

Εργαστήριο Ψηφιακών Συστημάτων Ελέγχου  
Ακαδημαϊκό Έτος 2015-2016

**4<sup>η</sup> Εργαστηριακή Άσκηση**  
**Έλεγχος (Feedback + Prefilter)**  
**Κινούμενου Ανεστραμμένου Εκκρεμούς**

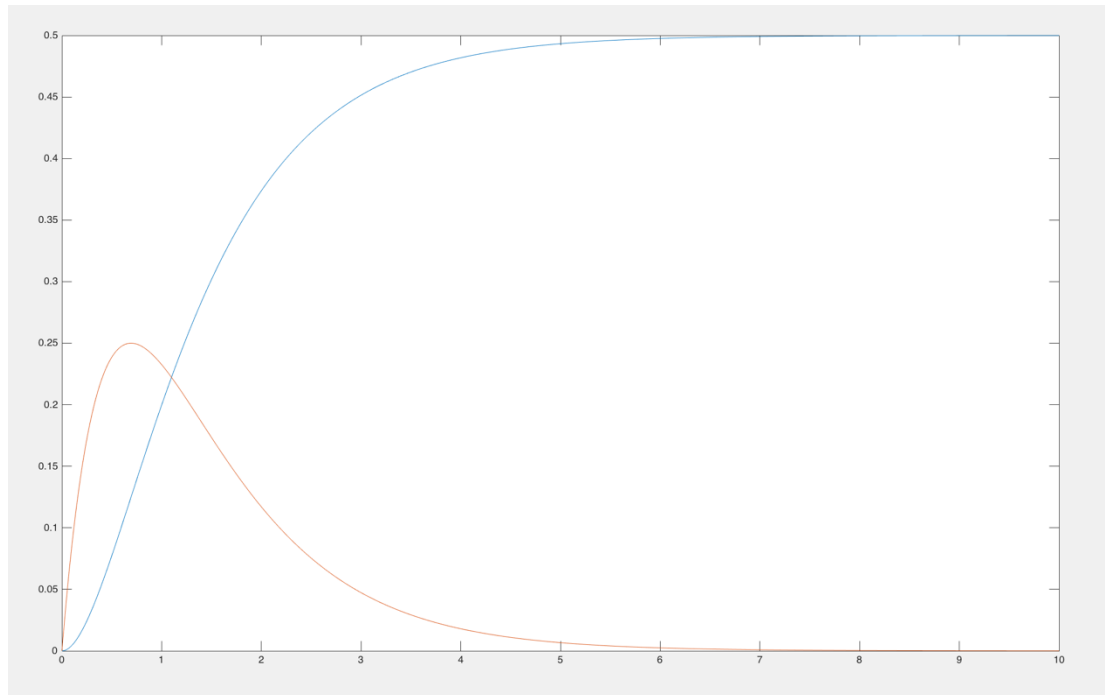
ΟΝΟΜΑ ΦΟΙΤΗΤΗ	Α.Μ.
ΜΠΟΛΑΤΟΓΛΟΥ ΓΕΩΡΓΙΟΣ	228424

**4.1 Ανατροφοδότηση θέσης και ταχύτητας**  
**βάσης κινούμενου εκκρεμούς**

Υλοποιούμε ελεγκτή  $K = [K_1 \ K_2]$  για το  $Dx$  και  $\dot{Dx}$  που να μεταφέρει τους πόλους

του κλειστού στο -1, -2. Ο κώδικας στην matlab είναι ο εξής:

```
1 -   clc;
2 -   clear;
3 -   M=1;m=1;l=1;B_l=0.3;B_r=0.3;g=10;
4
5 -   A_lin=zeros(4,4);B_lin=zeros(4,1);
6
7 -   A_lin(1,2)=1;
8 -   A_lin(2,2)=-B_l/M;
9 -   A_lin(2,3)=m*g/M;
10 -  A_lin(2,4)=-B_r/l/M;
11 -  A_lin(3,4)=1;
12 -  A_lin(4,2)=-B_l/M/l;
13 -  A_lin(4,3)=(m*g/M/l)+g/l;
14 -  A_lin(4,4)=(-B_r/m/l^2)+(-B_r/M/l^2);
15
16 -  B_lin(2)=1/M;B_lin(4)=1/M/l;
17 -  C_lin=eye(4);
18 -  D_lin=zeros(4,1);
19   % state space form for x and \dot{x}
20 -  A_xdx=A_lin([1:2],[1:2]);B_xdx=B_lin([1:2],:);
21   % u=K_1 x + K_2 \dot{x}
22 -  K_f_xdx=place(A_xdx,B_xdx,[-1;-2]);
23 -  A_xdx_cl=A_xdx-B_xdx*K_f_xdx;
24 -  t=[0:0.01:10];u=ones(size(t));
25 -  C_xdx=eye(2);D_xdx=[0;0];
26 -  xdx_cl=lsim(A_xdx_cl,B_xdx,C_xdx,D_xdx,u,t);
27 -  plot(t,xdx_cl);
```



Το ζητούμενο όμως, είναι αν ένας τέτοιος ελεγκτής μπορεί να οδηγήσει σε ευστάθεια το σύστημα (5) για  $K = [K_1 \ K_2 \ 0 \ 0]$ .

Το σύστημα (5) είναι

$$\Delta \dot{z} = \left[ A - \begin{bmatrix} \frac{0}{K_1} & \frac{0}{K_2} & \frac{0}{K_3} & \frac{0}{K_4} \\ \frac{M}{0} & \frac{M}{0} & \frac{M}{0} & \frac{M}{0} \\ \frac{K_1}{Ml} & \frac{K_2}{Ml} & \frac{K_3}{Ml} & \frac{K_4}{Ml} \end{bmatrix} \right] \Delta z + B N \Delta x^d.$$

