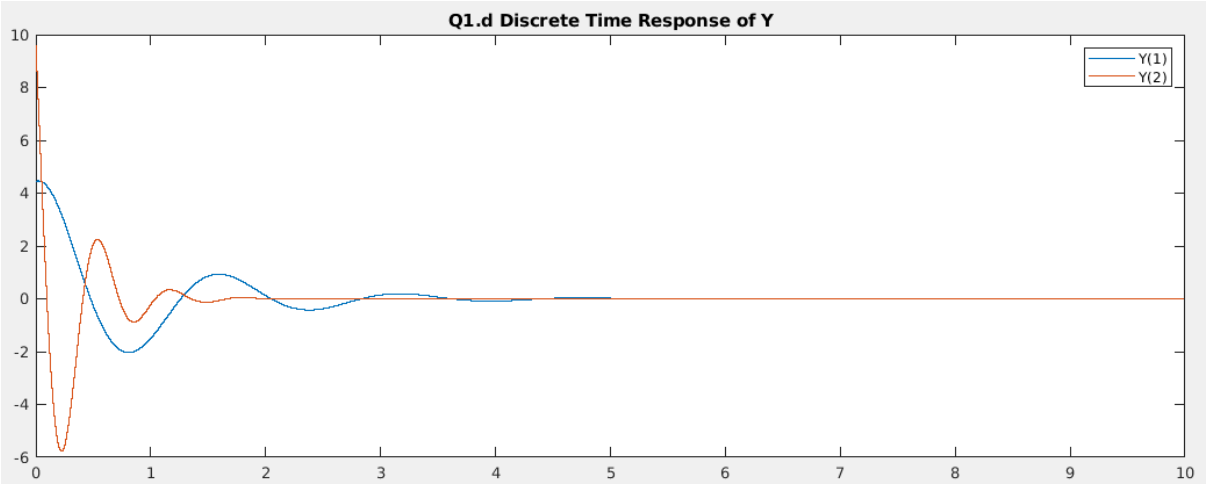
```
 D = [0 , 0 ;
    0 , 0 ]; %2x2
sys = ss(A,B,C,D);
% End Create System
T = 0:0.01:10;
X0 = [1, 1, 1, 2].';  % Column vector of initial state
U = [heaviside(T-1);
    2*heaviside(T-5)];  % Set up with heaviside inputs
[Y, Tsim, X] = lsim(sys, U, T, X0);
figure(5)
plot(Tsim,X)
title("Q1.c Time Response of X with Heaviside Step Functions");
legend("X(1)", "X(2)", "X(3)", "X(4)")
figure(6)
plot(Tsim,Y)
title("Q1.c Time Response of Y with Heaviside Step Functions");
legend("Y(1)", "Y(2)")
```

Joshua Benfell 300433229

1.d) Zero Order Hold discretisation

Graphs





With a time step of 0.001, you can't tell the difference and it looks the same as if it was continuous. You can't tell if your time step is small enough.

Joshua Benfell 300433229

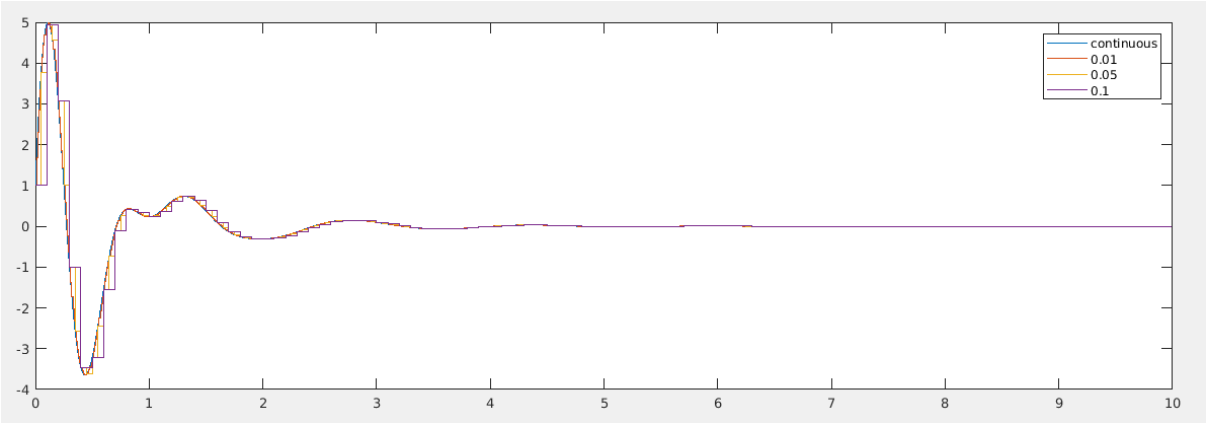
Matlab Code

```
clear
% Create System
A = [12.5314 , -91.36 , 28.7129 , 59.9628 ;
    21.316 , -115.6631 , 37.5584 , 75.0519 ;
    13.967 , -53.5817 , 15.4161 , 33.4391 ;
    20.5222 , -123.3107 , 42.267 , 79.7156]; %4x4
B = [1.7649 , 1.7649;
    2.8318 , 2.8318;
    2.2353 , 2.2353;
    2.7294 , 2.7294]; %2x4
C = [-0.46166 , -4.4635 , 1.9151 , 3.7274 ;
     1.0853 , -12.82 , 4.5753 , 8.3759]; %4x2
 D = [0 , 0 ;
    0 , 0 ]; %2x2
sys = ss(A,B,C,D);
timeStep = 0.0001;
T = 0:timeStep:10;
% End Create System
sysd = c2d(sys, timeStep);
X0 = [1, 1, 1, 2].';  % Column vector of initial state
U = zeros(2, length(T));  % Set up with no inputs 2x1001
[Y, Tsim, X] = lsim(sysd, U, T, X0);
figure(7)
stairs(Tsim, Y);
title("Q1.d Discrete Time Response of Y");
legend("Y(1)", "Y(2)")
figure(8)
stairs(Tsim, X);
title("Q1.d Discrete Time Response of X");
legend("X(1)", "X(2)", "X(3)", "X(4)")
```

