1. a, b

1.

a) $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = 0$

$$\Rightarrow \begin{cases} \alpha_z(\beta_z(g-y)-z)+f=0 \\ z=0 \\ -\alpha_x \cdot x = 0 \end{cases} \Rightarrow \begin{cases} y = +\frac{f}{\alpha_z\beta_z} + g \\ z=0 \\ x=0 \end{cases}$$

Since $x=0$, we can get $f=0$, $\Rightarrow y = g$, $x=0$, $z=0$

---

b) $\dot{x} = -a_x x$

$\Rightarrow$ Eigenvalue: $-\alpha_x - \lambda = 0$

$\lambda = -\alpha_x$

Since $\alpha_x > 0$, then $Real(\lambda) = -\alpha_x < 0$, which indicates that the canonical system is stable.

Now, let's consider the Tranformation System:

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

Since, we proved that the canonical systm is stable ~~and~~ at equilibrium point $x=0$. Since $f(0)=0$, then at equilibrium point, $f(x)$ could be ignored at this ~~ana~~ analysis.

$\Rightarrow \quad \dot{z} = \alpha_z(\beta_z(g-y)-z)$ ~~$+f(x$~~

$$\Rightarrow \begin{bmatrix} \dot{z} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -\alpha_z & -\alpha_z\beta_z \\ 1 & 0 \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} + \begin{bmatrix} \alpha_z \cdot \beta_z g \\ 0 \end{bmatrix}$$

$$\Rightarrow \begin{vmatrix} -\alpha_z - \lambda & -\alpha_z\beta_z \\ 1 & -\lambda \end{vmatrix} = 0 \quad \Rightarrow \lambda^2 + \alpha_z\lambda + \alpha_z\beta_z = 0$$

$$\Rightarrow \lambda = \frac{-\alpha_z \pm \sqrt{\alpha_z^2 - 4\alpha_z\beta_z}}{2}$$

$\Rightarrow$ if $\alpha_z^2 - 4\alpha_z\beta_z < 0$, then $\text{Real}(\lambda) = -\frac{\alpha_z}{2} < 0$, the system is stable.

if $\alpha_z^2 - 4\alpha_z\beta_z \geq 0$, $\text{Real}(\lambda) = \frac{-\alpha_z \pm \sqrt{\alpha^2 - 4\alpha_z\beta_z}}{2}$

$$\leq \frac{-\alpha_z + \sqrt{\alpha^2 - 4\alpha_z\beta_z}}{2}$$

$$< \frac{-\alpha_z + \sqrt{\alpha_z^2}}{2}$$

$$= \frac{-\alpha_z + \alpha_z}{2}$$

$$= 0$$

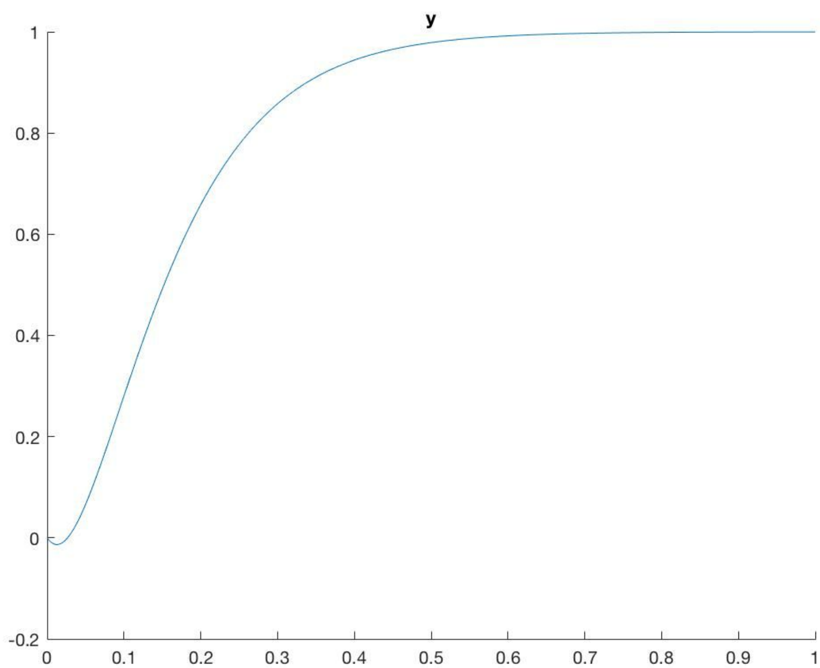
$\Rightarrow$ The system is stable in this case.

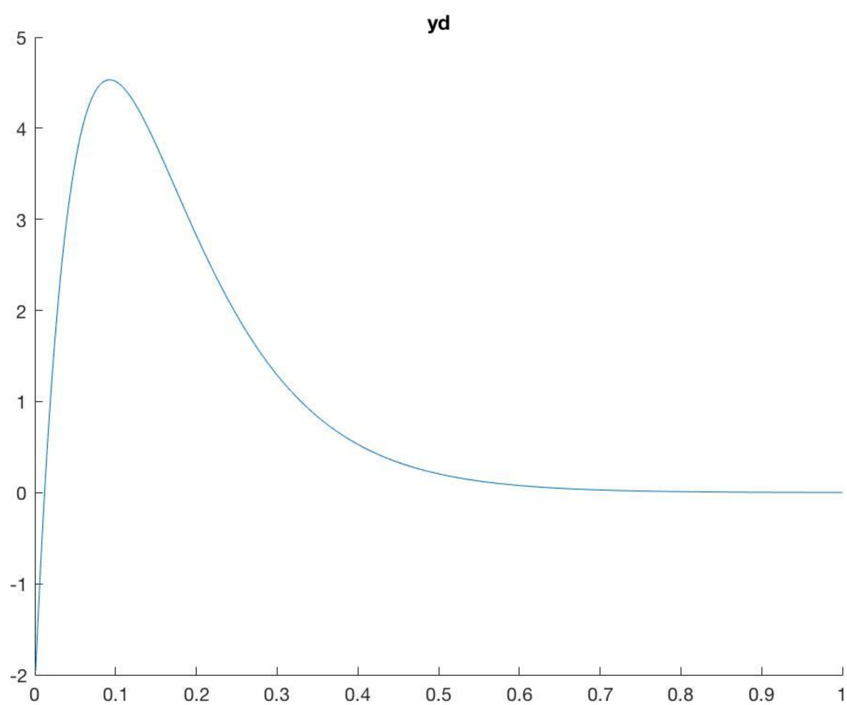
$\Rightarrow$ The system is stable in all cases.