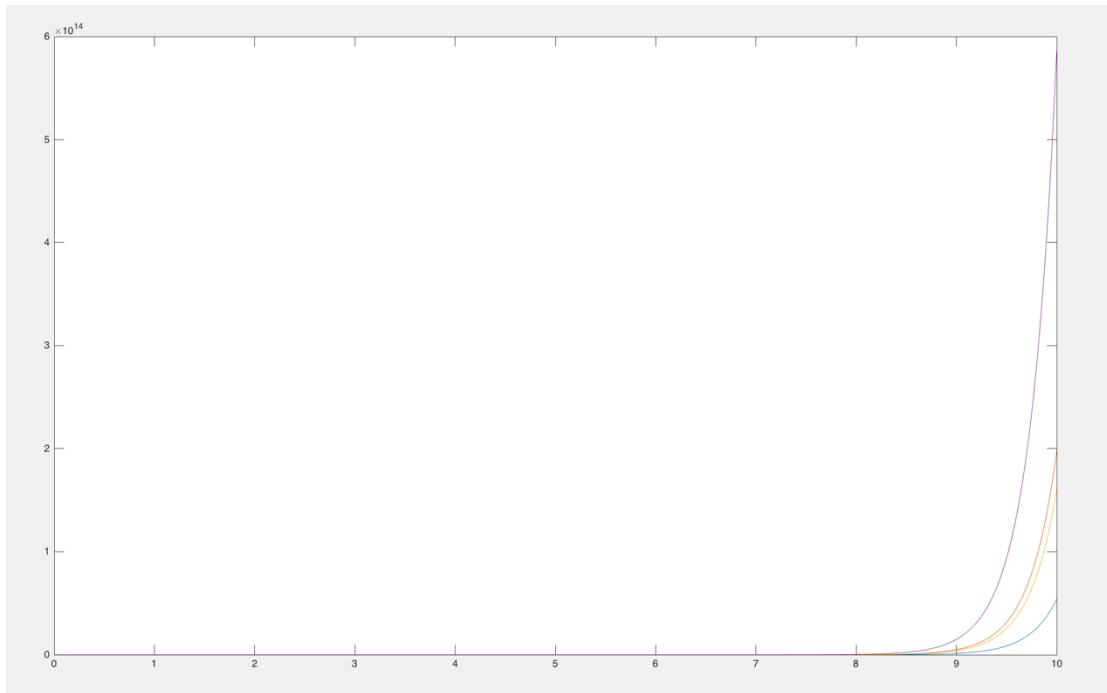
$$\Delta \dot{z} = C \Delta z$$

Υλοποιούμε τον παρακάτω κώδικα στην matlab:

```
1 - M=1;m=1;l=1;B_l=0.3;B_r=0.3;g=10;
2
3 - A_lin=zeros(4,4);B_lin=zeros(4,1);
4
5 - A_lin(1,2)=1;
6 - A_lin(2,2)=-B_l/M;
7 - A_lin(2,3)=m*g/M;
8 - A_lin(2,4)=-B_r/l/M;|
9 - A_lin(3,4)=1;
10 - A_lin(4,2)=-B_l/M/l;
11 - A_lin(4,3)=(m*g/M/l)+g/l;
12 - A_lin(4,4)=(-B_r/m/l^2)+(-B_r/M/l^2);
13
14 - B_lin(2)=1/M;B_lin(4)=1/M/l;
15 - C_lin=eye(4);
16 - D_lin=zeros(4,1);
17 % state space form for x and \dot{x}
18 - A_xdx=A_lin([1:2],[1:2]),B_xdx=B_lin([1:2],:)
19 % u=K_1 x + K_2 \dot{x}
20 - K_f_xdx=place(A_xdx,B_xdx,[-1;-2]);
21 - A_xdx_cl=A_xdx-B_xdx*K_f_xdx;
22 - t=[0:0.01:10];u=ones(size(t));
23 - C_xdx=eye(2);D_xdx=[0;0];
24 - xdx_cl=lsim(A_xdx_cl,B_xdx,C_xdx,D_xdx,u,t);
25 %plot(t,xdx_cl);
26
27 - K_f=[K_f_xdx 0 0];
28 - A_f=A_lin-B_lin*K_f
29 - xdx_cl_2=lsim(A_f,B_lin,C_lin,D_lin,u,t);
30 - plot(t,xdx_cl_2);
```

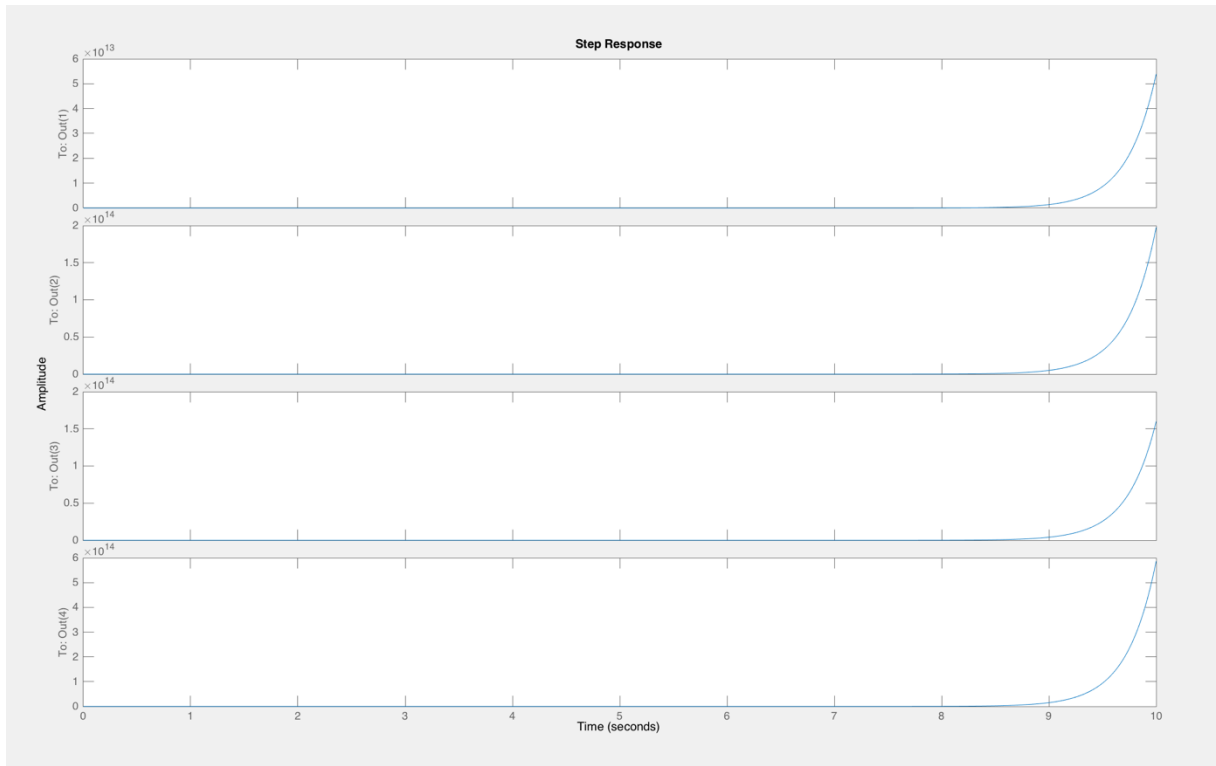
Και βρίσκουμε τον παρακάτω ελεγκτή:

$K_{f\_xdx} = [2,0 \quad 2,7 \quad 0 \quad 0]$



Α)Βηματική απόκριση

```
33 - G = ss(A_f,B_lin,C_lin,D_lin);  
34 - step(G,t);
```



B)

Όπως βλέπουμε από την βηματική απόκριση η απόκριση μετά από κάποιο χρόνο  $\tau$  απειρίζεται, άρα το σύστημα πάει στην αστάθεια.

Επίσης ένας άλλος τρόπος να ελέγξουμε αν το σύστημα είναι ευσταθές είναι με την εντολή `damp`. Όπως βλέπουμε ο ένας πόλος (στο 3.66) είναι στο δεξί ημιεπίπεδο, πράγμα που επίσης δηλώνει αστάθεια

35 - `damp(G)`

Pole	Damping	Frequency (rad/seconds)	Time Constant (seconds)
-7.41e-01 + 6.28e-01i	7.63e-01	9.72e-01	1.35e+00
-7.41e-01 - 6.28e-01i	7.63e-01	9.72e-01	1.35e+00
3.66e+00	-1.00e+00	3.66e+00	-2.73e-01
-5.78e+00	1.00e+00	5.78e+00	1.73e-01

Τέλος θα μπορούσα να χρησιμοποιήσουμε και την εντολή `isstable` η οποία επιστρέφει το λογικό 1 αν το σύστημα είναι ευσταθές και 0 αν είναι ασταθές.

B `isstable(G)`

B =

0

Γ)

Με την εντολή `ss2tf` βρίσκουμε τον αριθμητή και τον παρονομαστή της συνάρτησης μεταφοράς μας. Όμως, το `n(numerator)` αποτελείται από 4 γραμμές όπου κάθε μία γραμμή αντιστοιχεί στο  $\Delta x$ ,  $\Delta x \dot{}$ ,  $Dth$ ,  $Dth \dot{}$  αντίστοιχα.

```
33 - G = ss(A_f,B_lin,C_lin,D_lin);  
34 - [n,d]=ss2tf(A_f,B_lin,C_lin,D_lin)
```

n =

0	0	1.0000	0.3000	-10.0000
0	1.0000	0.3000	-10.0000	0
0	0	1.0000	-0.0000	-0.0000
0	1.0000	0	0	0

d =

1.0000	3.6000	-17.1000	-29.4000	-20.0000
--------	--------	----------	----------	----------

Επομένως για να βρούμε τις 4 συναρτήσεις μεταφοράς θα χρησιμοποιήσουμε για κάθε μία την γραμμή που της αντιστοιχεί.

Με την εντολή `tf` δημιουργούμε την συνάρτηση μεταφοράς.

Με τις εντολές `pole`, `zero` βρίσκουμε τα μηδενικά και του πόλους κάθε μίας.

Με την εντολή `dcgain` βρίσκουμε το σταθερό κέρδος της συνάρτησης. Αυτό προκύπτει παίρνοντας το όριο για  $s \rightarrow 0$  της συν. μεταφοράς.

```
36 - n1=n(1,:);
37 - Dx == tf(n1,d)
38 - poles==pole(Dx)
39 - zeroes == zero(Dx)
40 - k == dcgain(Dx)
41
42 - n2=n(2,:);
43 - Dxdot == tf(n2,d)
44 - poles==pole(Dxdot)
45 - zeroes == zero(Dxdot)
46 - k == dcgain(Dxdot)
47
48 - n3=n(3,:);
49 - Dth == tf(n3,d)
50 - poles==pole(Dth)
51 - zeroes == zero(Dth)
52 - k == dcgain(Dth)
53
54 - n4=n(4,:);
55 - Dthdot == tf(n4,d)
56 - poles==pole(Dthdot)
57 - zeroes == zero(Dthdot)
58 - k == dcgain(Dthdot)
```



Dx =

$$\frac{s^2 + 0.3 s - 10}{s^4 + 3.6 s^3 - 17.1 s^2 - 29.4 s - 20}$$

Continuous-time transfer function.

poles =

```
-5.7816 + 0.0000i  
 3.6639 + 0.0000i  
-0.7411 + 0.6284i  
-0.7411 - 0.6284i
```

zeroes =

```
-3.3158  
 3.0158
```

k =

```
0.5000
```

Dxdot =

$$\frac{s^3 + 0.3 s^2 - 10 s}{s^4 + 3.6 s^3 - 17.1 s^2 - 29.4 s - 20}$$

Continuous-time transfer function.

poles =

```
-5.7816 + 0.0000i  
 3.6639 + 0.0000i  
-0.7411 + 0.6284i  
-0.7411 - 0.6284i
```

zeroes =

```
 0  
-3.3158  
 3.0158
```

k =

```
0
```

Dth =

$$\frac{s^2 - 1.527e-16 s - 2.761e-16}{s^4 + 3.6 s^3 - 17.1 s^2 - 29.4 s - 20}$$

Continuous-time transfer function.

poles =

$$\begin{aligned} &-5.7816 + 0.0000i \\ &3.6639 + 0.0000i \\ &-0.7411 + 0.6284i \\ &-0.7411 - 0.6284i \end{aligned}$$

zeroes =

$$\begin{aligned} &1.0e-07 * \\ &0.1662 \\ &-0.1662 \end{aligned}$$

k =

$$1.3805e-17$$

Εδώ επειδή ο συντελεστής του s και του σταθερού όρου είναι πάρα πολλοί μικροί θα μπορούσαμε προσεγγιστικά να τους πάρουμε ίσους με το 0 και έτσι θα έμενε μόνο ο s<sup>2</sup>, δηλαδή θα είχαμε 2 μηδενικά στο 0.