Joshua Benfell 300433229

1.e) Varied Sampling Period

Graphs



We can see that the only real change is the resolution of the plot. It matches up fine with the model.

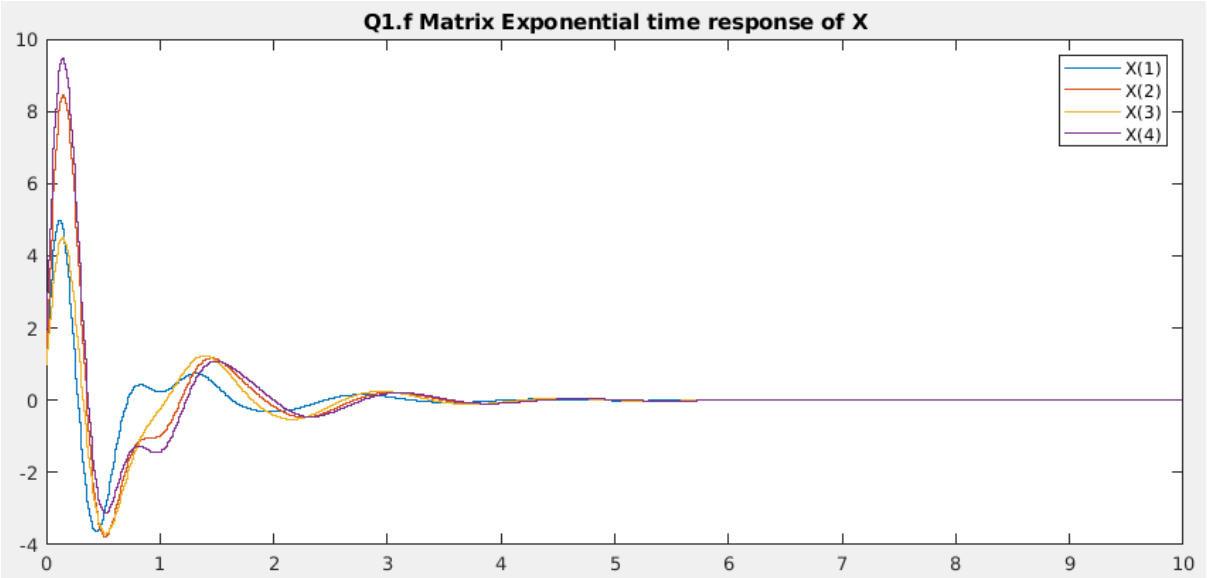
Matlab Code

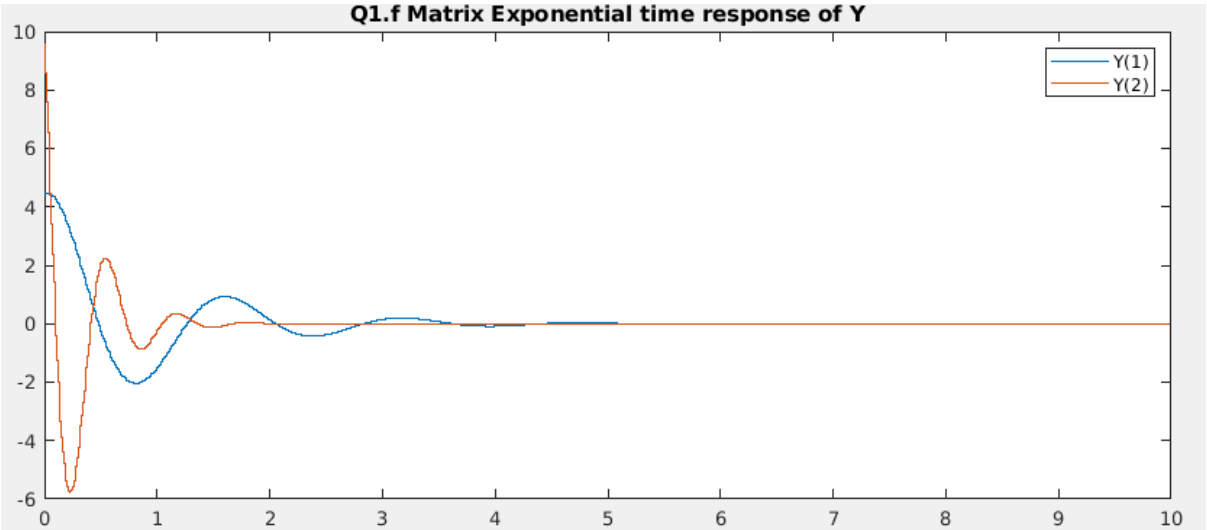
```
figure(9)
plot(Tsim, X(:,1));
hold on
for t = [0.01 0.050 0.1]
   T = 0:t:10;
   sysd = c2d(sys, t);
   X0 = [1, 1, 1, 2].';
   U = zeros(2, length(T));

   [Y, Tsim, X] = lsim(sysd, U, T, X0);

   stairs(Tsim, X(:,1))
end
legend("continuous", "0.01", "0.05", "0.1")
hold off
```

Joshua Benfell 300433229

1.f) Discrete Exponential Matrix

Graphs





Look
It looks the same
With a small enough time step you can't tell the difference.
Give marks I plotted graph 6 time and each time look pretty close to same.

