Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών Πάτρας
Τομέας Συστημάτων και Αυτομάτου Ελέγχου

Εργαστήριο Ψηφιακών Συστημάτων Ελέγχου
Ακαδημαϊκό Έτος 2015-2016

4η Εργαστηριακή Άσκηση

Έλεγχος (Feedback + Prefilter)

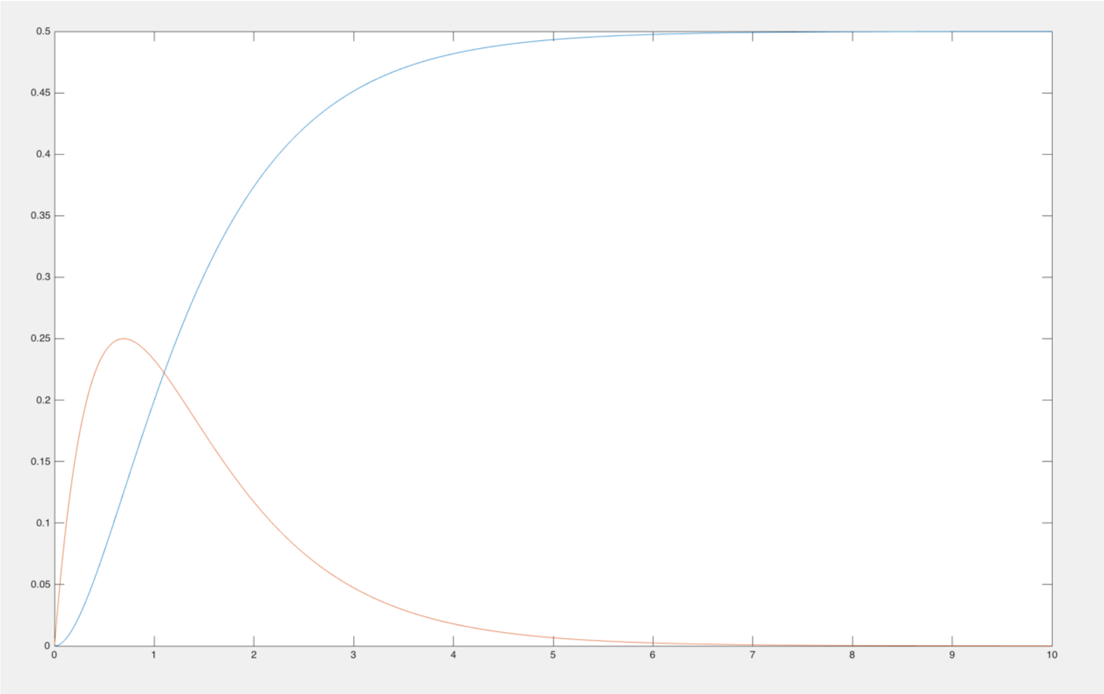
Κινούμενου Ανεστραμμένου Εκκρεμούς

ΟΝΟΜΑ ΦΟΙΤΗΤΗ	Α.Μ.
ΜΠΟΛΑΤΟΓΛΟΥ ΓΕΩΡΓΙΟΣ	228424

4.1 Ανατροφοδότηση θέσης και ταχύτητας βάσης κινούμενου εκκρεμούς

Υλοποιούμε ελεγκτή K= [K1 K2] για το Dx και Dxdot που να μεταφέρει τους πόλους του κλειστού στο -1, -2. Ο κώδικας στην matlab είναι ο εξής:

```
clc;
clear;
M=1;m=1;l=1;B_l=0.3;B_r=0.3;g=10;

A_lin=zeros(4,4);B_lin=zeros(4,1);

A_lin(1,2)=1;
A_lin(2,2)=-B_l/M;
A_lin(2,3)=m*g/M;
A_lin(2,4)=-B_r/l/M;
A_lin(3,4)=1;
A_lin(4,2)=-B_l/M/l;
A_lin(4,3)=(m*g/M/l)+g/l;
A_lin(4,4)=(-B_r/m/l^2)+(-B_r/M/l^2);

B_lin(2)=1/M;B_lin(4)=1/M/l;
C_lin=eye(4);
D_lin=zeros(4,1);
% state space form for x and \dot{x}
A_xdx=A_lin([1:2],[1:2]);,B_xdx=B_lin([1:2],:);
% u=K_1 x + K_2 \dot{x}
K_f_xdx=place(A_xdx,B_xdx,[-1;-2]);
A_xdx_cl=A_xdx-B_xdx*K_f_xdx;
t=[0:0.01:10];u=ones(size(t));
C_xdx=eye(2);D_xdx=[0;0];
xdx_cl=lsim(A_xdx_cl,B_xdx,C_xdx,D_xdx,u,t);
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) ειναι

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

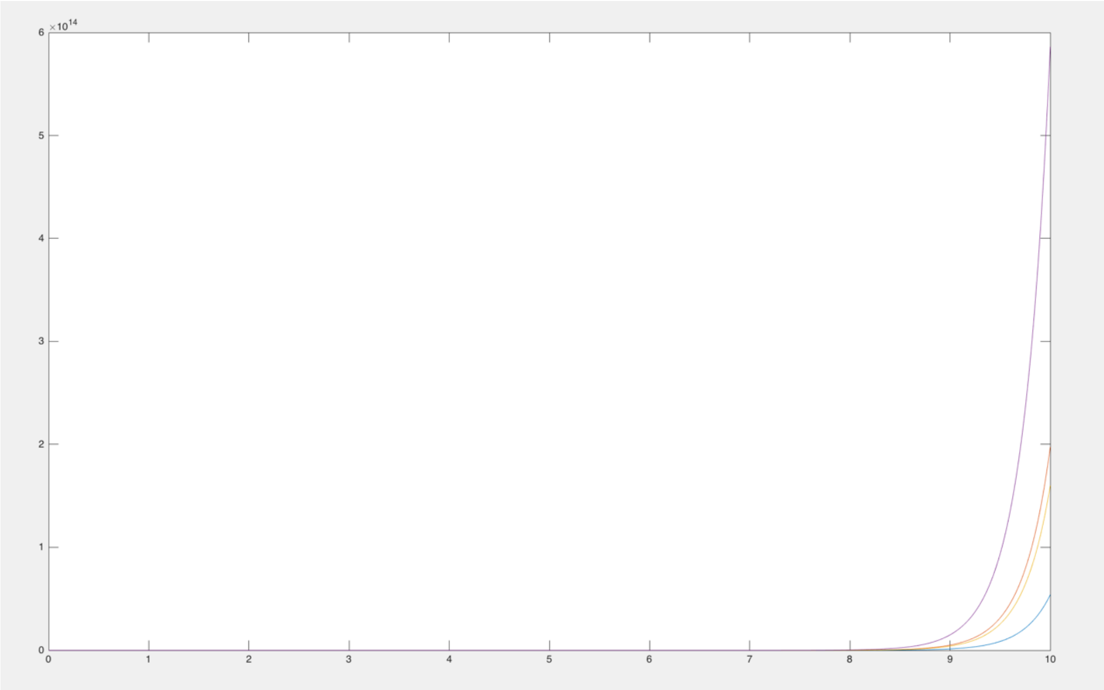
$$\Delta z = C\Delta z$$

Υλοποιούμε τον παρακάτω κώδικα στην matlab:

```
M=1;m=1;l=1;B_l=0.3;B_r=0.3;g=10;

A_lin=zeros(4,4);B_lin=zeros(4,1);

A_lin(1,2)=1;
A_lin(2,2)=-B_l/M;
A_lin(2,3)=m*g/M;
A_lin(2,4)=-B_r/l/M;
A_lin(3,4)=1;
A_lin(4,2)=-B_l/M/l;
A_lin(4,3)=(m*g/M/l)+g/l;
A_lin(4,4)=(-B_r/m/l^2)+(-B_r/M/l^2);

B_lin(2)=1/M;B_lin(4)=1/M/l;
C_lin=eye(4);
D_lin=zeros(4,1);
% state space form for x and \dot{x}
A_xdx=A_lin([1:2],[1:2]),B_xdx=B_lin([1:2],:)
% u=K_1 x + K_2 \dot{x}
K_f_xdx=place(A_xdx,B_xdx,[-1;-2]);
A_xdx_cl=A_xdx-B_xdx*K_f_xdx;
t=[0:0.01:10];u=ones(size(t));
C_xdx=eye(2);D_xdx=[0;0];
xdx_cl=lsim(A_xdx_cl,B_xdx,C_xdx,D_xdx,u,t);
%plot(t,xdx_cl);