Dthdot =

$$\frac{s^3}{s^4 + 3.6 s^3 - 17.1 s^2 - 29.4 s - 20}$$

Continuous-time transfer function.

poles =

-5.7816 + 0.0000i  
3.6639 + 0.0000i  
-0.7411 + 0.6284i  
-0.7411 - 0.6284i

zeroes =

0  
0  
0

k =

0

## 4.2 Ανατροφοδότηση γωνίας και γωνιακής ταχύτητας του περιστρεφόμενου εκκρεμούς

Δ) Υλοποιούμε ελεγκτή  $K = [K_3 \ K_4]$  για το  $Dx$  και  $Dx_{dot}$  που να μεταφέρει τους πόλους του κλειστού στο -3, -4

```
1 - M=1;m=1;l=1;B_l=0.3;B_r=0.3;g=10;
2
3 - A_lin=zeros(4,4);B_lin=zeros(4,1);
4
5 - A_lin(1,2)=1;
6 - A_lin(2,2)=-B_r/(m*l^2)-B_r/(M*l^2);
7 - A_lin(2,1)=(m+M)*g/M*l;
8 - A_lin(2,3)=m*g/M;
9 - A_lin(2,4)=-B_r/l/M;
10 - A_lin(3,4)=1;
11 - A_lin(4,2)=-B_l/M/l;
12 - A_lin(4,3)=(m*g/M/l)+g/l;
13 - A_lin(4,4)=(-B_r/m/l^2)+(-B_r/M/l^2);
14
15 - B_lin(2)=1/M*l;
16 - B_lin(4)=1/M/l;
17
18 - C_lin=eye(4);
19 - D_lin=zeros(4,1);
20 % state space form for x and \dot{x}
21 - A_xdx=A_lin([1:2],[1:2]),B_xdx=B_lin([1:2],:);
22 % u=K_1 x + K_2 \dot{x}
23 - K_f_xdx=place(A_xdx,B_xdx,[-3;-4]);
24 - A_xdx_cl=A_xdx-B_xdx*K_f_xdx;
25 - t=[0:0.01:10];u=ones(size(t));
26 - C_xdx=eye(2);D_xdx=[0;0];
27 - xdx_cl=lsim(A_xdx_cl,B_xdx,C_xdx,D_xdx,u,t);
28 - plot(t,xdx_cl);
```

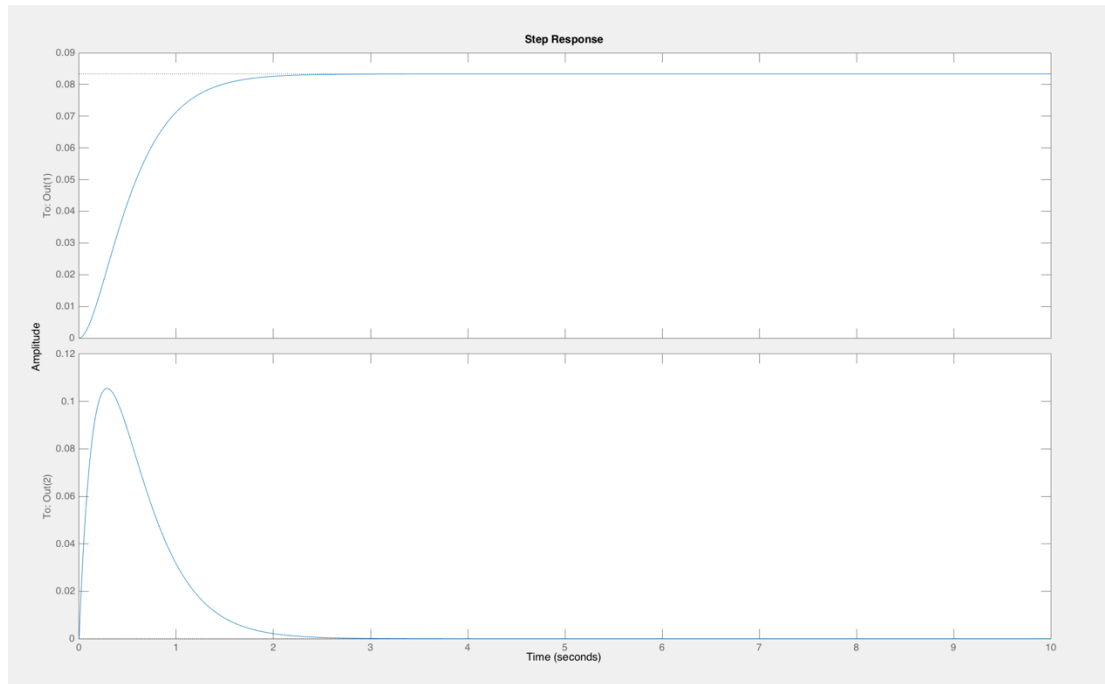
Για την βηματική απόκριση:

```
G = ss(A_xdx_cl,B_xdx,C_xdx,D_xdx);
```

```
step(G,t);
```

Όπου  $t=[0:0.01:10]$

Υπολογίζεται  $K_f x_{dx} = [32 \quad 6,4]$



E)