Matlab Code

```
A = [12.5314 , -91.36 , 28.7129 , 59.9628 ;
    21.316 , -115.6631 , 37.5584 , 75.0519 ;
    13.967 , -53.5817 , 15.4161 , 33.4391 ;
```

```
    20.5222 , -123.3107 , 42.267 , 79.7156]; %4x4
B = [1.7649 , 1.7649;
    2.8318 , 2.8318;
    2.2353 , 2.2353;
    2.7294 , 2.7294]; %2x4
C = [-0.46166 , -4.4635 , 1.9151 , 3.7274 ;
     1.0853 , -12.82 , 4.5753 , 8.3759]; %4x2
 D = [0 , 0 ;
    0 , 0 ]; %2x2
% End Create System
T = 0:0.01:10;
T_idx = 2:1:length(T);
X0 = [1, 1, 1, 2].';  % Column vector of initial state
X = zeros(4, length(T));
X(:,1) = X0;
k_range = 10;
for t = T_idx
   phi = 0;
   for k = 0:1:k_range
      phi = phi + ((T(t)-T(t-1))*A)^k;
   end
   X(:,t) = expm((T(t) - T(t-1))*A)*X(:,(t-1));
end
figure(11)
stairs(T.',X.')
title("Q1.f Matrix Exponential time response of X")
legend("X(1)", "X(2)", "X(3)", "X(4)")
Y = C * X;
figure(12)
stairs(T.',Y.')
title("Q1.f Matrix Exponential time response of Y")
legend("Y(1)", "Y(2)")
```

Joshua Benfell 300433229

2.a) State Space Representation

```
A = [
    -2/(R*C), 1/(R*C), -R2/(R1 * R * C);
    1/(R*C), -2/(R*C), 1/(R*C);
    0, 1/(R*C), (-1/(R*C))-(1/(R1*C)) ];
```

$x_{next} = Ax$

There is no input. There is no output y either.
With
$R = 1k\Omega$
$C = 1mF$
$R_1 = R$

2.b) Oscillator Gain Ratio

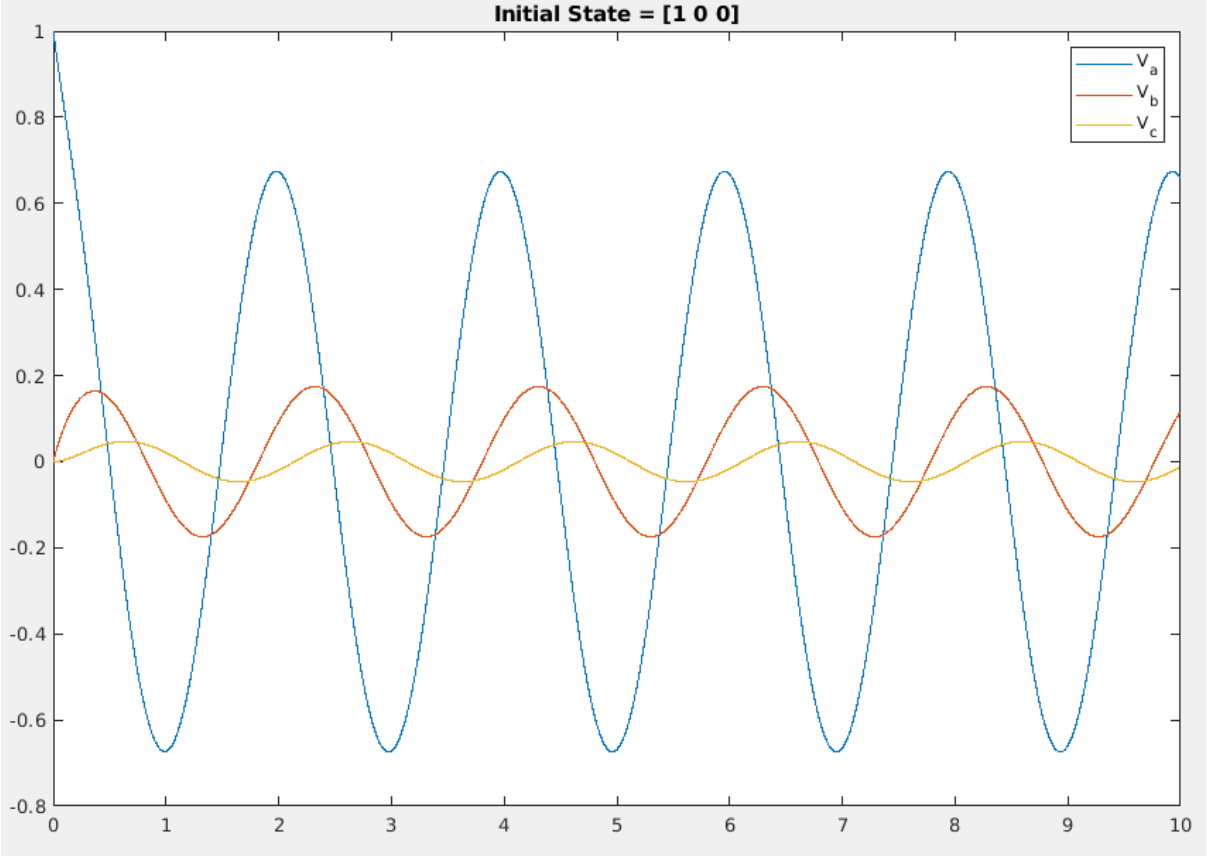
We know the system will be purely oscillatory when the two oscillatory modes are dominant and (when the real component is near 0) so we can determine this by trialing different values of R_2 and checking when two of the eigenvalues are (close to) purely imaginary as the real eigenvalue will be located at a position that makes it non-dominant. This occurs when $R_2 = 56R_1$. So a Gain of 56 gives us pure oscillation on the output.