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

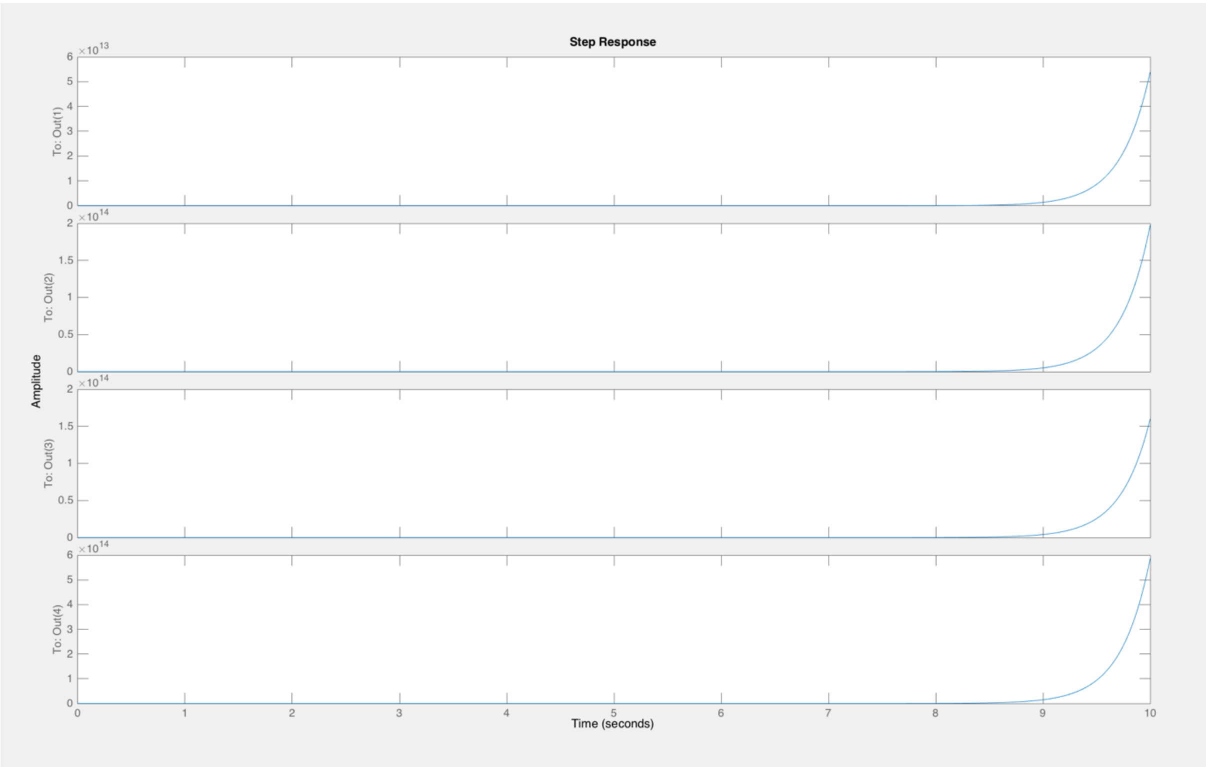
Και βρίσκουμε τον παρακάτω ελεγκτή:

K_f_xdx= [2,0 2,7 0 0]



A)Βηματική απόκριση

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



B)

Όπως βλέπουμε από την βηματική απόκριση η απόκριση μετά από κάποιο χρόνο τ 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      Time Constant
                                          (rad/seconds)     (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 Δx, Δxdot, Dth, Dthdot αντίστοιχα.

```
G = ss(A_f,B_lin,C_lin,D_lin);
[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->0 της συν. μεταφοράς.

