CSCI 545: Homework 4

Deepika Anand

May 1, 2017

Problem 1. (a)

Equilibrium point. Given that

$$\dot{z} = \alpha_z(\beta_z(g - y) - z) + f \tag{1}$$

and

$$\dot{y} = z \tag{2}$$

Hence, at equilibrium $f = 0$
Velocity that is $\dot{y}$ is 0 and hence $\dot{z}$ is 0
Therefore

$$\alpha_z(\beta_z(g - y)) = 0 \tag{3}$$

Implies $y = g$ at equilibrium

Problem 1. (b)

Derivation for stability analysis. We need to linearise the system and then find eigen values.

$$\dot{z} = \alpha_z(\beta_z(g - y) - z) + f \tag{4}$$

It can be re-written as

$$\dot{z} = \alpha_z\beta_z g - \alpha_z\beta_z y - \alpha_z z + f \tag{5}$$

$$\dot{z} = \alpha_z\beta_z(g + \frac{f}{\alpha_z\beta_z} - y) - \alpha_z z \tag{6}$$

Assume, the following fraction to be a random variable say x

$$\frac{f}{\alpha_z\beta_z} = x \tag{7}$$

hence

$$\dot{z} = \alpha_z\beta_z(x - y) - \alpha_z z \tag{8}$$

and

$$\dot{y} = z \tag{9}$$

now write it in the form of matrix

$$\begin{bmatrix} \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial y} \\ \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial y} \end{bmatrix}$$