2.c) Oscillation Frequency from state space model

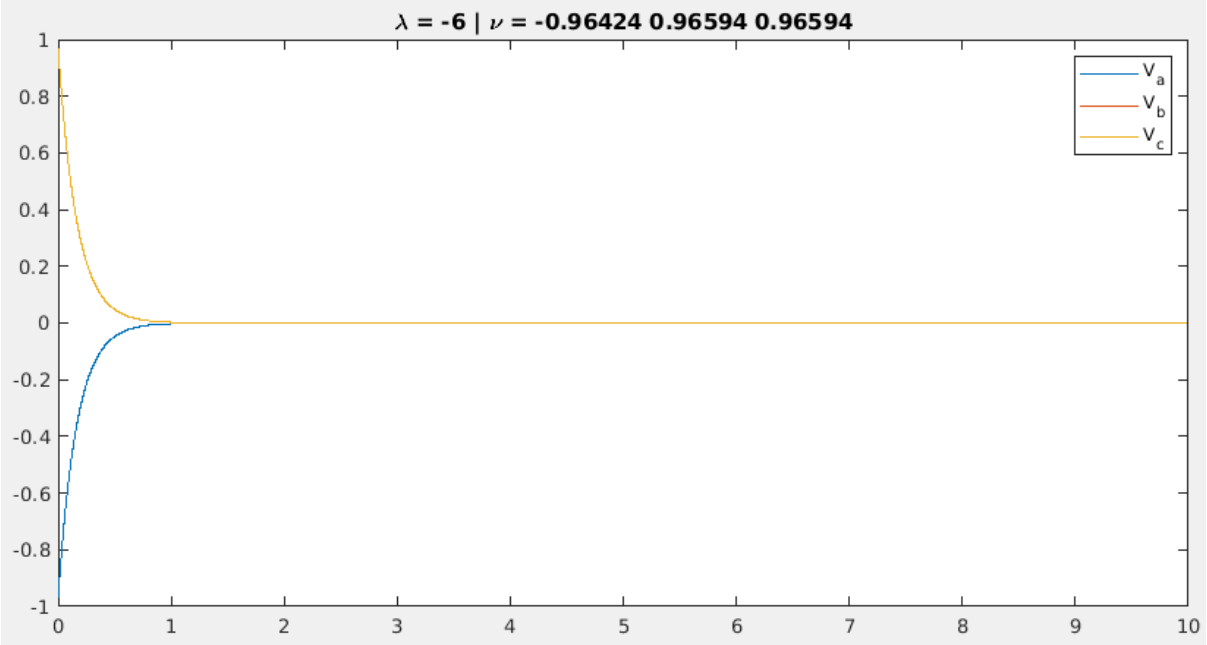
Eigenvalues
[-6.0000 + 0.0000i
0.0000 + 3.1623i
0.0000 - 3.1623i]
We have a complex pair a +- 3.1623 rad/s which corresponds to a frequency of $\frac{3.1623}{2\pi} = 0.5033Hz$
Which is significantly faster than the expected frequency of $\frac{1}{\sqrt{6}\times 1000\times 10^{-3}} = 0.04\,rad/s$
But this doesn't take into account any loading of the RC stages, and this nominal frequency equation also doesn't consider the effect of the inverting Amp and it's resistors.