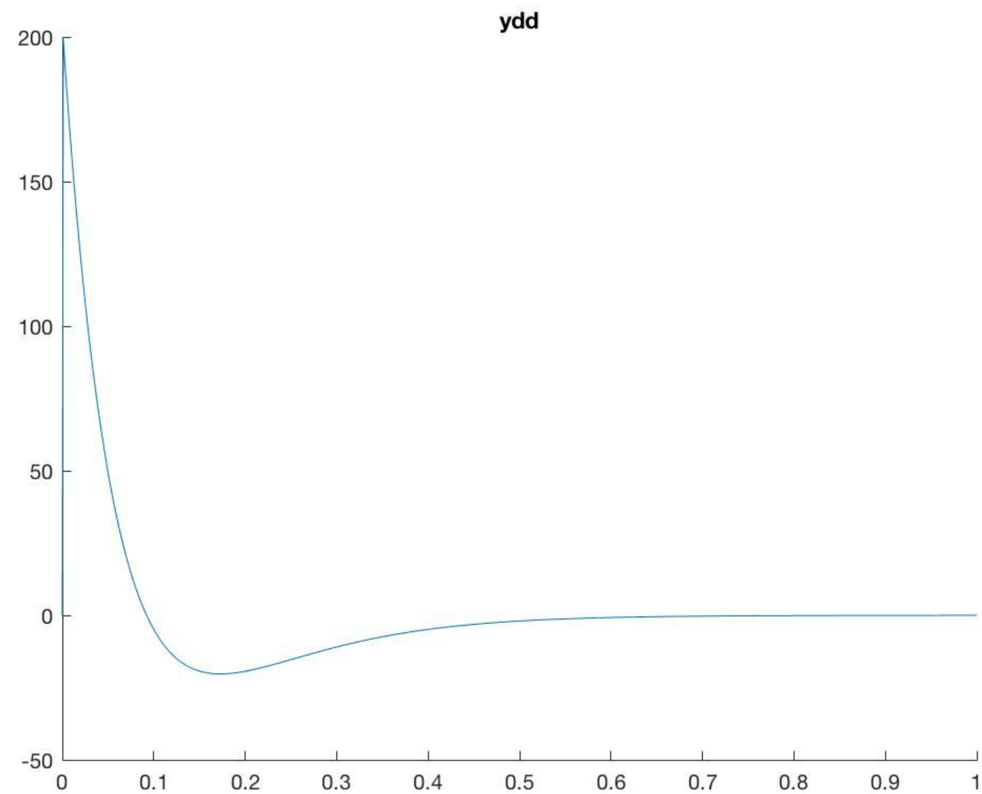
1.C

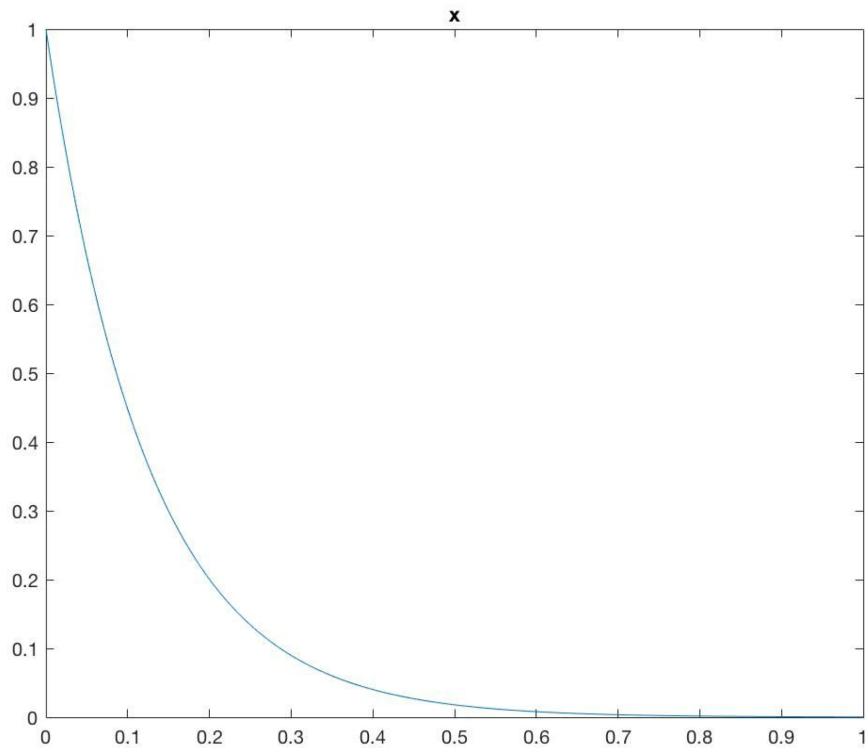
Here I provide a snippet of my codes(also available at src/hw4_c.m)