The aim here is to present in the equation in the form of

$$\dot{x} = Ax + Bu \tag{10}$$

In this case A =

$$\begin{bmatrix} -\alpha_z & -\alpha_z\beta_z \\ 1 & 0 \end{bmatrix}$$

To find eign values

$$det(A - sI) = 0 \tag{11}$$

det(

$$\begin{bmatrix} -(s + \alpha_z) & -\alpha_z\beta_z \\ 1 & -s \end{bmatrix}$$

) = 0

$$s(s + \alpha_z) + \alpha_z\beta_z = 0 \tag{12}$$

$$s^2 + s\alpha_z - \alpha_z\beta_z = 0 \tag{13}$$

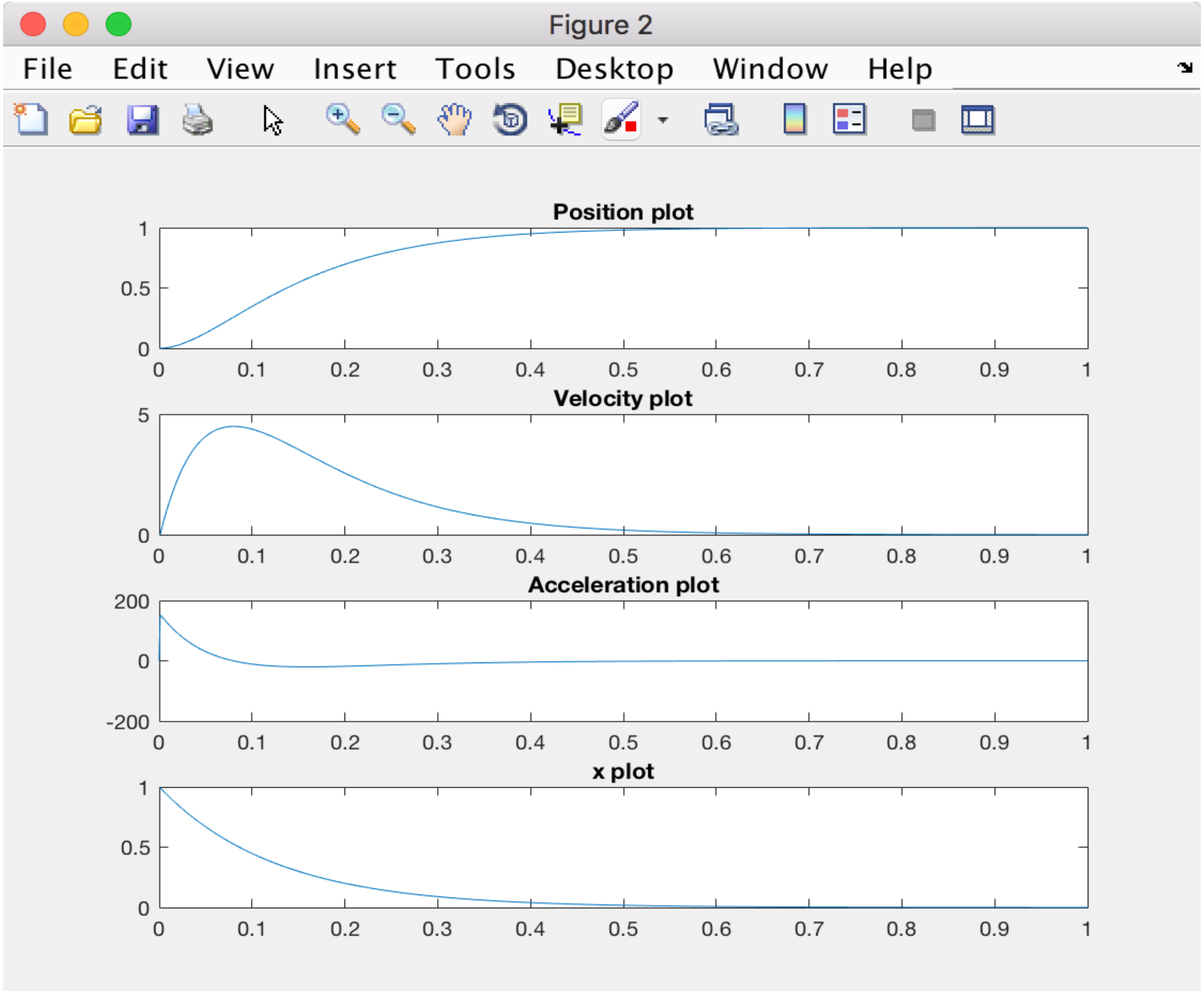
The roots are,

$$s_{1,2} = \frac{-\alpha_z \pm \sqrt{\alpha_z^2 - 4\alpha_z\beta_z}}{2} \tag{14}$$

For stability $s_{1,2}$ should be < 0
To ensure stability $\alpha_z > \sqrt{\alpha_z^2 - 4\alpha_z\beta_z}$

Problem 1. (c)