Joshua Benfell 300433229

2.d) Plot the thing

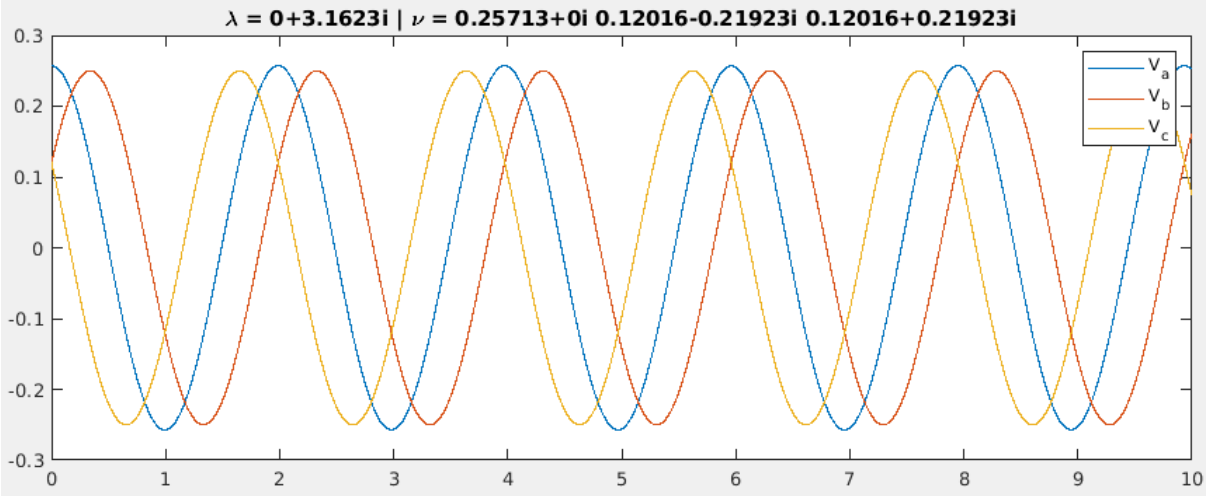


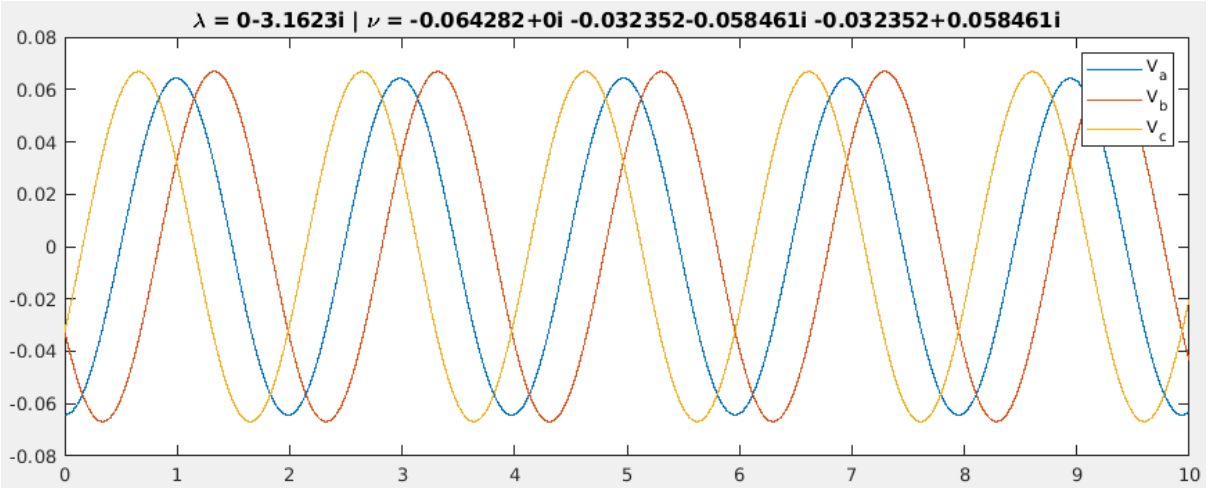
This plot was just plotting with any initial state as all oscillator circuits need either noise or an initial charge on the capacitors to kickstart them. Otherwise it's a very uninteresting circuit that has 0 output.



This plot is done by setting the initial conditions to the first eigenvector which allows us to replace the A matrix with the first eigenvalue, the pure real component, and we can see that there is a decay component to the system, however, it is quite far out from the origin, so

does not dominate the system. We can also see the phase shift of 180 degrees from going through the 3 stages





These two are plotted with the oscillatory modes, using the corresponding eigenvalues and eigenvectors. They are 180 degrees out of phase with each other.

Matlab Code

```
clear

R = 1000;

C = 1e-3;

R1 = 1*R;

R2 = R1*56;

A = [

    -2/(R*C), 1/(R*C), -R2/(R1 * R * C);

    1/(R*C), -2/(R*C), 1/(R*C);

    0, 1/(R*C), (-1/(R*C))-(1/(R1*C))

   ];
```

```
pts = 10000;
T = linspace(0,10,pts);
T_idx = 2:1:pts;
X0 = [1, 0, 0].';
X = zeros(size(T));
X = repmat(X, 3, 1);
X(:,1) = X0;
for t = T_idx
   X(:,t) = expm((T(t) - T(t-1))*A)*X(:,(t-1));
end
plot(T,X)
legend("V_a", "V_b", "V_c");
title("Initial State = [1 0 0]")
[eigenvectors, eigenvalues] = eig(A);
for idx = [1 2 3]
   X0 = eigenvectors(idx,:).';
   X = zeros(3, length(T));
   X(:,1) = X0;
   for t = T_idx
       X(:,t) = expm((T(t) -
T(t-1))*eigenvalues(idx,idx))*X(:,(t-1));
   end
   figure(idx + 1);
   plot(T,X)
   legend("V_a", "V_b", "V_c");
   title("\lambda = " + strjoin(string(eigenvalues(idx,idx))) + " |
\nu = " + strjoin(string(X0)))
end
```