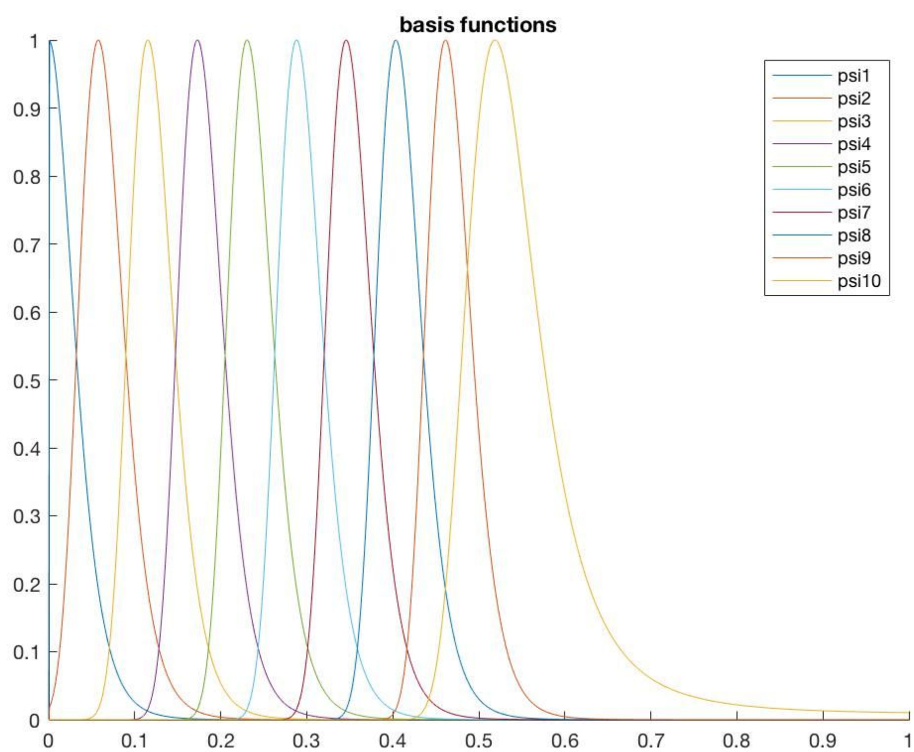
```
alpha_z=25;
beta_z=6;
alpha_c=8;
y0=0;
x0=1;
z0=0;
N=10;
g=1;
c=[1.0000 0.6294 0.3962 0.2494 0.1569 0.0988 0.0622 0.0391 0.0246 0.0155];
sigma_2 = [ 41.6667 16.3934 6.5359 2.5840 1.0235 0.4054 0.1606 0.0636 0.0252 0.0252]/1000;
w = [ 0 0 0 0 0 0 0 0 0 0];

dt = 0.001;
ticks= 0:dt:1;

basis_function=zeros(N,size(ticks,2));
yd = zeros(1001, 1);
ydd = zeros(1001, 1);
x= zeros(1001, 1);
y= zeros(1001, 1);
z= zeros(1001, 1);
target_f = zeros( size(ticks,2), 1 );

y(1) = y0;
yd(1) = yd0;
x(1) = x0;

% Calculate x for all ticks
for j = 2:1001
    dx = - alpha_x * x(j-1);
    x(j) = x(j-1) + dx * dt;
end
```