w = [0 0 0 0 0 0 0 0 0 0 0]
Position, Velocity, Acceleration and x graph



ψ Graph

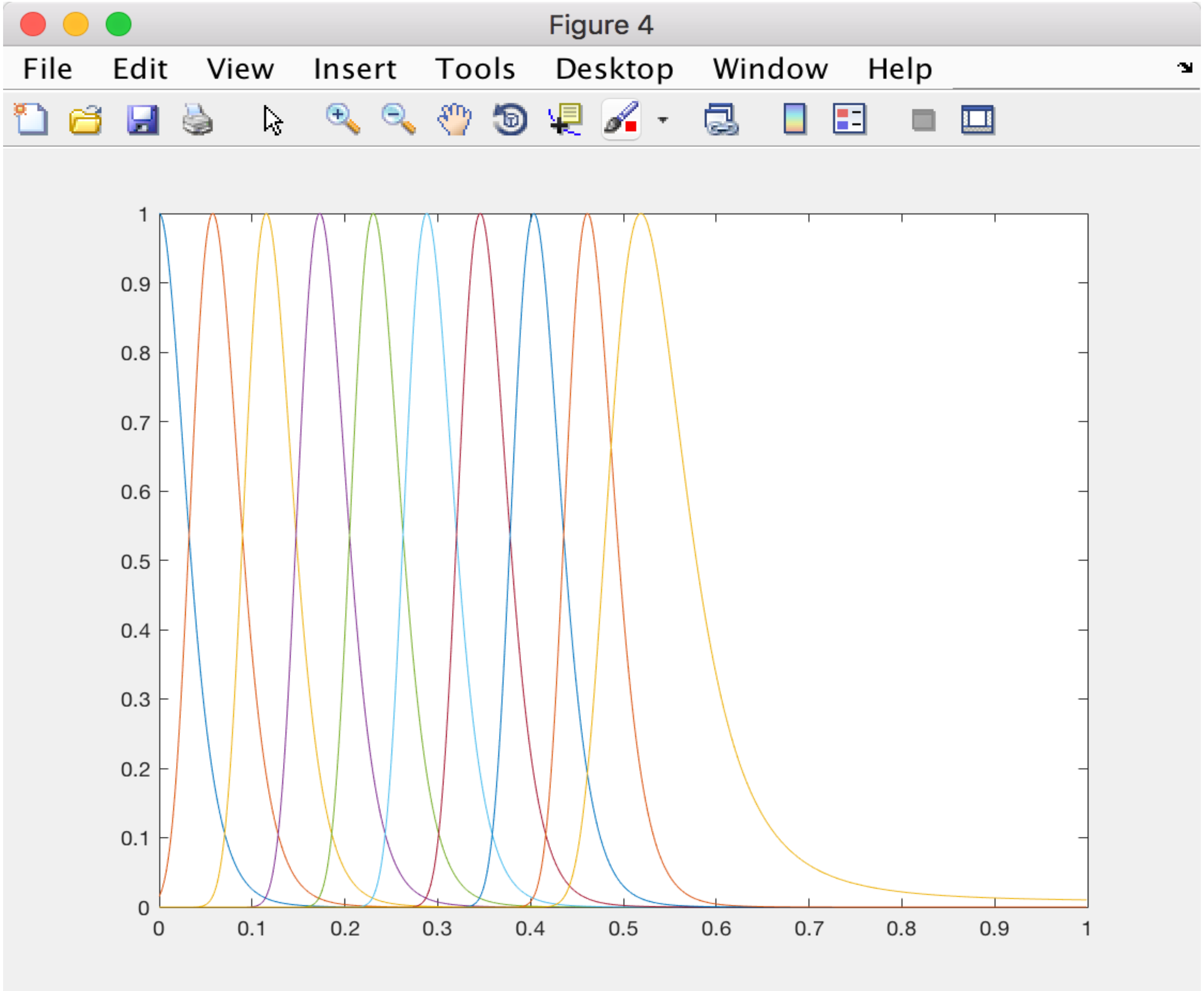


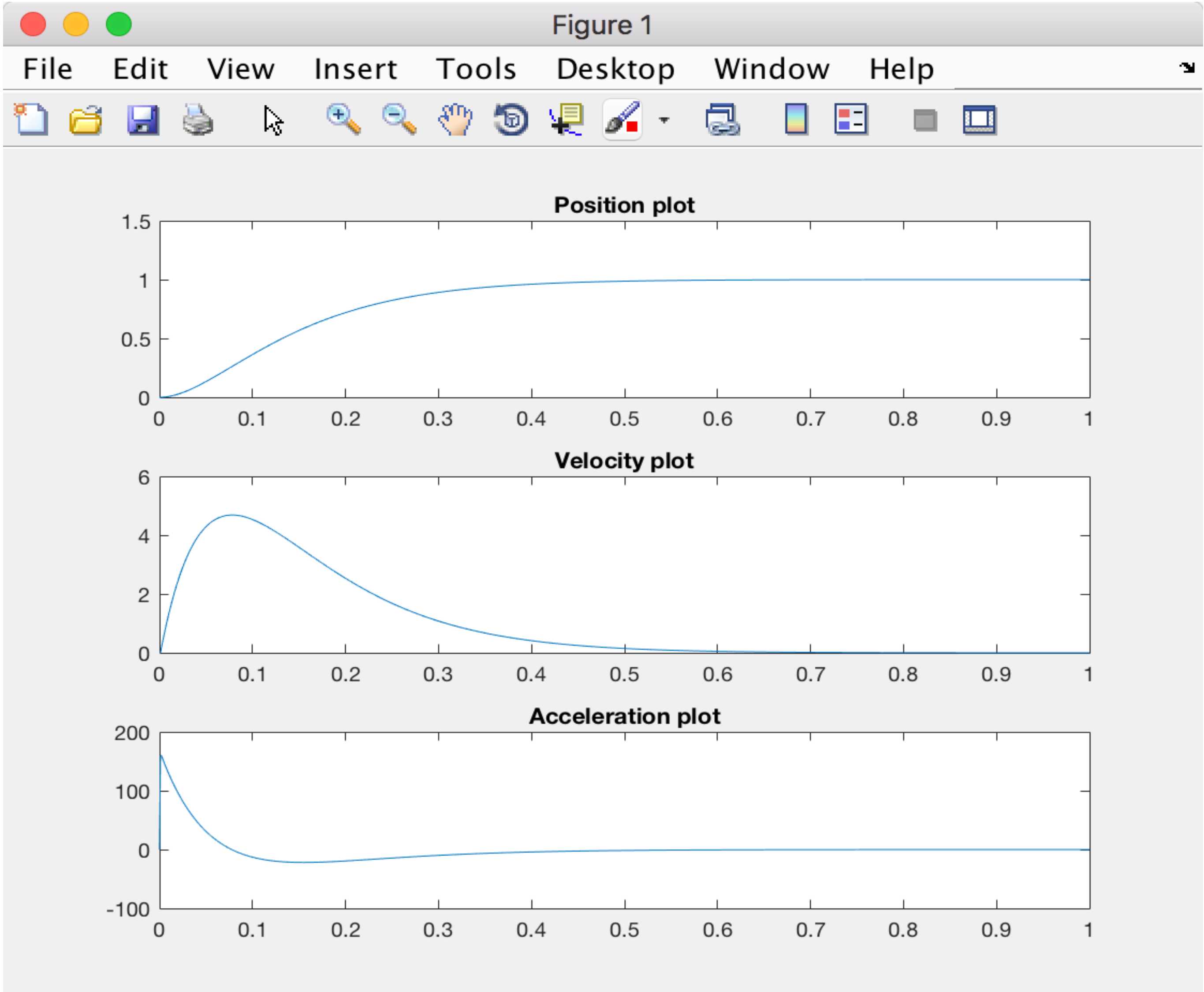
Figure 4
File
Edit
View
Insert
Tools
Desktop
Window
Help
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Code