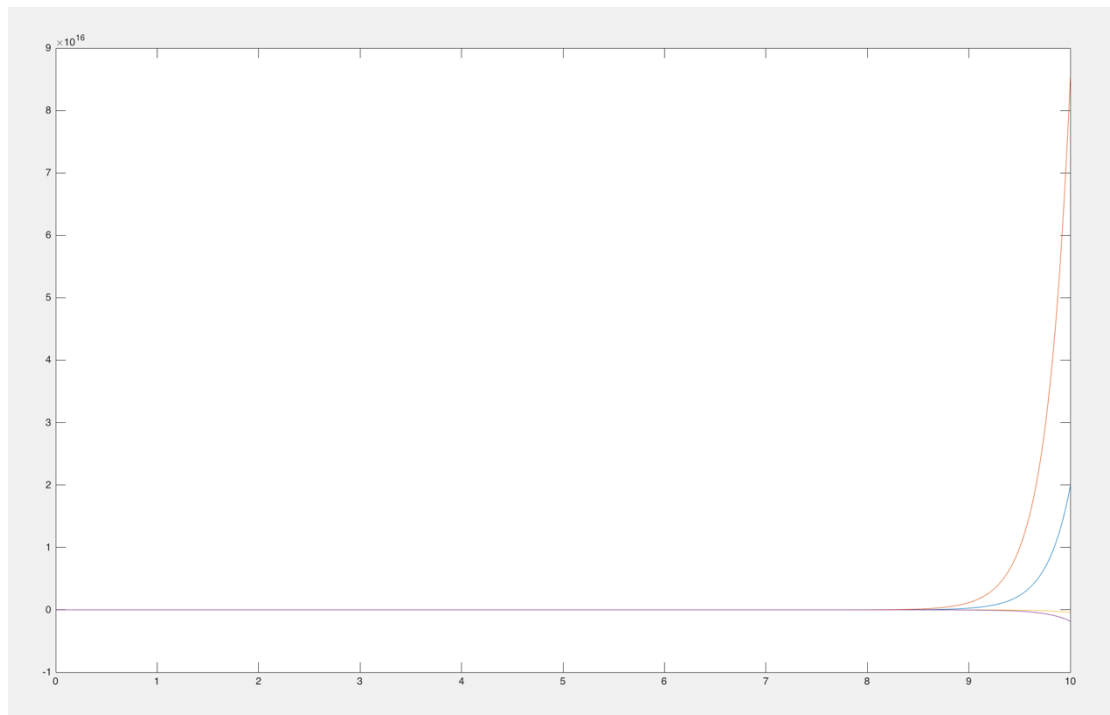
Όπως βλέπουμε από την βηματική απόκριση σταθεροποιείται σε κάποια τιμή.

Άρα το σύστημα είναι ευσταθές.

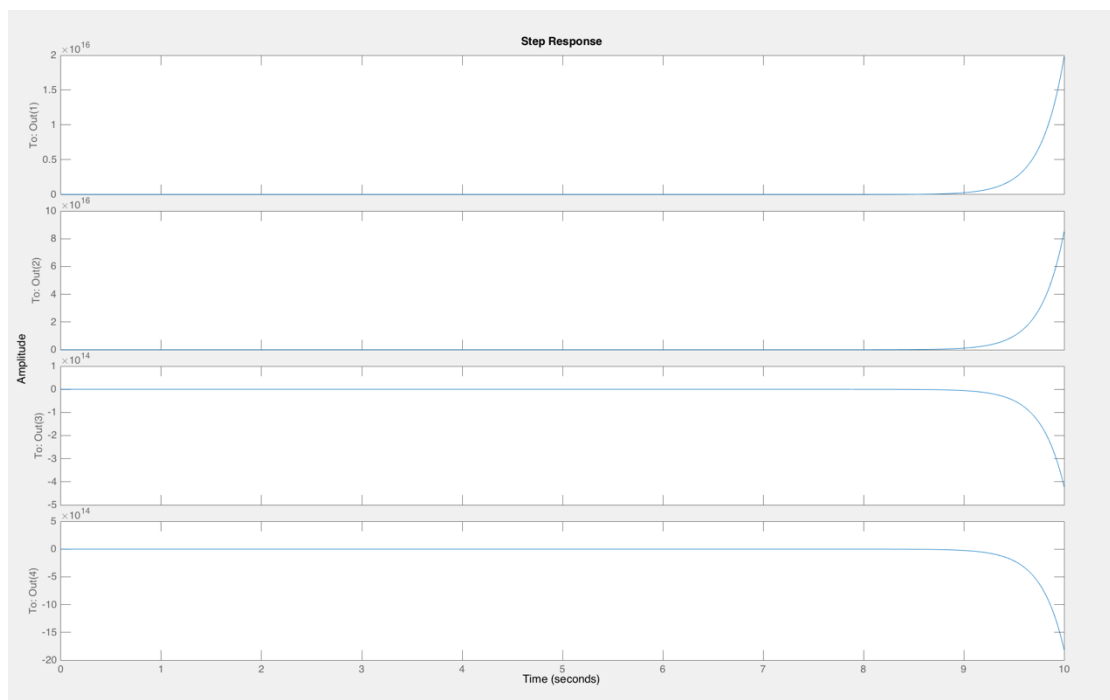
ΣΤ)

Το ζητούμενο είναι αν ένας τέτοιος ελεγκτής μπορεί να οδηγήσει σε ευστάθεια το σύστημα (5) για  $K=[0 \ 0 \ K_3 \ K_4]$ .

```
32 - K_f=[0 0 K_f_xdx];  
33 - A_f=A_lin-B_lin*K_f;  
34 - xdx_cl_2=lsim(A_f,B_lin,C_lin,D_lin,u,t);  
35 - plot(t,xdx_cl_2);
```



```
G = ss(A_f,B_lin,C_lin,D_lin);  
step(G,t);
```



H)

Βλέπουμε από την γραφική πως η βηματική απόκριση απειρίζεται, κάτι που δηλώνει αστάθεια.

Επίσης όπως και στο παραπάνω ερώτημα μπορούμε να ελέγξουμε την αστάθεια και με το damp όπου βλέπουμε πως υπάρχει πόλος στο 4,3 και με το isstable που επιστρέφει λογικό 0.

```
G = ss(A_f,B_lin,C_lin,D_lin)
damp(G)
```

Pole	Damping	Frequency (rad/seconds)	Time Constant (seconds)
-2.84e+00	1.00e+00	2.84e+00	3.52e-01
-3.59e+00	1.00e+00	3.59e+00	2.78e-01
4.30e+00	-1.00e+00	4.30e+00	-2.32e-01
-5.47e+00	1.00e+00	5.47e+00	1.83e-01

```
B = isstable(G)
```

B =

0

Θ)

Ομοίως όπως και στο παραπάνω ερώτημα για να βρούμε τις συναρτήσεις μεταφοράς χρησιμοποιούμε τις εξής εντολές:

```
38 - [n,d]=ss2tf(A_f,B_lin,C_lin,D_lin);
39 - n1=n(1,:);
40 - Dx = tf(n1,d)
```

Βρίσκουμε numerator και denominator και χρησιμοποιούμε κάθε γραμμή του num ξεχωριστά για τις αντίστοιχες tf.

n =

```
0      0      1.0000      0.3000 -10.0000
0      1.0000      0.3000 -10.0000      0
0      0      1.0000      0.3000 -20.0000
0      1.0000      0.3000 -20.0000      0
```

d =

```
1.0000      7.6000 -5.8100 -139.4000 -240.0000
```



```
39 - Dx = tf(n1,d);  
40 - poles = pole(Dx)  
41 - zeroes = zero(Dx)  
42 - k = dcgain(Dx)
```