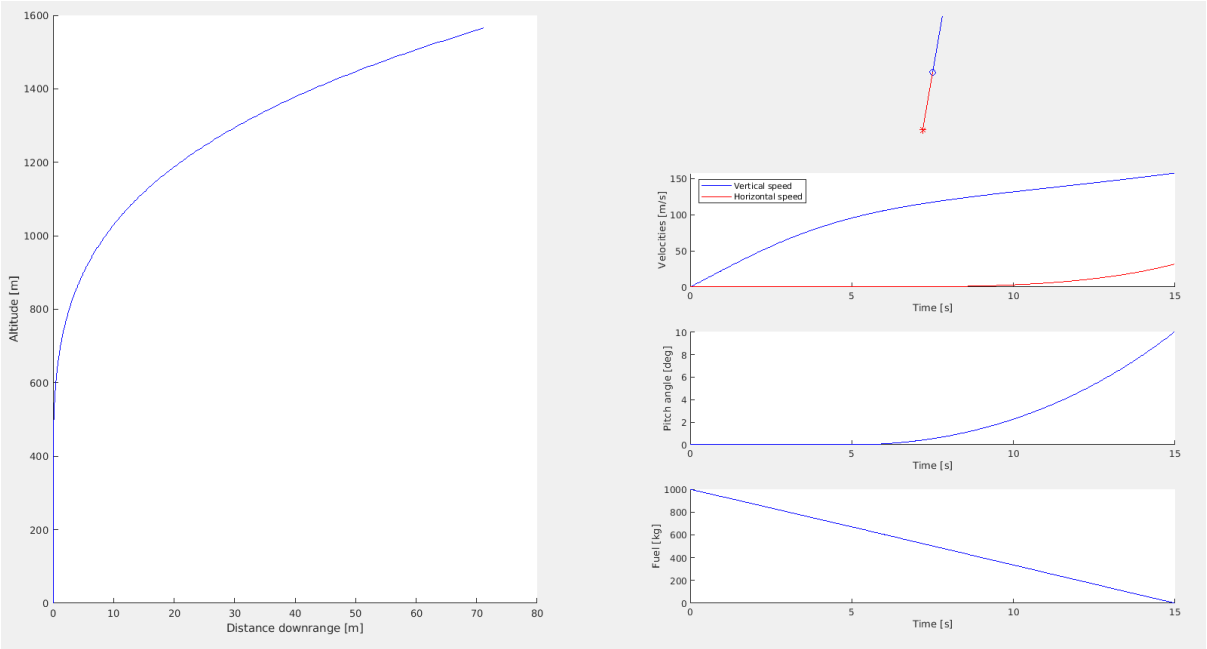
Joshua Benfell 300433229

B. Summative

3.a) Initial Values for U

$U = [66000, 0]^T \; for\, t < 5, [66000, 1025] \; for\, t >= 5$

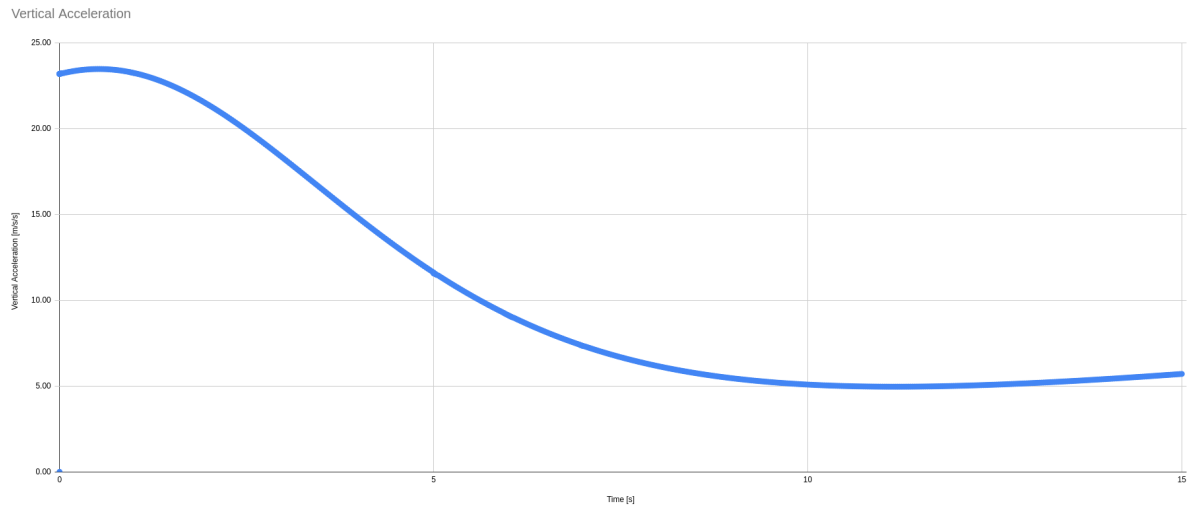
These values pitch it over 10 degrees between t=5 and t=15 and use up (pretty much) all of the fuel.