y
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


yd
5
4
3
2
1
0
-1
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

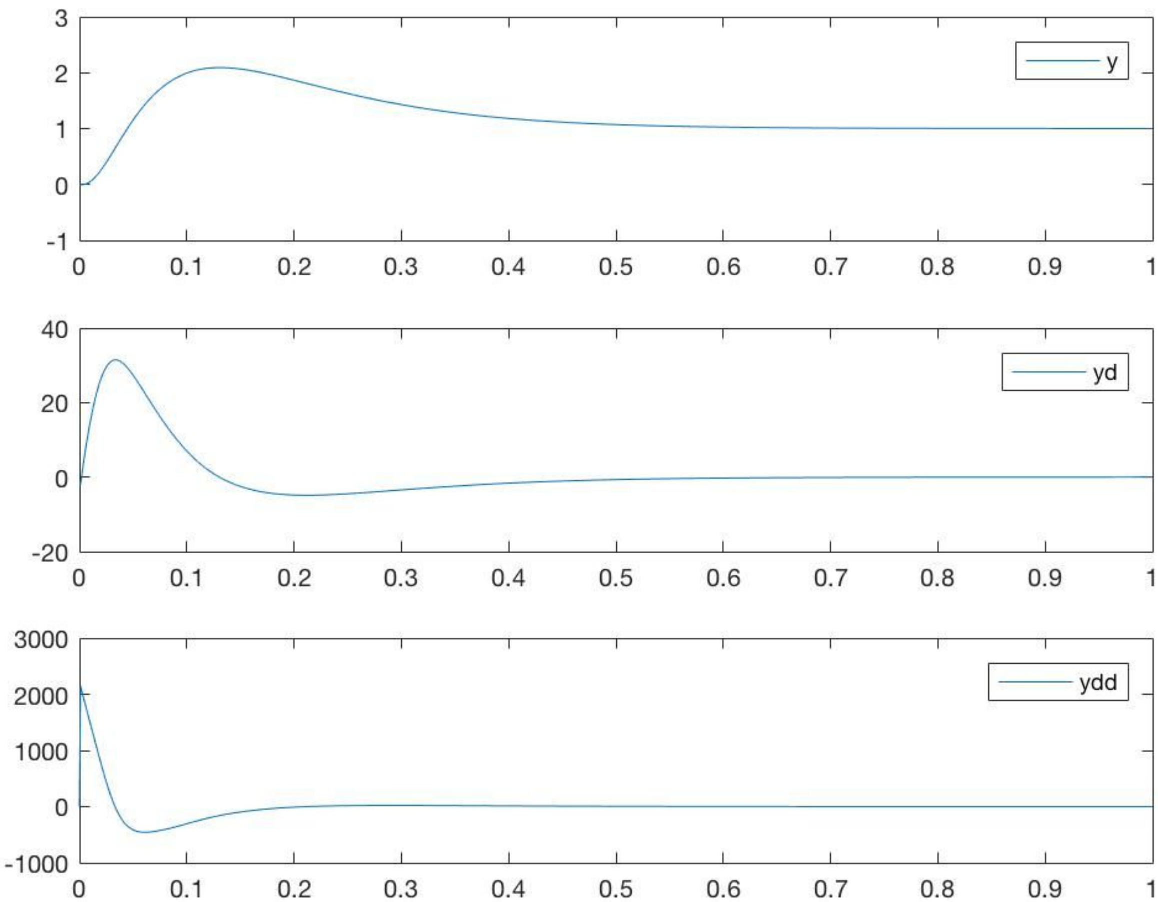

ydd
200
150
100
50
0
-50
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

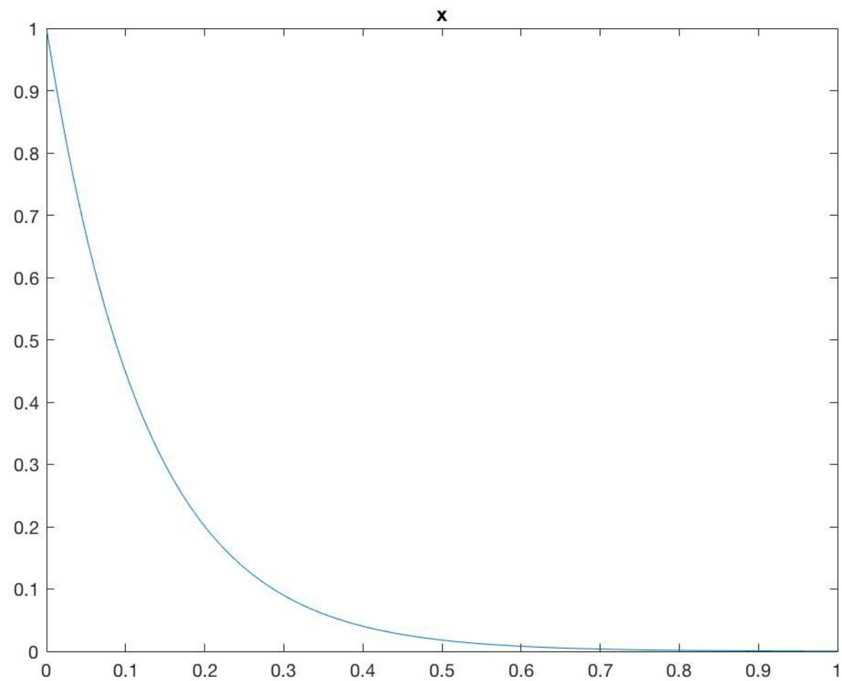

x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

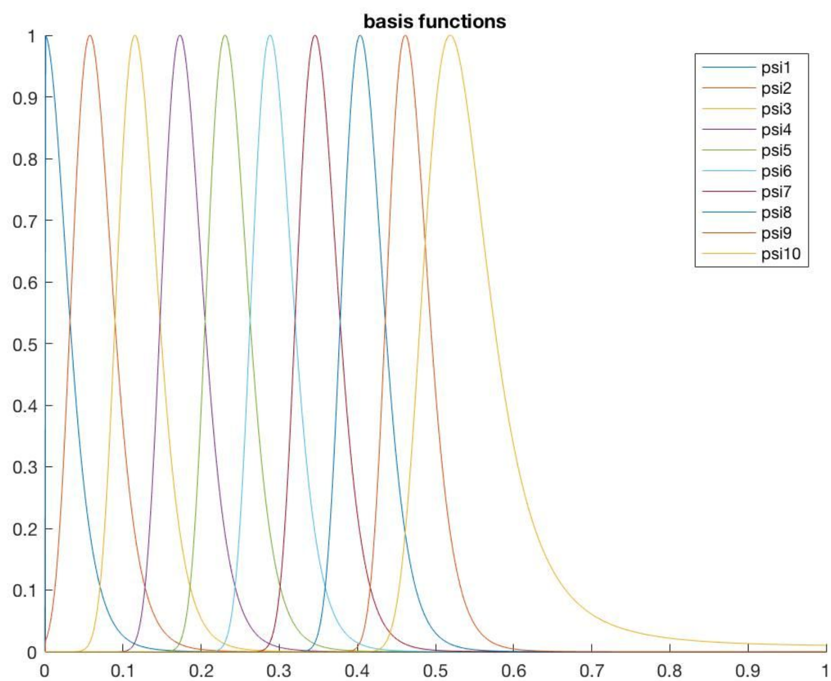

basis functions
psi1
psi2
psi3
psi4
psi5
psi6
psi7
psi8
psi9
psi10
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

2.d
The codes are basically the same as c) except that I used a different w vector.
When W = [2000 0 0 0 0 0 0 0 0 0];

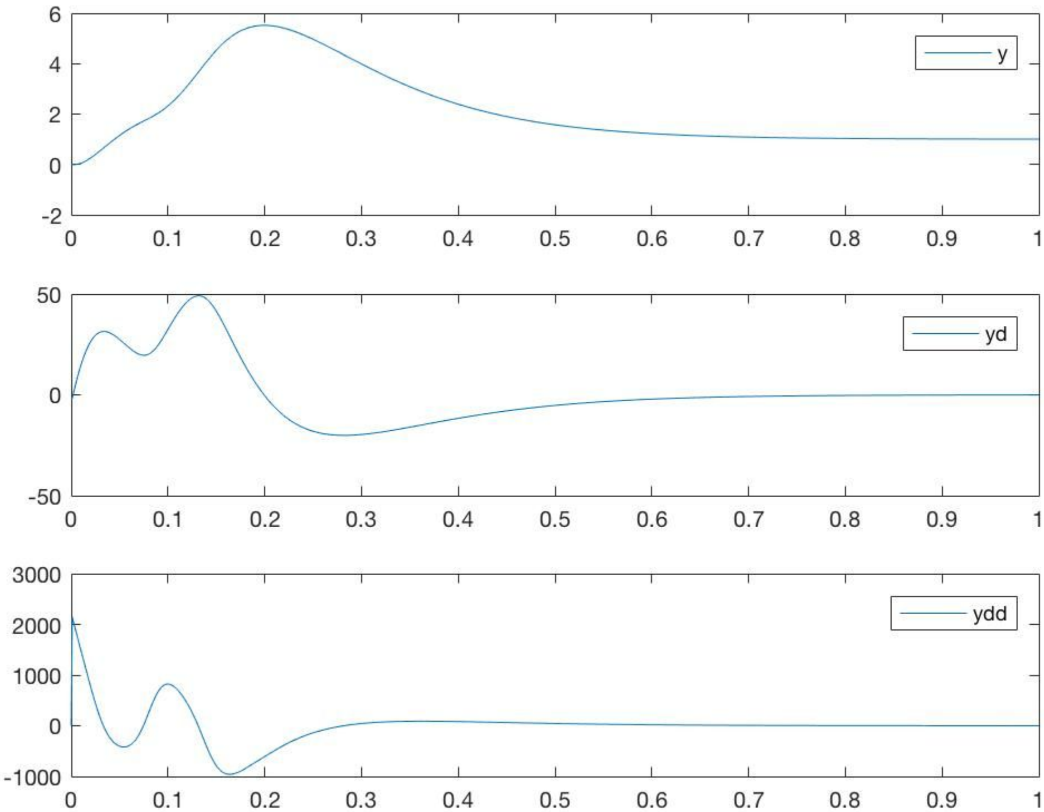
This W results in a much larger range of ydd.