Dx =

$$\frac{s^2 + 0.3 s - 10}{s^4 + 7.6 s^3 - 5.81 s^2 - 139.4 s - 240}$$

Continuous-time transfer function.

poles =

```
4.3013  
-5.4686  
-3.5927  
-2.8399
```

zeroes =

```
-3.3158  
3.0158
```

k =

```
0.0417
```

```
38 - n2=n(2,:);  
39 - Dxdot = tf(n2,d)  
40 - poles=pole(Dxdot)  
41 - zeroes = zero(Dxdot)  
42 - k = dcgain(Dxdot)
```

Dxdot =

$$\frac{s^3 + 0.3 s^2 - 10 s}{s^4 + 7.6 s^3 - 5.81 s^2 - 139.4 s - 240}$$

Continuous-time transfer function.

poles =

```
4.3013  
-5.4686  
-3.5927  
-2.8399
```

zeroes =

```
0  
-3.3158  
3.0158
```

k =

```
0
```

Dth =

$$\frac{s^2 + 0.3 s - 20}{s^4 + 7.6 s^3 - 5.81 s^2 - 139.4 s - 240}$$

Continuous-time transfer function.

poles =

4.3013  
-5.4686  
-3.5927  
-2.8399

zeroes =

-4.6247  
4.3247

k =

0.0833

Dthdot =

$$\frac{s^3 + 0.3 s^2 - 20 s}{s^4 + 7.6 s^3 - 5.81 s^2 - 139.4 s - 240}$$

Continuous-time transfer function.

poles =

4.3013  
-5.4686  
-3.5927  
-2.8399

zeroes =

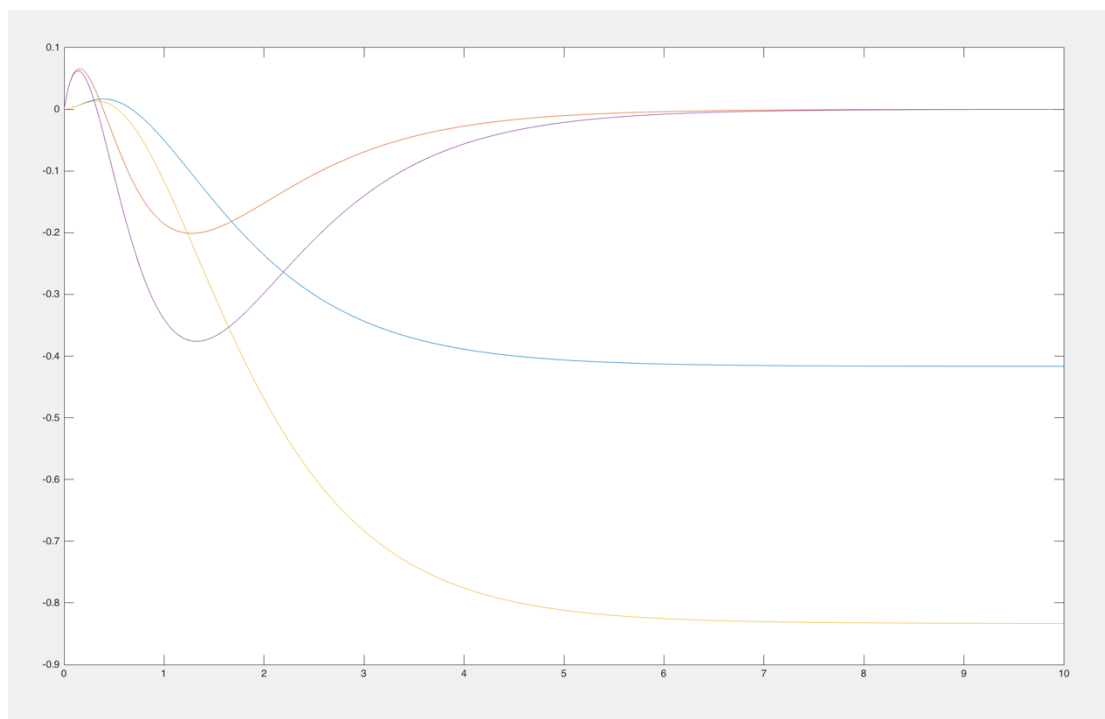
0  
-4.6247  
4.3247

k =

0

## 4.3 Ανατροφοδότηση θέσης, γραμμικής ταχύτητας, γωνίας και γωνιακής ταχύτητας του κινούμενου/περιστρεφόμενου εκκρεμούς

```
EDITOR PUBLISH VIEW
New Open Save Find Files Compare Go To Comment % fx f6
Insert Comment % fx f6 Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE EDIT BREAKPOINTS RUN
lab04.m lab04_2.m lab04_3.m
1 - clc;
2 - clear;
3
4 - M=1;m=1;l=1;B_l=0.3;B_r=0.3;g=10;
5
6 - A_lin=zeros(4,4);B_lin=zeros(4,1);
7
8 - A_lin(1,2)=1;
9 - A_lin(2,2)=-B_r/(m*l^2)-B_r/(M*l^2);
10 - A_lin(2,1)=(m+M)*g/M*l;
11 - A_lin(2,3)=m*g/M;
12 - A_lin(2,4)=-B_r/l/M;
13 - A_lin(3,4)=1;
14 - A_lin(4,2)=-B_l/M/l;
15 - A_lin(4,3)=(m*g/M/l)+g/l;
16 - A_lin(4,4)=(-B_r/m/l^2)+(-B_r/M/l^2);
17
18 - B_lin(2)=1/M*l;
19 - B_lin(4)=1/M/l;
20
21 - C_lin=eye(4);
22 - D_lin=zeros(4,1);
23 % state space form for x and \dot{x}
24 % use K_1 x + K_2 \dot{x}
25 - K_f_xdx=place(A_lin,B_lin,[-1;-2;-3;-4]);
26 - A_lin_2=A_lin-B_lin*K_f_xdx;
27 - t=[0:0.01:10];u=ones(size(t));
28 - xdx_cl=lsim(A_lin_2,B_lin,C_lin,D_lin,u,t);
29 - plot(t,xdx_cl);
30 - G = ss(A_lin_2,B_lin,C_lin,D_lin);
31 - step(G,t);
32
33
```