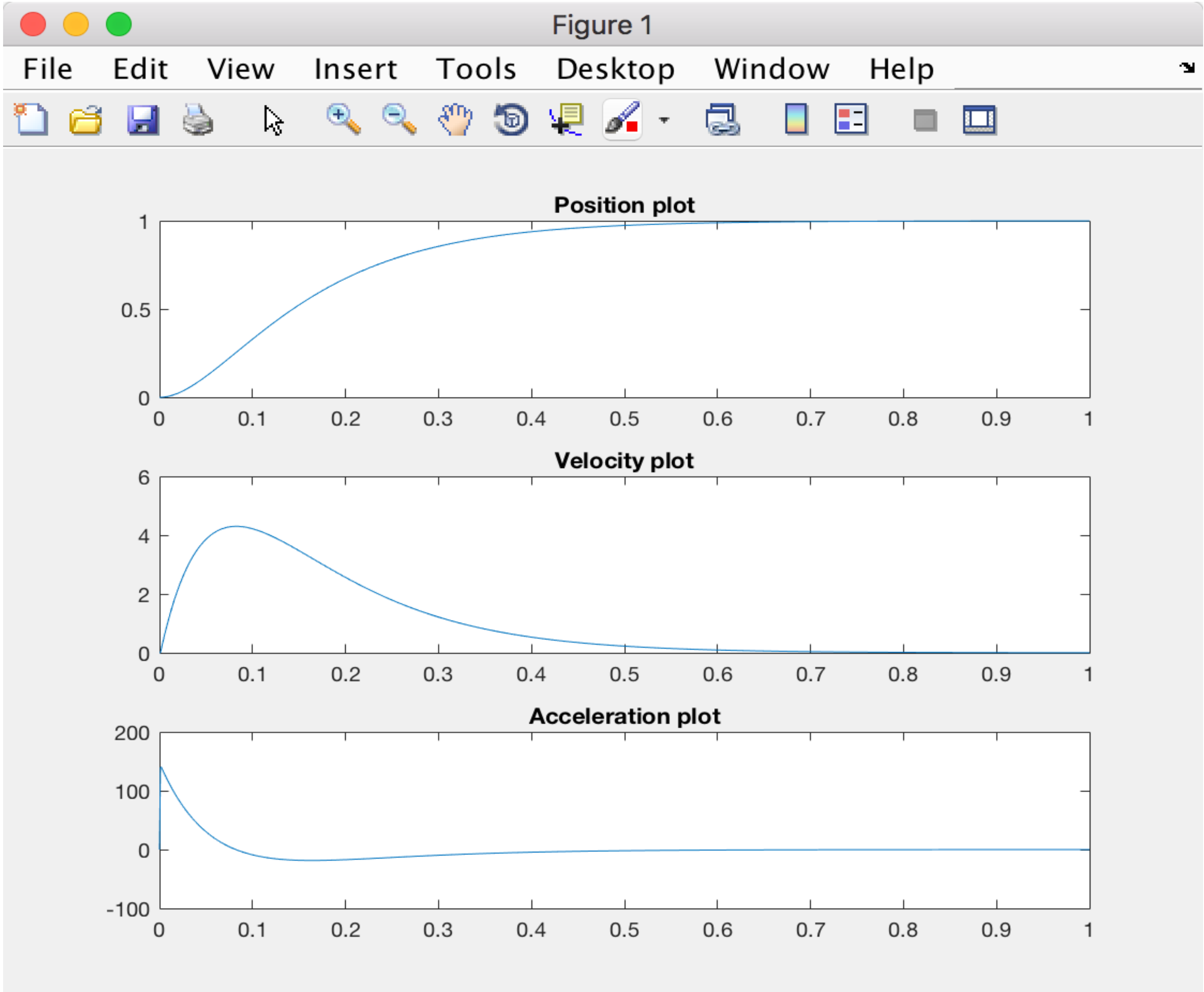
```
si(1)=0;
phi(1)=0;

finalmatrix = [];
for i = 1:length(t)-1
    xdot(i) = - alpha_x * x(i);
    x(i+1) = x(i) + xdot(i) * dt;

    for j = 1:10
        si(j)= exp((-1/(2 * sigmaSquare(j))) * ( (x(i) - c(j)) * (x(i) - c(j)) ));
    end
    for j = 1:10
        phi(j) = (si(j) * x(i))/sum(si);
    end
    finalmatrix = [finalmatrix; si];
    force = phi * transpose(w);

    y_d(i+1)=y_d(i) + dt * y_dd(i);
    y_dd(i+1)=y_dd(i) + dt * (25 * ( 6 * ( 1 - y(i) ) - y_d(i))) + force;
    y(i+1) = y(i) + dt*y_d(i+1);
```

Problem 1. (d)