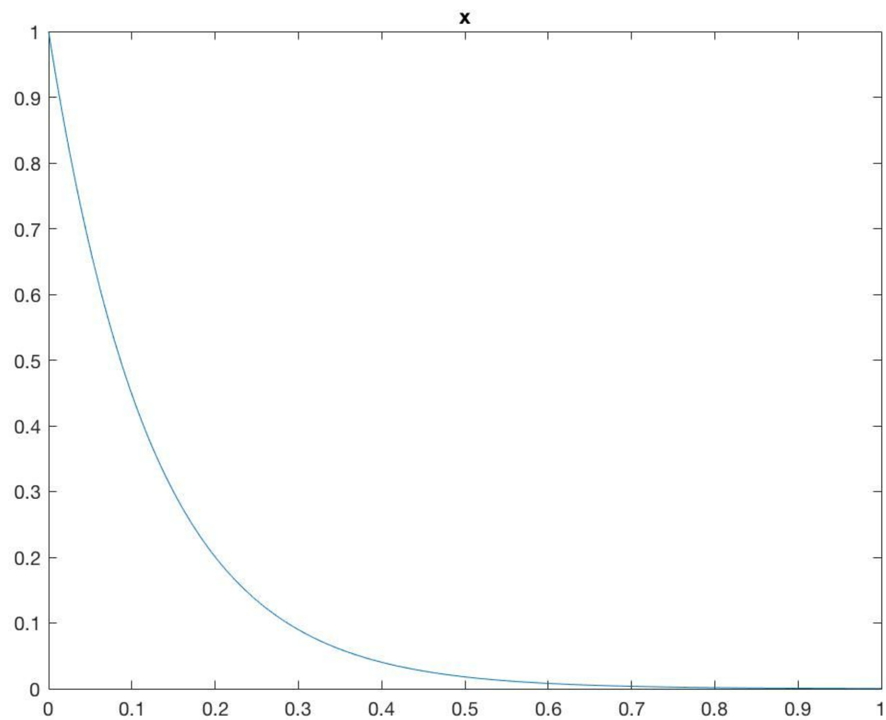


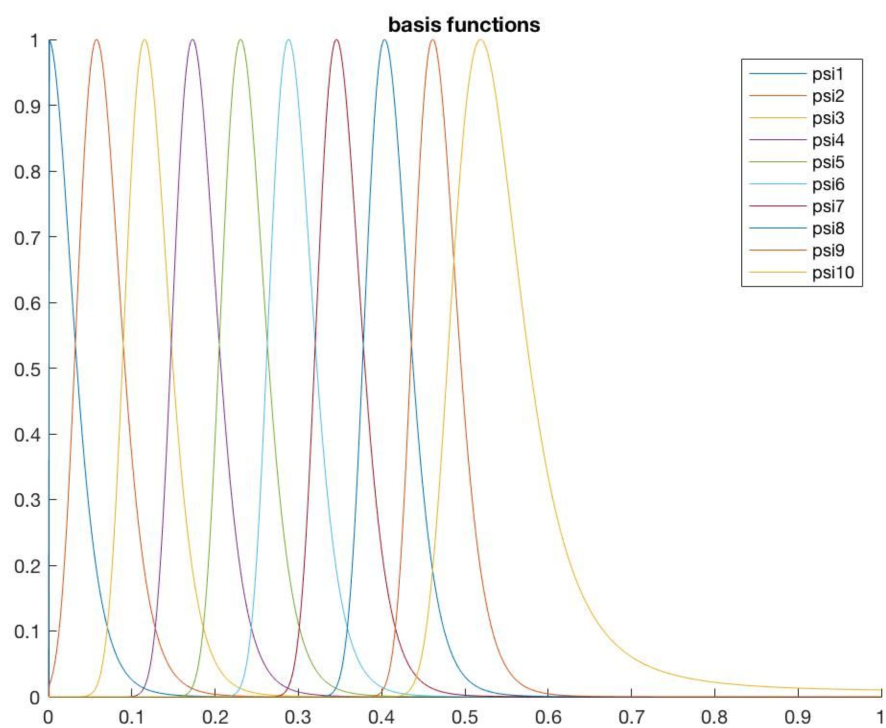

x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


basis functions
psi1
psi2
psi3
psi4
psi5
psi6
psi7
psi8
psi9
psi10
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

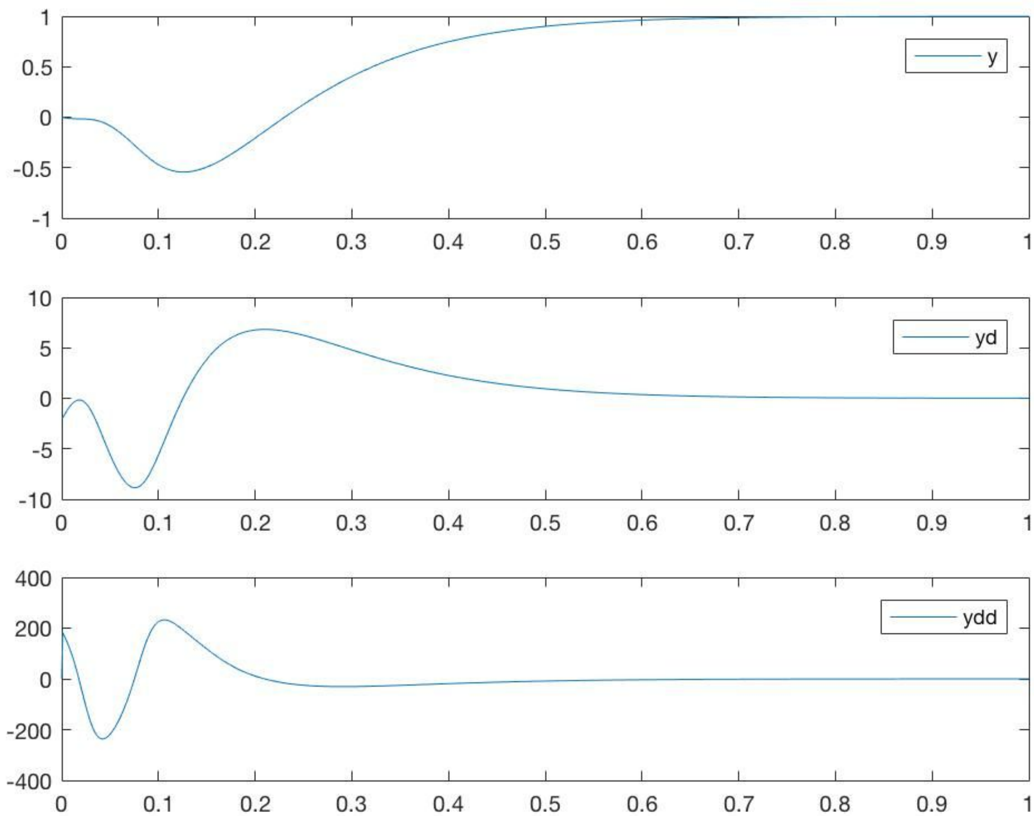
W = [2000 0 6000 0 0 0 0 0 0 0];