```
    n1=n(1,:);
    Dx = tf(n1,d)
    poles=pole(Dx)
    zeroes = zero(Dx)
    k = dcgain(Dx)

    n2=n(2,:);
    Dxdot = tf(n2,d)
    poles=pole(Dxdot)
    zeroes = zero(Dxdot)
    k = dcgain(Dxdot)

    n3=n(3,:);
    Dth = tf(n3,d)
    poles=pole(Dth)
    zeroes = zero(Dth)
    k = dcgain(Dth)

    n4=n(4,:);
    Dthdot = tf(n4,d)
    poles=pole(Dthdot)
    zeroes = zero(Dthdot)
    k = dcgain(Dthdot)
```

```
Dx =

            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 =

            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 =

      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 =

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


zeroes =

   1.0e-07 *

    0.1662
   -0.1662


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^2 , δηλαδή θα είχαμε 2 μηδενικά στο 0.

```
Dthdot =

                  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= [K3 K4] για το Dx και Dxdot που να μεταφέρει τους πόλους του κλειστού στο -3, -4

```
M=1;m=1;l=1;B_l=0.3;B_r=0.3;g=10;

A_lin=zeros(4,4);B_lin=zeros(4,1);

A_lin(1,2)=1;
A_lin(2,2)=-B_r/(m*l^2)-B_r/(M*l^2);
A_lin(2,1)=(m+M)*g/M*l;
A_lin(2,3)=m*g/M;
A_lin(2,4)=-B_r/l/M;
A_lin(3,4)=1;
A_lin(4,2)=-B_l/M/l;
A_lin(4,3)=(m*g/M/l)+g/l;
A_lin(4,4)=(-B_r/m/l^2)+(-B_r/M/l^2);

B_lin(2)=1/M*l;
B_lin(4)=1/M/l;