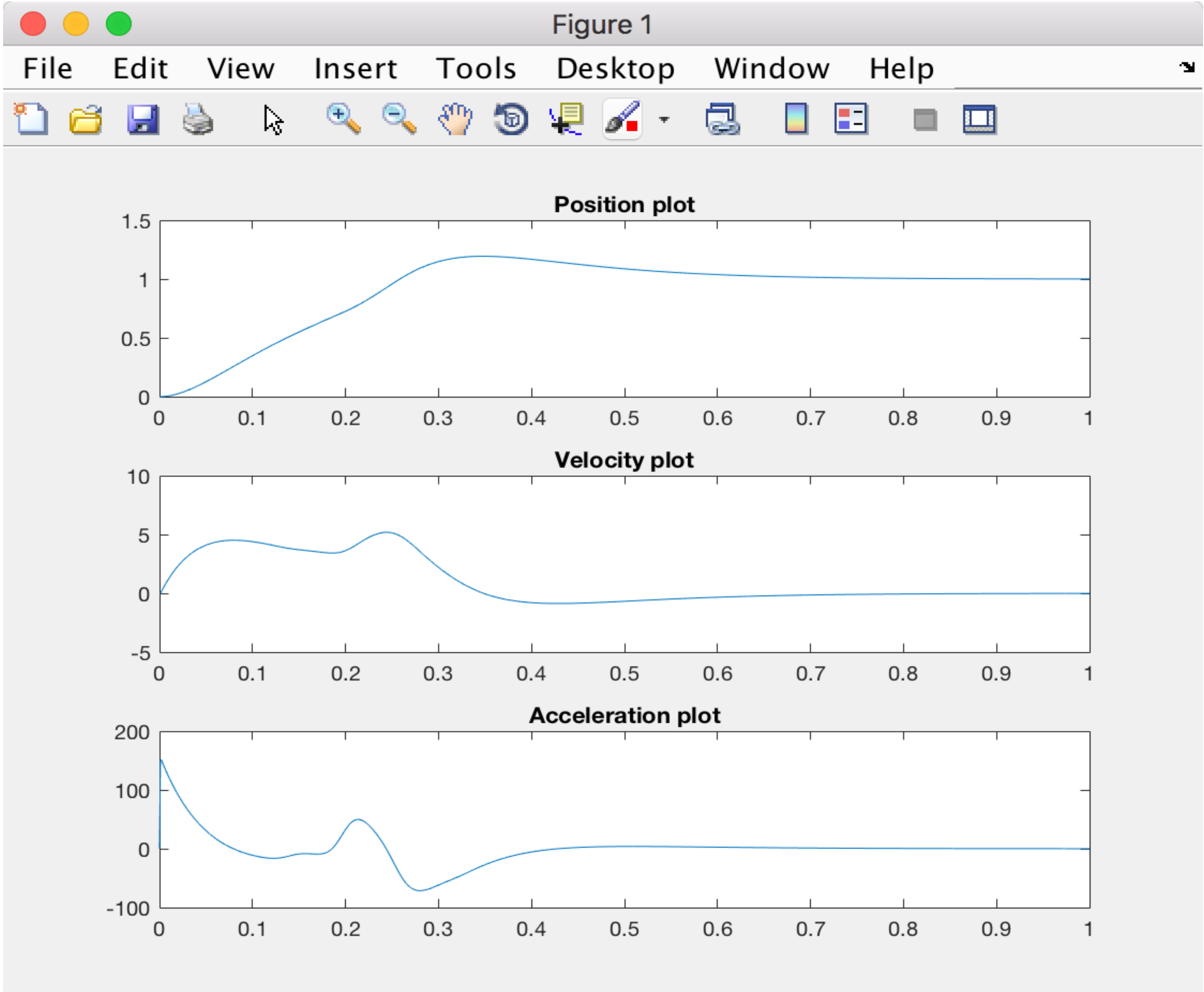
- When w = [10 10 10 10 10 10 10 10 10 10 10]
  Position, Velocity, Acceleration and x graph