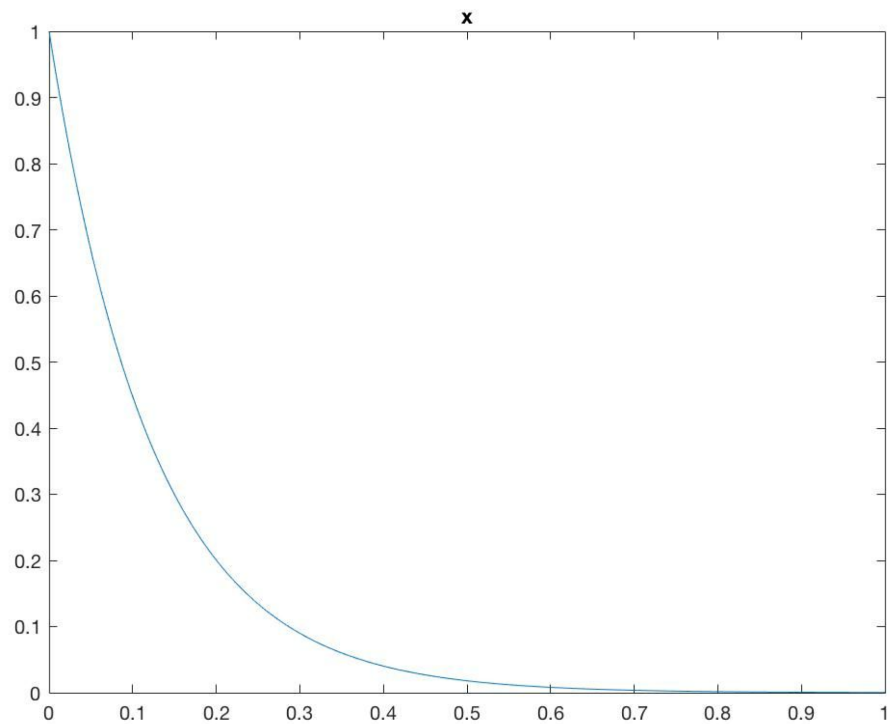


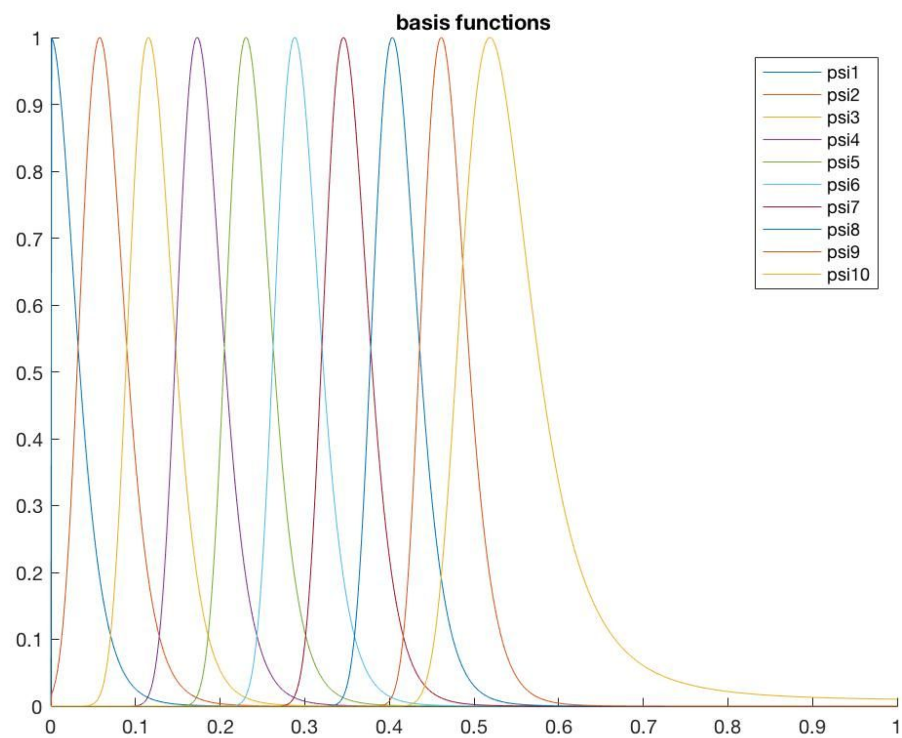

x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


basis functions
psi1
psi2
psi3
psi4
psi5
psi6
psi7
psi8
psi9
psi10
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

W = [0 -1000 0 0 0 0 0 0 0 0];