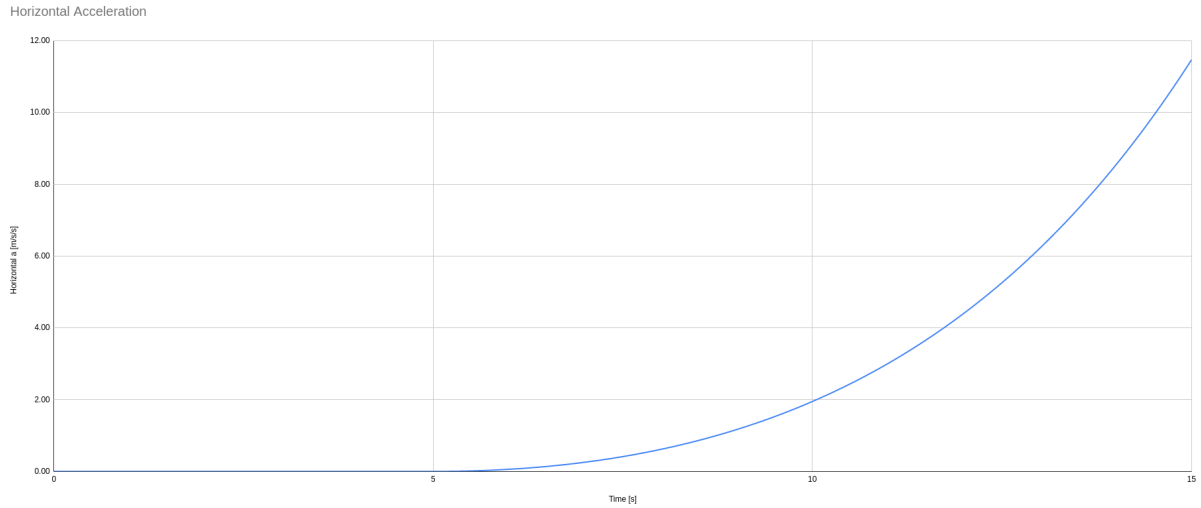
3.b) Acceleration

Max Horizontal a = 11.47 @ t=15s
Max vertical a = 23.48 @ t=0.4588s to t=0.5838s
Max total a = 23.48027 @ t=0.5338s which is in the range of when the maximum vertical acceleration occurs.

Vertical and Horizontal obtained by $a(t) = \frac{v(t+dt)-v(t)}{dt}$ where dt = 0.025 (from the



Interestingly it starts to increase after 10s, indicating that the portion adding to the vertical starts to be comparably bigger as it tilts more.
But the fuel leaving is not the main factor to the acceleration when a constant force is applied.



As the velocity vector is pushed more horizontally, it starts to pick up horizontal speed faster.

3.c) Non-linear Variables

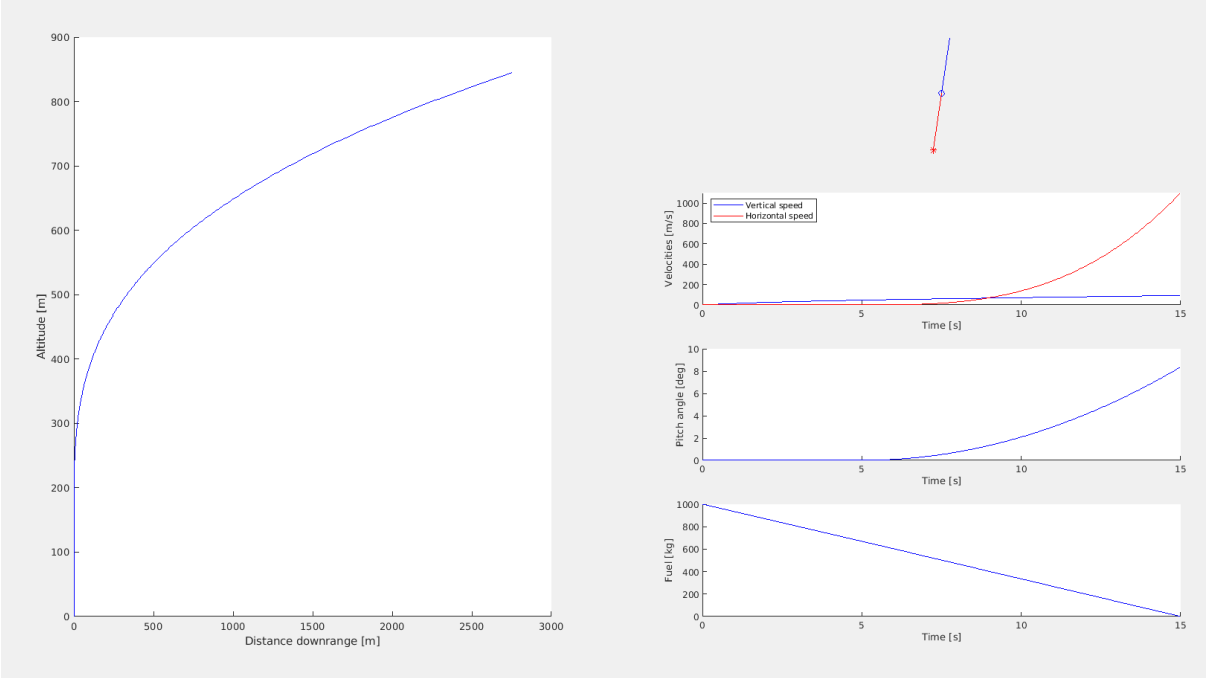
The variables that make the system non-linear
X2, X4, and X6 as the vertical, horizontal, and angular velocity are inversely proportional to the remaining fuel (the weight of the rocket)
Consequently, because they are the differential of the positions, the vertical, horizontal, and angular position will be non-linear as they will have an inverse square relationship to the remaining fuel.

The vertical velocity is also affected by drag, which is a squared relationship to the velocity, making the vertical position have a cubed relationship to the velocity.

We can linearise the sine and cosine functions because the angle is small (ish) (maximum 10 degrees) so that can be linearised such that sin(x) = x

Joshua Benfell 300433229

3.d) Linearisation at t=5



Using the values at t=5 to linearise it shows us that the horizontal velocity will be greatly exaggerated, while the vertical is underestimated a bit, which results in the angle being a bit short as well cause the rocket is not flying up fast enough, which doesn't contribute as much to the horizontal as it is tilted. However, the fuel is pretty good.