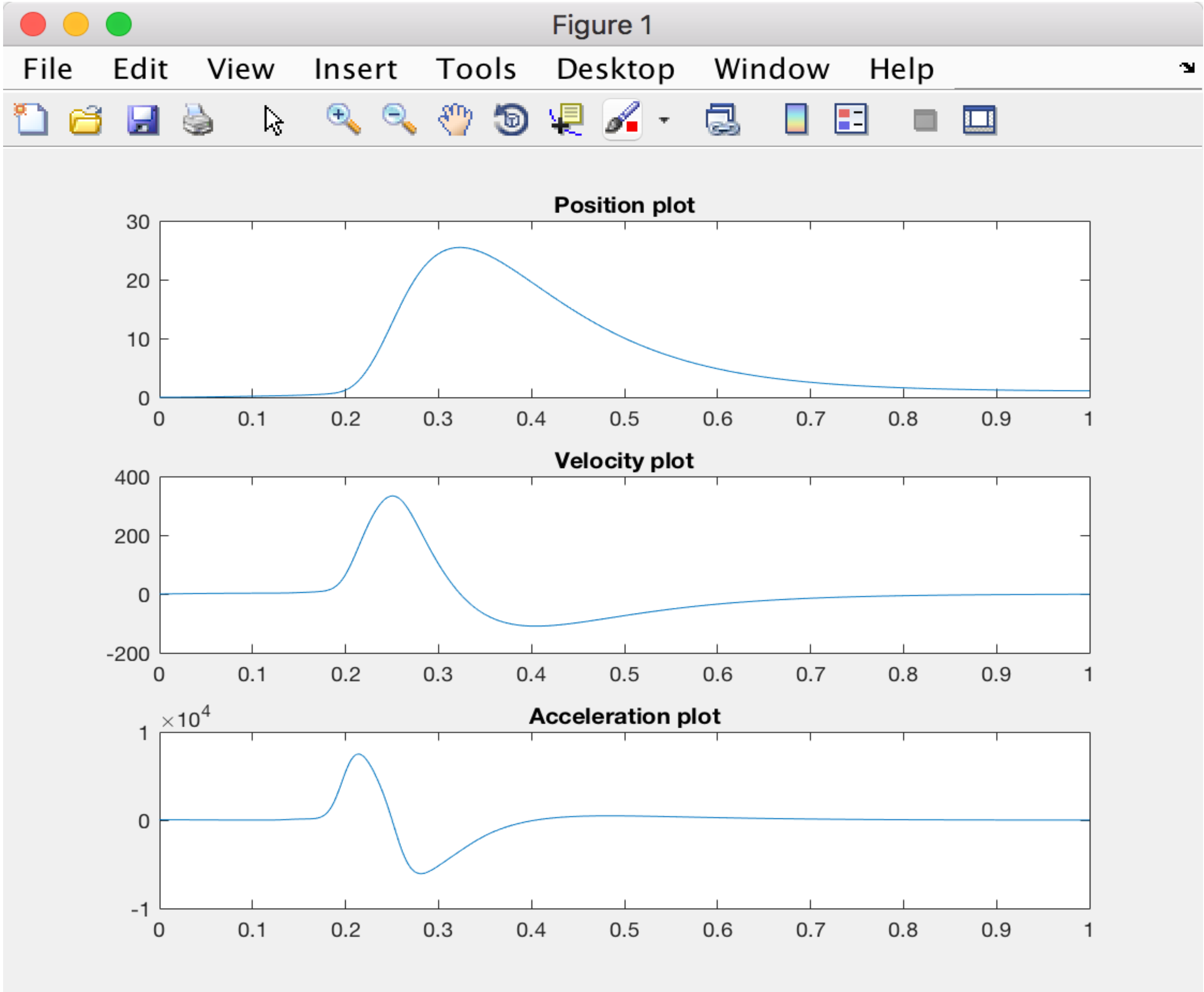
- When w = [-10 -10 -10 -10 -10 -10 -10 -10 -10 -10 -10]
  Position, Velocity, Acceleration and x graph



- When w = [1 1 1 100 1000 100 1 1 1 1] Position, Velocity, Acceleration and x graph



- When w = [-100 -100 -100 1000 100000 1000 1 1 1 1] Position, Velocity, Acceleration and x graph



Problem 1. (e)

For imitation learning.I have used two method. One as mentioned in control theory lecture in which w is given as

$$(\phi^T \phi)^{-1} \phi^T * f \tag{15}$$

and other in which w is given as

$$w_i = \frac{\mathbf{s}^T \boldsymbol{\psi}_i \mathbf{f}_d}{\mathbf{s}^T \boldsymbol{\psi}_i \mathbf{s}},$$

where

$$\mathbf{s} = \begin{pmatrix} x_{t_0}(g - y_0) \\ \vdots \\ x_{t_N}(g - y_0) \end{pmatrix}, \quad \boldsymbol{\psi}_i = \begin{pmatrix} \psi_i(t_0) & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \psi_i(t_n) \end{pmatrix}$$

and

$$f_d = \ddot{y}_d - \alpha(\beta(g - y) - \dot{y}) \tag{16}$$

- CASE 1: When w is given as $(\phi^T \phi)^{-1} \phi^T * f$

where f is given as

$$f = \ddot{y}_d - \alpha(\beta(g - y) - \dot{y}) \tag{17}$$

Code is given in **PartE-LinearRegressionFormula.m**

Editor – /Users/deepika/Documents/MATLAB/projects/hello_1/Hw4/PartE_LinearRegressionFormula.m

inverse_kinematics.m | forward_kinematics.m | ArmAnimation.m | hw3.m | PartE.m | PartC.m | PartE_L