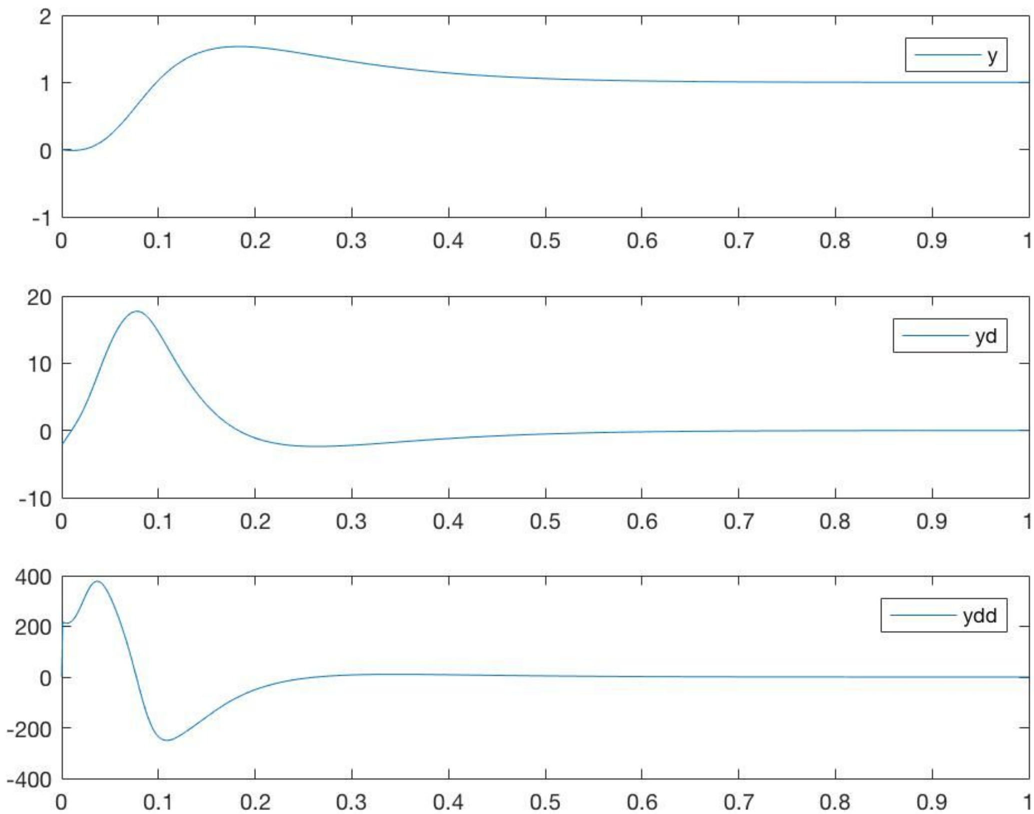


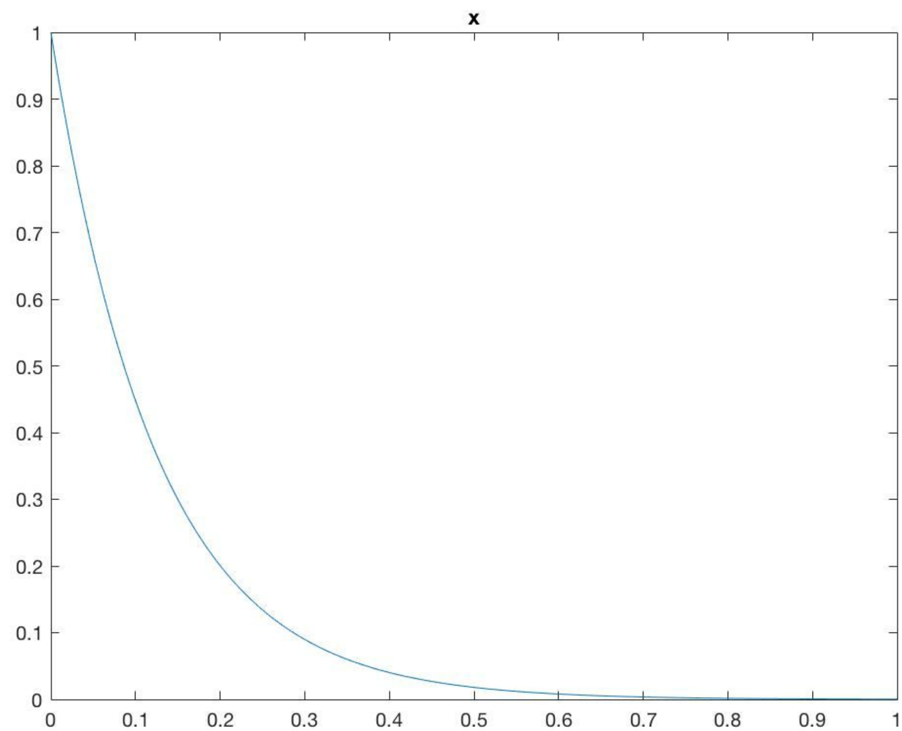

x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

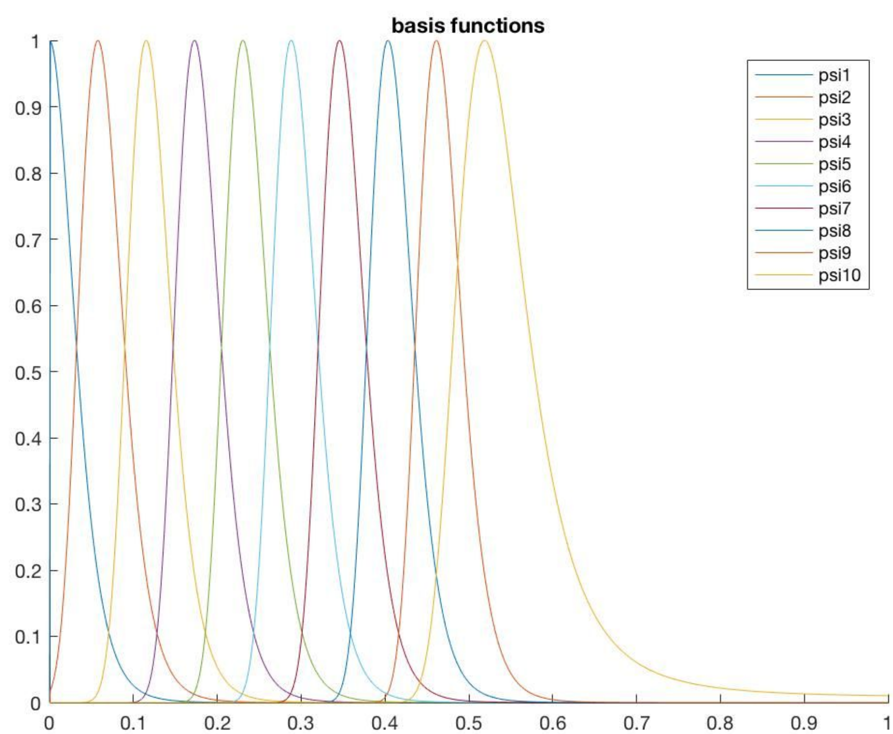

basis functions
psi1
psi2
psi3
psi4
psi5
psi6
psi7
psi8
psi9
psi10
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

W = [0 1000 0 0 0 0 0 0 0 0];

This W has behavior that is opposite to [0 -1000 0 0 0 0 0 0 0 0] for ydd.