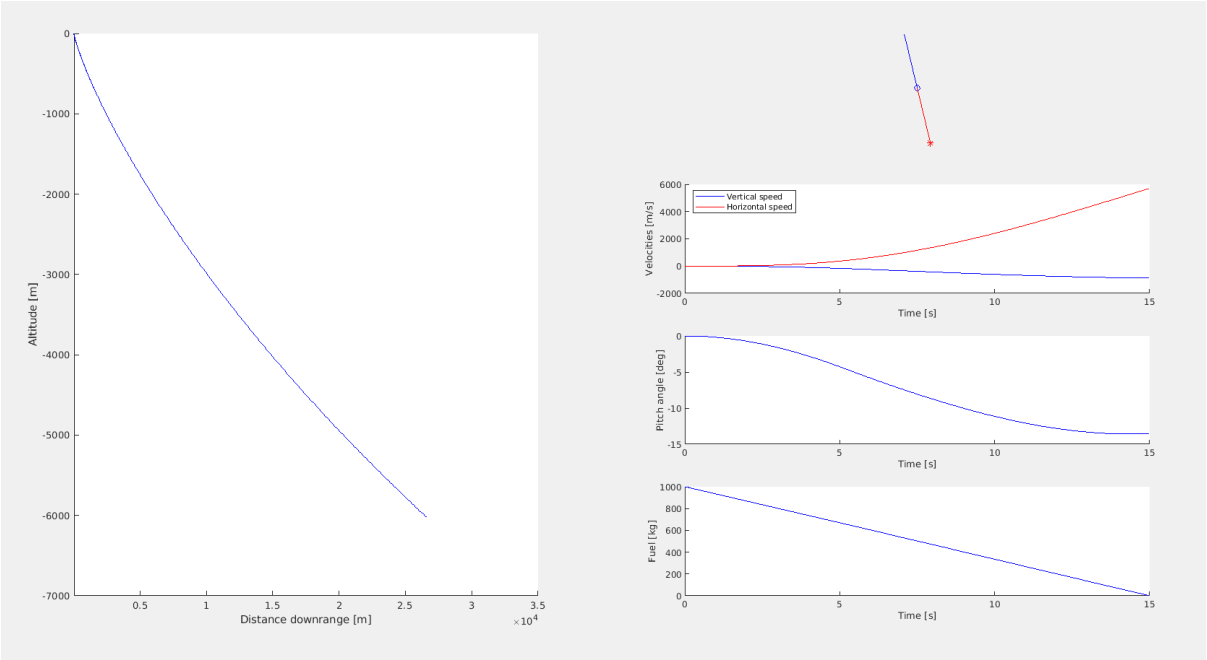
Matlab Code

```
%-------------------------------------------------------------------------
% Linearisation from t = 5 (When it starts pitching over) 3d
%-------------------------------------------------------------------------
syms u1 u2
df = jacobian(f_sym, [x1 x2 x3 x4 x5 x6 x7]);
dg = jacobian(f_sym, [u1 u2]);
A = double(subs(df, [x1 x2 x3 x4 x5 x6 x7 u1 u2], [x_store(230,1:7) 66000 0]));
B = double(subs(dg, [x5 x7], [x_store(230, 5) x_store(230, 7)]));


elapsed_time = 0;
figure(2)
t = 0:0.025:15;
len_t = length(t);
% set up input Matrix
u = zeros(2,len_t);
u(1,:) = 66000 * ones(1, len_t);
u(2, t>=5) = 1025;

sys = ss(A,B,[],[]);

[Y, Tsim, X] = lsim(sys, u, t, x_0);
```

Joshua Benfell 300433229

```
X(:,8:9) = u.';
plot_telemetry(Tsim, X);


% End Linearisation 1 ------------------------------------------------
```

3.e) Linearisation at t=15



From this we can see that t=15 is way too far ahead to simulate the whole flight. From the graph in 3a we can see that the vertical velocity does start slowing down, which would be a good explanation for why we start going negative immediately using the starting conditions. The angle may be off, but the visualisation could just be 180 degrees off and it is pointing 10 degrees down. Essentially the model really breaks down and does not function near correctly.

Matlab Code

```
%--------------------------------------------------------------------------
% Linearisation from t = 15 (When it starts pitching over) 3e
%--------------------------------------------------------------------------
syms u1 u2
A = double(subs(df, [x1 x2 x3 x4 x5 x6 x7 u1 u2], [x_store(end,1:7) 66000 1025]));
B = double(subs(dg, [x5 x7], [x_store(end, 5) x_store(end, 7)]));


elapsed_time = 0;
figure(3)
t = 0:0.025:15;
len_t = length(t);
% set up input Matrix
```

```
u = zeros(2,len_t);
u(1,:) = 66000 * ones(1, len_t);
u(2, t>=5) = 1025;

sys = ss(A,B,[],[]);

[Y, Tsim, X] = lsim(sys, u, t, x_0);
X(:,8:9) = u.';
plot_telemetry(Tsim, X);


% End Linearisation 2 ----------------------------------------------------
```

3.f)

Using the discussion above, I would use the linearisation at t=5 as this is closer to the starting conditions, allowing it to better model the initial flight of the rocket, which starts us in the correct direction. Which is more important than being off in how far you have travelled horizontally, as we'd rather have a rocket that flies in the first place.