C_lin=eye(4);
D_lin=zeros(4,1);
% state space form for x and \dot{x}
A_xdx=A_lin([1:2],[1:2]),B_xdx=B_lin([1:2],:)
% u=K_1 x + K_2 \dot{x}
K_f_xdx=place(A_xdx,B_xdx,[-3;-4]);
A_xdx_cl=A_xdx-B_xdx*K_f_xdx;
t=[0:0.01:10];u=ones(size(t));
C_xdx=eye(2);D_xdx=[0;0];
xdx_cl=lsim(A_xdx_cl,B_xdx,C_xdx,D_xdx,u,t);
plot(t,xdx_cl);
```

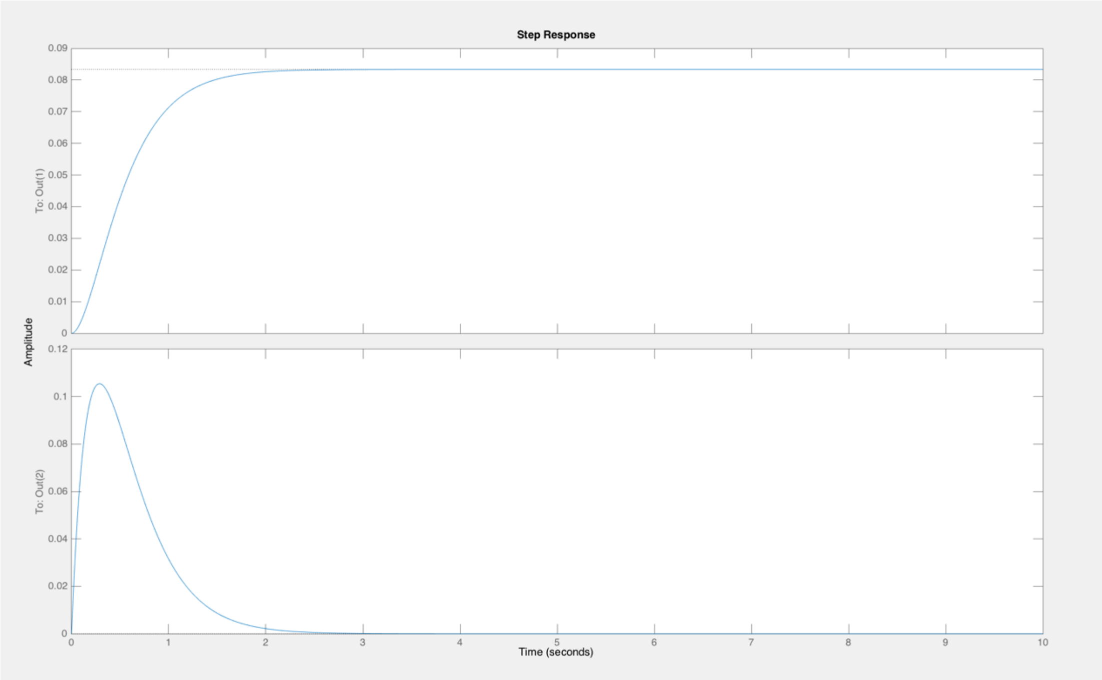
Για την βηματική απόκριση:

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

step(G,t);

Όπου t=[0:0.01:10]

Υπολογίζεται K_f_xdx= [32   6,4]



E)

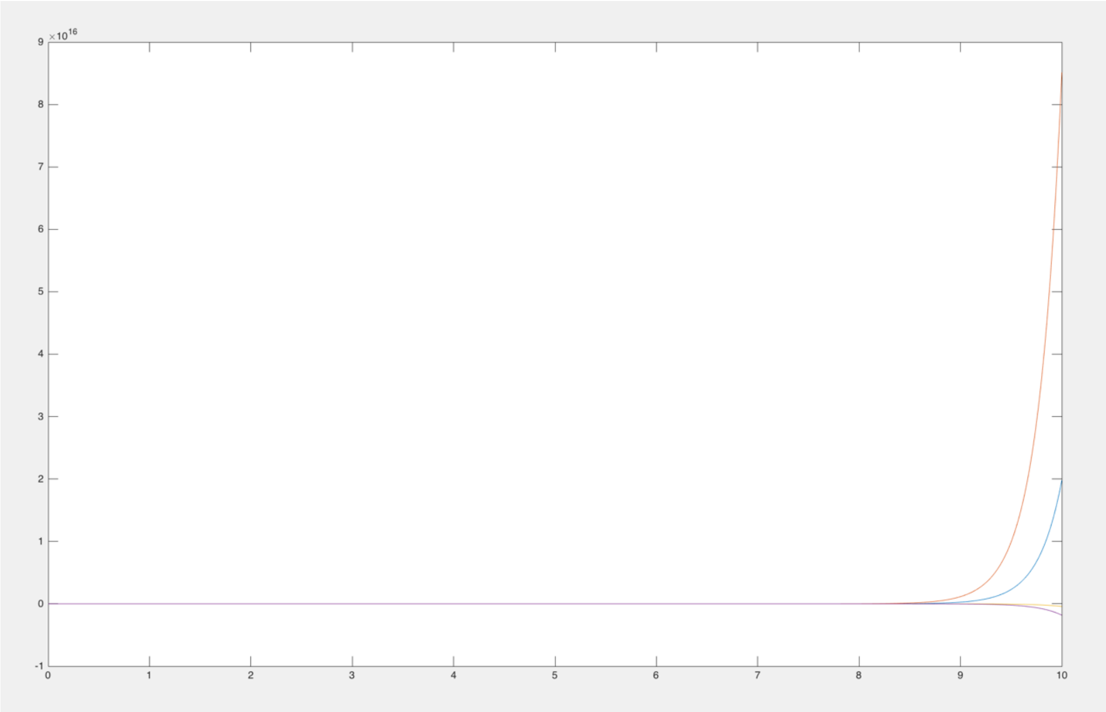