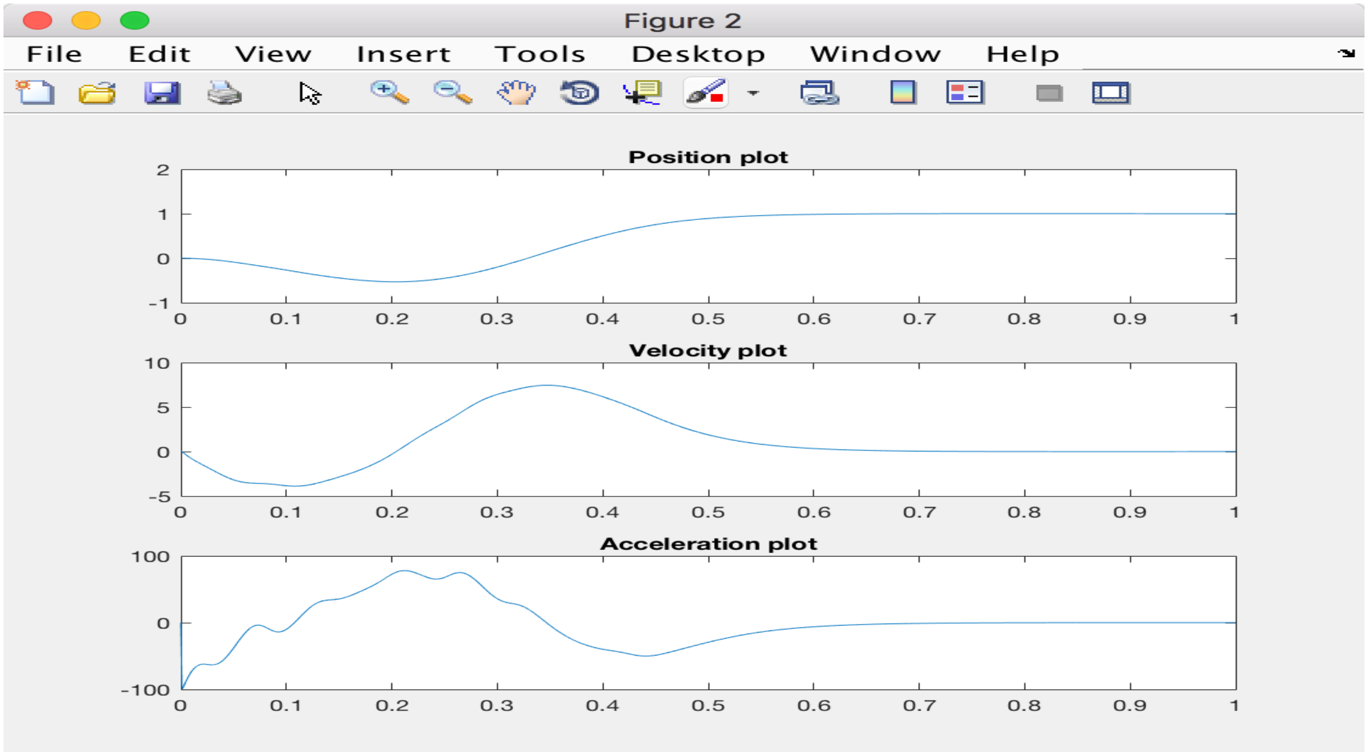
```
for i = 1:length(t)-1
    x(i+1) = x(i) - alpha_x*x(i)*dt;
end
s=x; % 1000 X 1
finalmatrix=[];
for i = 1:length(t)
    xdot(i) = - alpha_x * x(i);
    x(i+1) = x(i) + xdot(i) * dt;
    phi=[];
    si=[];
    for j = 1:10
        si(j)= exp((-1/(2 * sigmaSquare(j))) * ( (x(i) - c(j)) * (x(i) - c(j)) ));
    end
    for j = 1:10
        phi(j) = (si(j) * x(i))/sum(si);
    end
    finalmatrix = [finalmatrix; phi];

end
w = ((finalmatrix'*finalmatrix)^-1)*finalmatrix'*f_d;
disp(w')
```

Command Window

```
>> PartE_LinearRegressionFormula
   1.0e+03 *

   -0.2462   -0.4626   -0.7596   -0.9319   -0.6426    0.1984    1.0303    0.9916    0.2390    0.0332
```

Ouptut in this case

-246.1932594 -462.5872036 -759.5941195 -931.9358686 -642.6412049 198.350185 1030.307108 991.5533221 238.9686813 33.24762351



Second case where derivation taken by paper written by the professor.