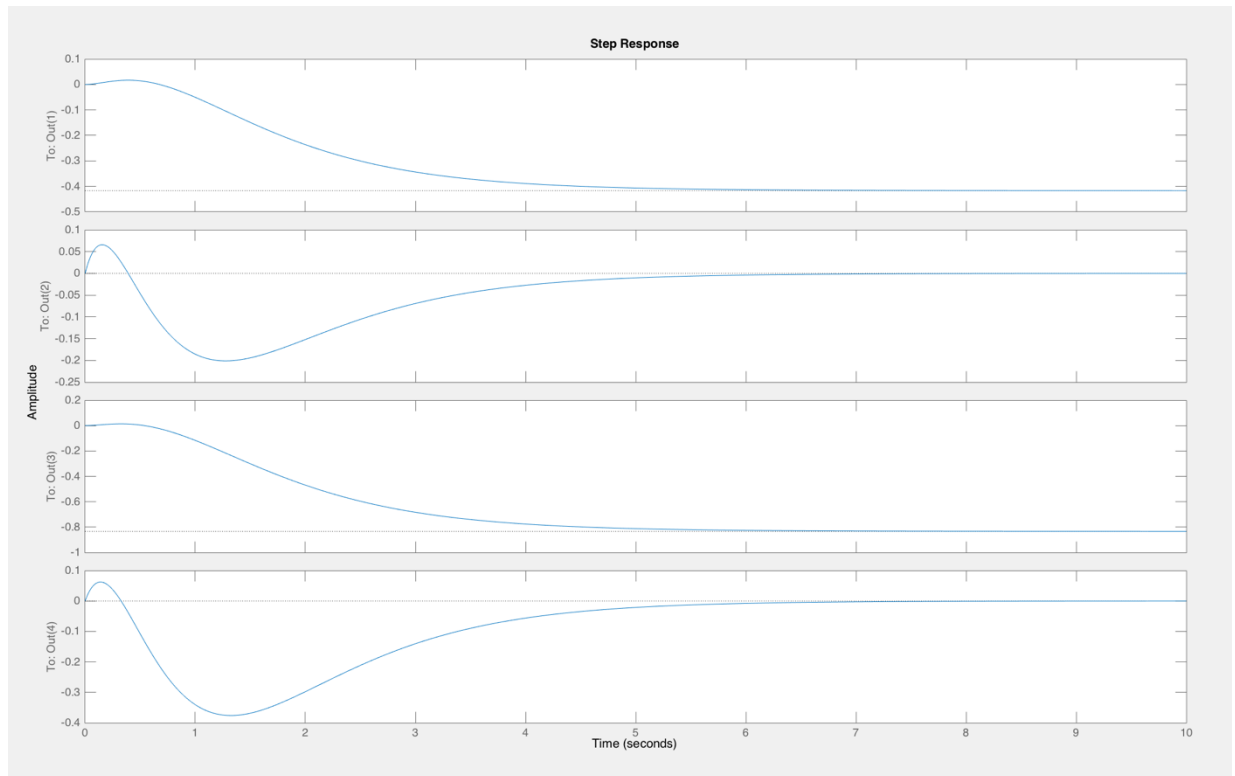
l)  
K\_f\_xdx=place(A\_lin,B\_lin,[-1;-2;-3;-4]);  
A\_lin\_2=A\_lin-B\_lin\*K\_f\_xdx;  
t=[0:0.01:10];  
u=ones(size(t));

Υπολογίζονται τα  $K = [K_1 \ K_2 \ K_3 \ K_4]$

$$K_{f\_xdx} = [106,579 \quad 22,537 \quad -34,489 \quad -13,737]$$

K)

Με την εντολή `step(G,t)` όπως φαίνεται στον παραπάνω κώδικα:



Λ)

Παρατηρούμε πως και οι 4 αποκρίσεις σταθεροποιούνται σε κάποια τιμή, άρα το σύστημα είναι ευσταθές.

Επίσης η εντολή `isstable` μας επιστρέφει λογικό 1, που σημαίνει ότι η G είναι ευσταθής.

`B=isstable (G)`

`B = 1`

M)

Ομοίως όπως και στα προηγούμενα ερωτήματα έχουμε:

```
32 - [n,d]=ss2tf(A_lin_2,B_lin,C_lin,D_lin);
33
34 - n1=n(1,:);
35 - Dx = tf(n1,d)
36 - poles=pole(Dx)
37 - zeroes = zero(Dx)
38 - k = dcgain(Dx)
39
40 - n2=n(2,:);
41 - Dxdot = tf(n2,d)
42 - poles=pole(Dxdot)
43 - zeroes = zero(Dxdot)
44 - k = dcgain(Dxdot)
45
46 - n3=n(3,:);
47 - Dth = tf(n3,d)
48 - poles=pole(Dth)
49 - zeroes = zero(Dth)
50 - k = dcgain(Dth)
51
52 - n4=n(4,:);
53 - Dthdot = tf(n4,d)
54 - poles=pole(Dthdot)
55 - zeroes = zero(Dthdot)
56 - k = dcgain(Dthdot)
```

$Dx =$

$$\frac{s^2 + 0.3 s - 10}{s^4 + 10 s^3 + 35 s^2 + 50 s + 24}$$

Continuous-time transfer function.

poles =

-4.0000  
-3.0000  
-2.0000  
-1.0000

zeroes =

-3.3158  
3.0158

k =

-0.4167



Dxdot =

$$\frac{s^3 + 0.3 s^2 - 10 s}{s^4 + 10 s^3 + 35 s^2 + 50 s + 24}$$

Continuous-time transfer function.

poles =

-4.0000  
-3.0000  
-2.0000  
-1.0000

zeroes =

0  
-3.3158  
3.0158

k =

0

Dth =

$$\frac{s^2 + 0.3 s - 20}{s^4 + 10 s^3 + 35 s^2 + 50 s + 24}$$

Continuous-time transfer function.

poles =

-4.0000  
-3.0000  
-2.0000  
-1.0000

zeroes =

-4.6247  
4.3247

k =

-0.8333

Dthdot =

$$\frac{s^3 + 0.3 s^2 - 20 s}{s^4 + 10 s^3 + 35 s^2 + 50 s + 24}$$

Continuous-time transfer function.