Όπως βλέπουμε από την βηματική απόκριση σταθεροποιείται σε κάποια τιμή.

Άρα το σύστημα είναι ευσταθές.

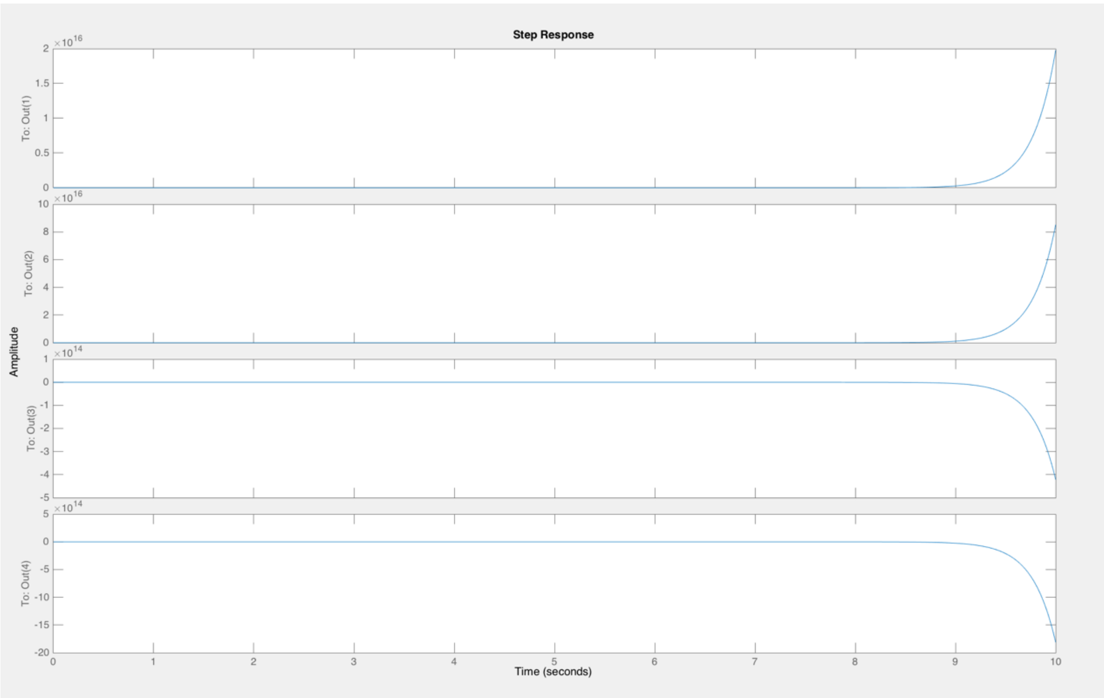
ΣΤ)

Το ζητούμενο είναι αν ένας τέτοιος ελεγκτής μπορεί να οδηγήσει σε ευστάθεια το σύστημα (5) για K=[0 0 K3 K4] .

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



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



Η)

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        Time Constant
                              (rad/seconds)        (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 εντολες:

```
[n,d]=ss2tf(A_f,B_lin,C_lin,D_lin);
n1=n(1,:);
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
```

```
Dx = tf(n1,d);
poles=pole(Dx)
zeroes = zero(Dx)
k = dcgain(Dx)
```

```
Dx =

             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
```

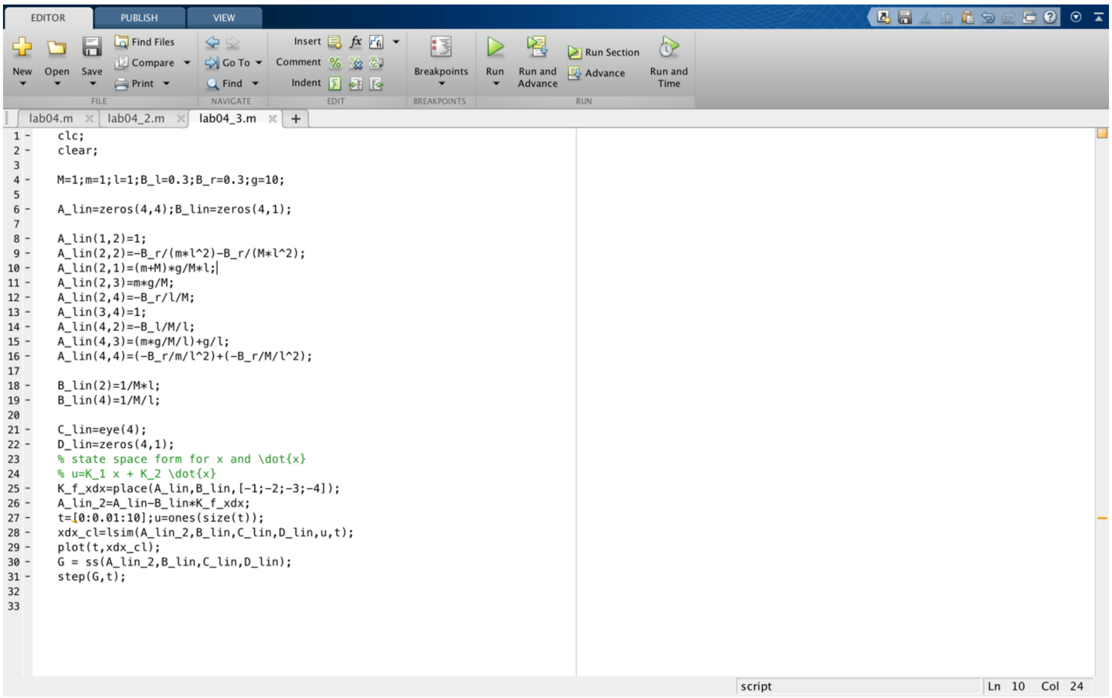
```
n2=n(2,:);
Dxdot = tf(n2,d)
poles=pole(Dxdot)
zeroes = zero(Dxdot)
k = dcgain(Dxdot)
```