$$w_i = \frac{\mathbf{s}^T \boldsymbol{\psi}_i \mathbf{f}_d}{\mathbf{s}^T \boldsymbol{\psi}_i \mathbf{s}},$$

where

$$\mathbf{s} = \begin{pmatrix} x_{t_0}(g - y_0) \\ \vdots \\ x_{t_N}(g - y_0) \end{pmatrix}, \quad \boldsymbol{\psi}_i = \begin{pmatrix} \psi_i(t_0) & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \psi_i(t_n) \end{pmatrix}$$

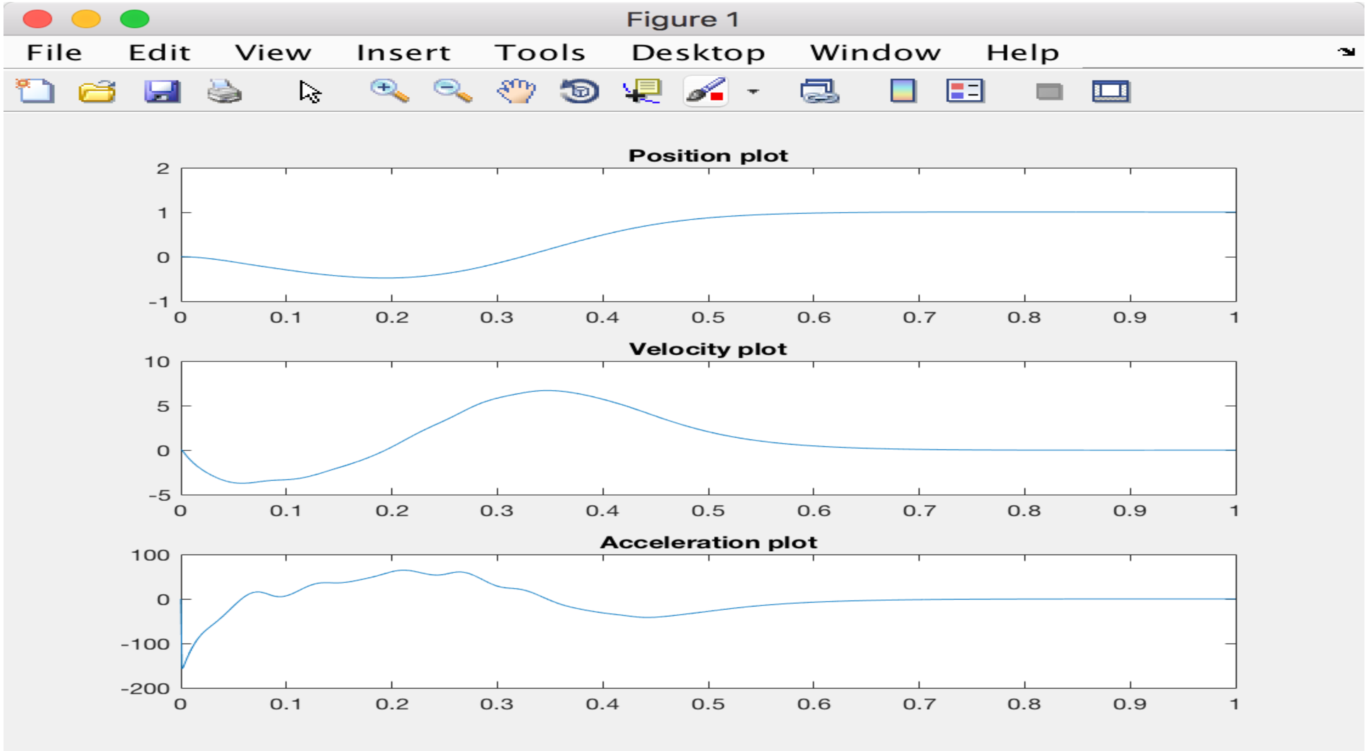
Code

```
filename = 'imitation.data';
delimiter=' ';
data=importdata(filename,delimiter);
yn=data(:,1);
y_dn=data(:,2);
y_ddn=data(:,3);

g = ones(1001);
g = g(:,1);
f_d = y_ddn - 25.*(6.*(g - yn) - y_dn);

t=0:dt:1;
for i = 1:length(t)-1
    x(i+1) = x(i) - alpha_x*x(i)*dt;
end
s=x; % 1000 X 1

for i=1:10
    psi=ones(1001);
    for j=1:1001
        psi(j,j) = exp((-1/(2 * sigmaSquare(i))) * ( (x(j) - c(i)) * (x(j) - c(i)) ));
    end
    w(i) = det(transpose(s) * psi * f_d)/det(transpose(s) * psi * s);
end
disp(w)
```

Output w

```
-305.1741 -435.7856 -686.6491 -840.2790 -627.4147   37.7373  761.4741  896.6861  406.31
```

Position, Velocity, Acceleration and x graph



Comments: When we use imitation learning, complexity of trajectory increases. also the first case works better than second case and the trajectory in first case is same as the one given in data file.