poles =

-4.0000  
-3.0000  
-2.0000  
-1.0000

zeroes =

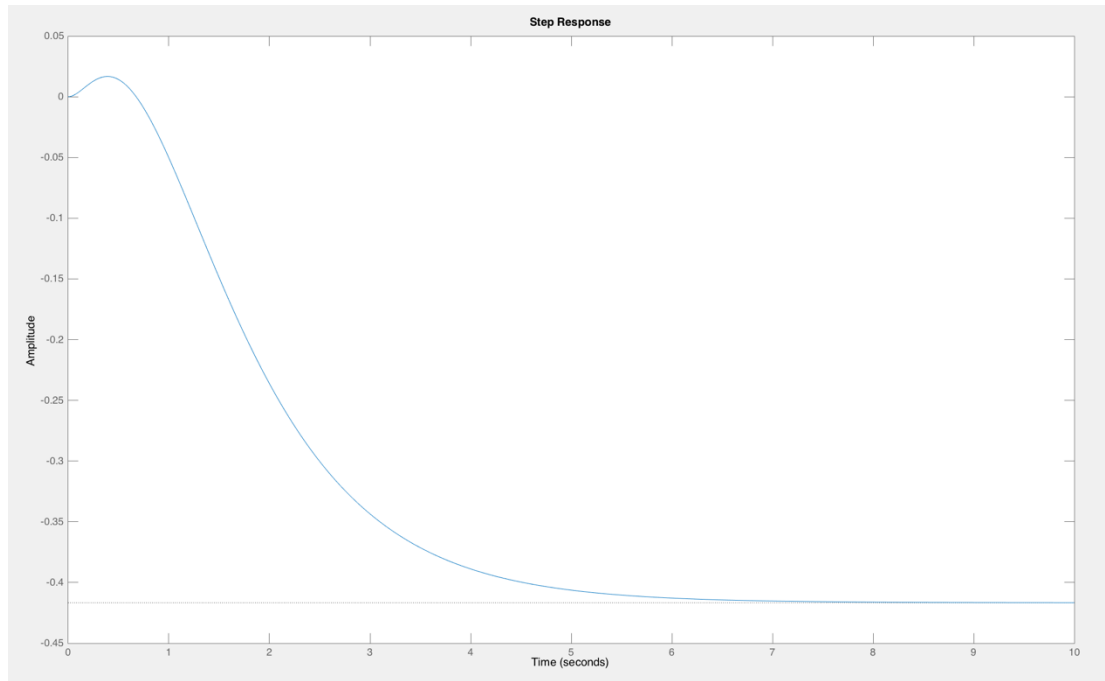
0  
-4.6247  
4.3247

k =

0

## 4.4 Χρήση prefilter για επίτευξη μηδενικού σφάλματος

Η απόκριση της Dx:



Παρατηρούμε πως έχουμε μεγάλο σφάλμα. Θα χρησιμοποιήσουμε έναν προαντισταθμιστή για να έχουμε μηδενικό.

```
33 %παίρνουμε την Dx
34 - n1=n(1,:);
35 - Dx = tf(n1,d);
36 %βρίσκουμε dcgain
37 - k = dcgain(Dx)
38 %βρίσκομε πόσο πρέπει να γίνει η ανύψωση
39 - prelifter=1/k
40 % το πολλαπλασιάζουμε με την συν. μεταφοράς
41 - Output=prelifter*Dx
42
43 - step(Output,t);
```

k =

-0.4167

prelifter =

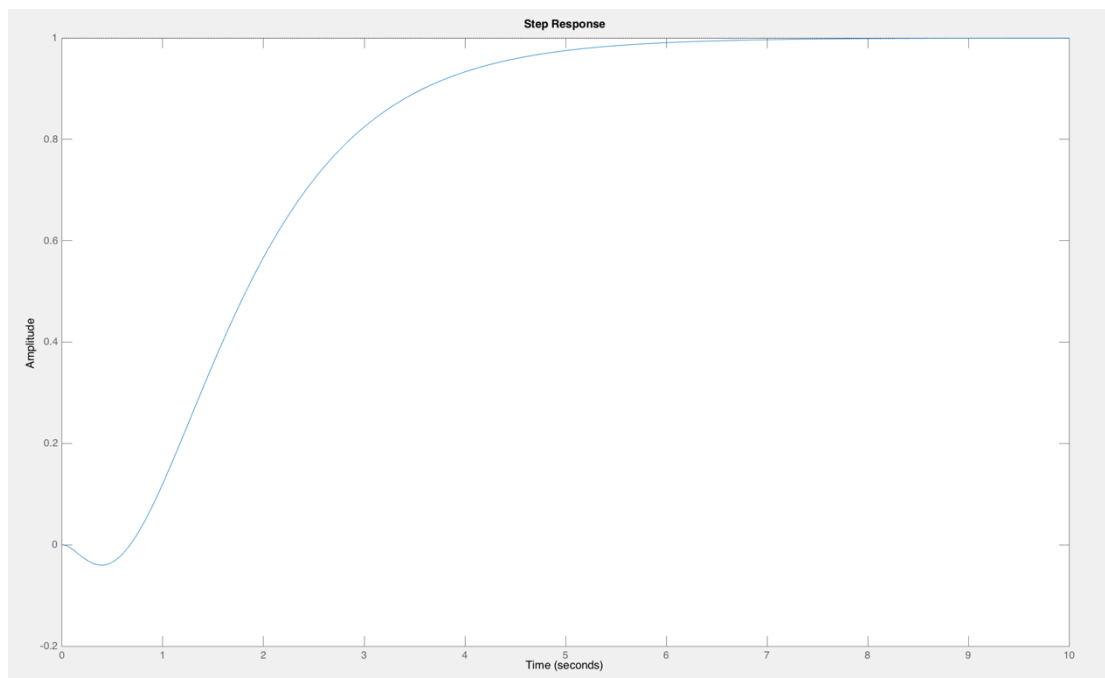
-2.4000

Output =

$$\frac{-2.4 s^2 - 0.72 s + 24}{s^4 + 10 s^3 + 35 s^2 + 50 s + 24}$$

Continuous-time transfer function.

Η καινούρια βηματική απόκριση:



Μηδενικό σφάλμα!