```
Dxdot =

            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
```

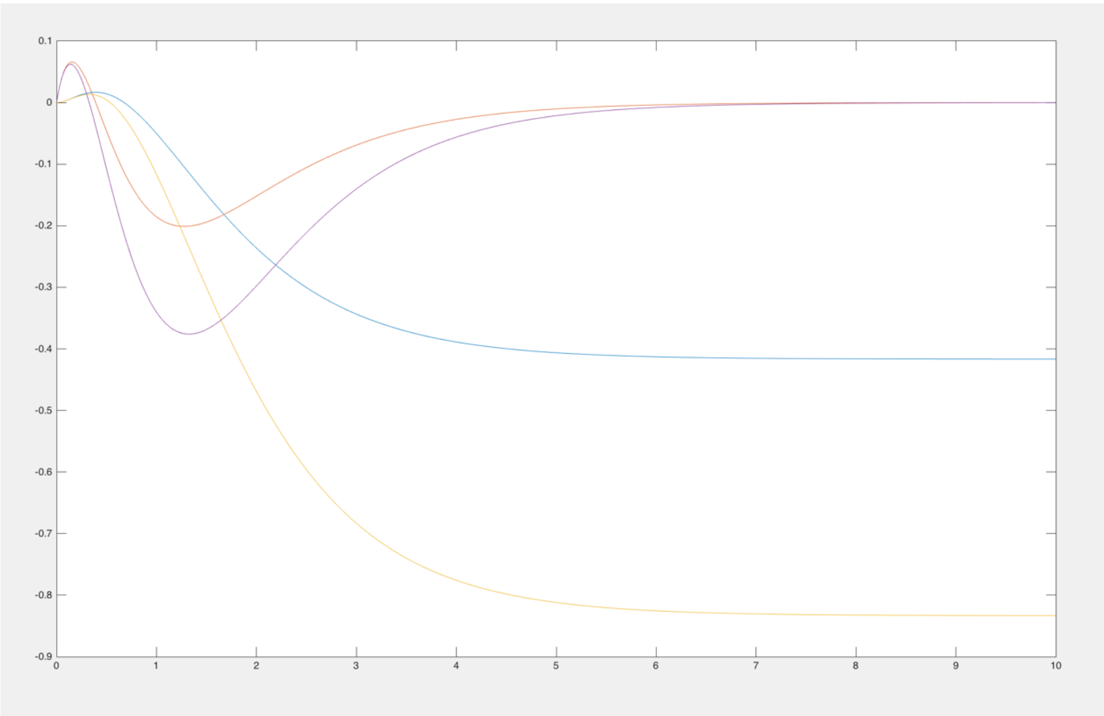
```
Dth =

              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 =

            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




```
clc;
clear;

M=1;m=1;l=1;B_l=0.3;B_r=0.3;g=10;

A_lin=zeros(4,4);B_lin=zeros(4,1);

A_lin(1,2)=1;
A_lin(2,2)=-B_r/(m*l^2)-B_r/(M*l^2);
A_lin(2,1)=(m+M)*g/M*l;
A_lin(2,3)=m*g/M;
A_lin(2,4)=-B_r/l/M;
A_lin(3,4)=1;
A_lin(4,2)=-B_l/M/l;
A_lin(4,3)=(m*g/M/l)+g/l;
A_lin(4,4)=(-B_r/m/l^2)+(-B_r/M/l^2);

B_lin(2)=1/M*l;
B_lin(4)=1/M/l;

C_lin=eye(4);
D_lin=zeros(4,1);
% state space form for x and \dot{x}
% u=K_1 x + K_2 \dot{x}
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));
xdx_cl=lsim(A_lin_2,B_lin,C_lin,D_lin,u,t);
plot(t,xdx_cl);
G = ss(A_lin_2,B_lin,C_lin,D_lin);
step(G,t);
```



I)