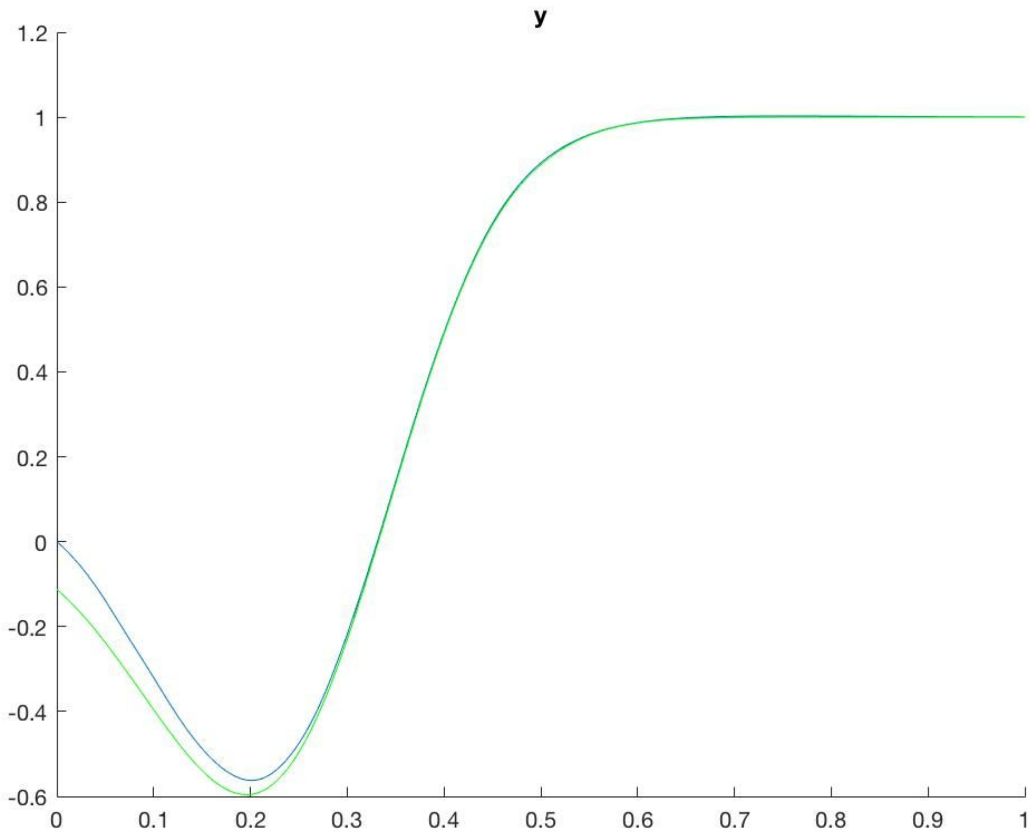


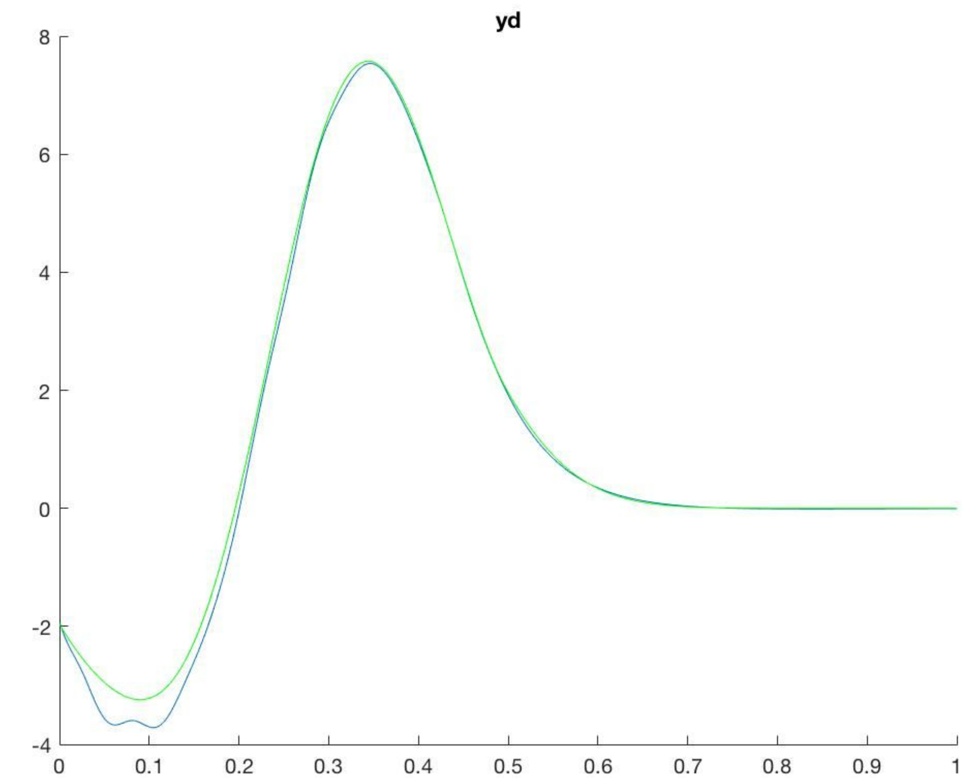

x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

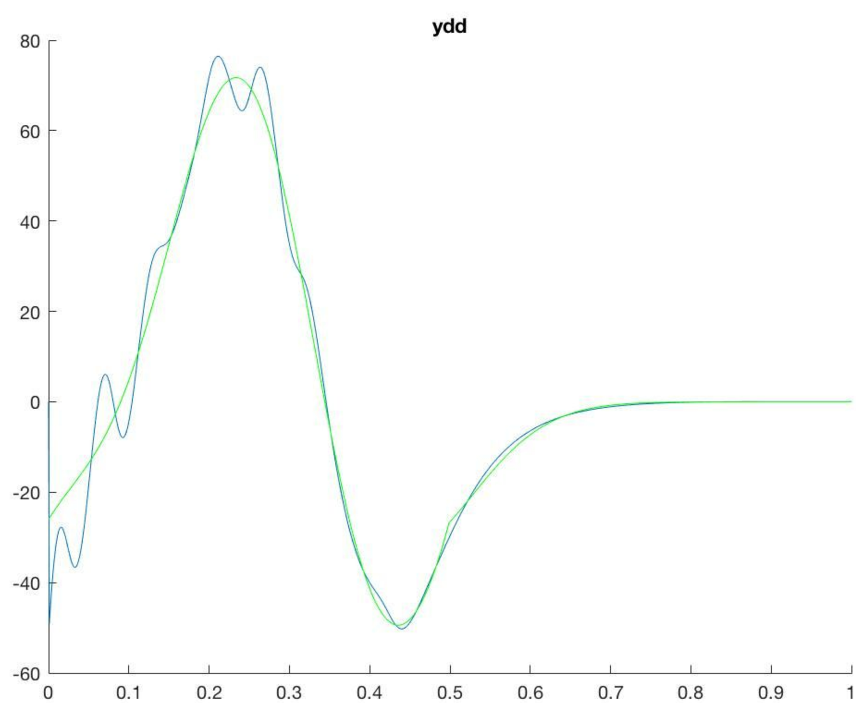

basis functions
psi1
psi2
psi3
psi4
psi5
psi6
psi7
psi8
psi9
psi10
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

1.e

According to the lecture notes, w can be learnt by the formula: w = inv(X'X)X't




yd
8
6
4
2
0
-2
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


ydd
80
60
40
20
0
-20
-40
-60
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Here, we can see that the W allows y and yd has nice imitation and slightly worse performance on ydd. Even though for y, yd and ydd did not imitate the data very well at the beginning, they all converge to a very small error and imitated the target data.