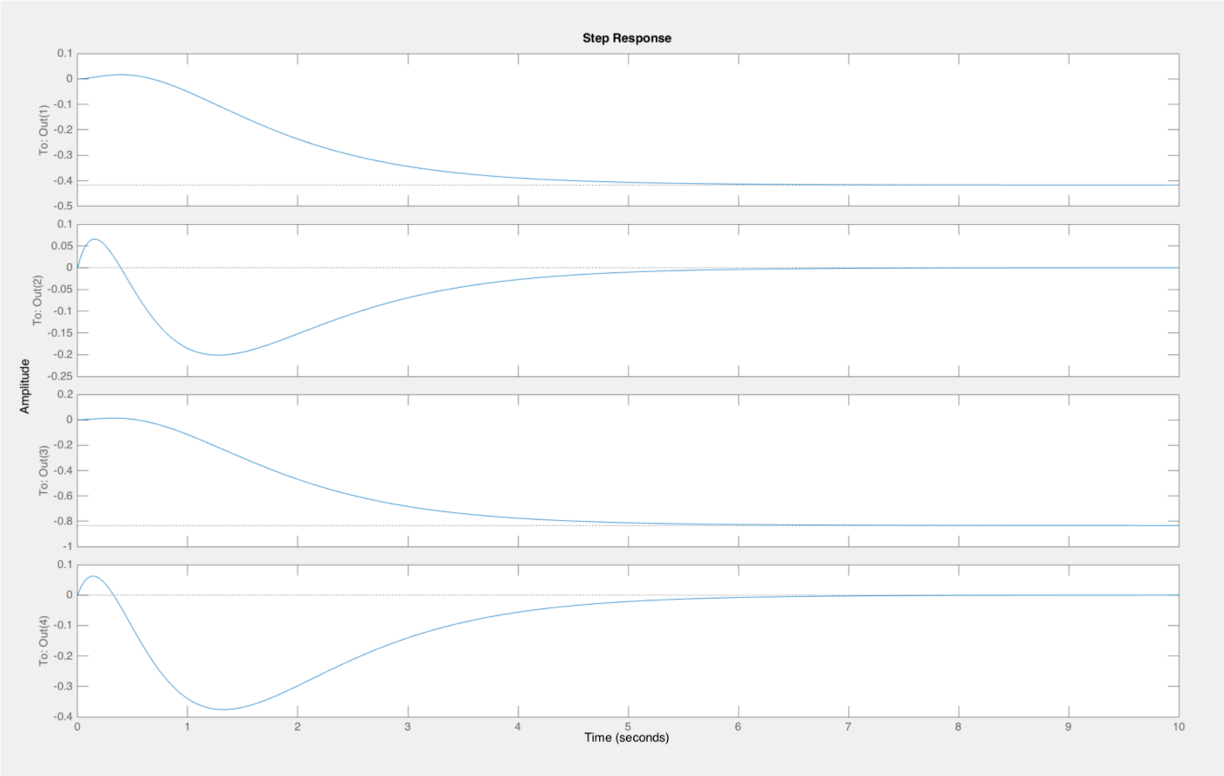
```
K_f_xdx=place(A_lin,B_lin,[-1;-2;-3;-4]);
A_lin_2=A_lin-B_lin*K_f_xdx;
t=[0:0.01:10];
u=ones(size(t));
```

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

K_f_xdx = [106,579   22,537   -34,489 -13,737]

Κ)

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



Λ)
Παρατηρούμε πως και οι 4 αποκρίσεις σταθεροποιούνται σε κάποια τιμη, 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
```

Μ)

Ομοίως όπως και στα προηγούμενα ερωτήματα έχουμε:

```
[n,d]=ss2tf(A_lin_2,B_lin,C_lin,D_lin);

n1=n(1,:);
Dx = tf(n1,d)
poles=pole(Dx)
zeroes = zero(Dx)
k = dcgain(Dx)

n2=n(2,:);
Dxdot = tf(n2,d)
poles=pole(Dxdot)
zeroes = zero(Dxdot)
k = dcgain(Dxdot)

n3=n(3,:);
Dth = tf(n3,d)
poles=pole(Dth)
zeroes = zero(Dth)
k = dcgain(Dth)

n4=n(4,:);
Dthdot = tf(n4,d)
poles=pole(Dthdot)
zeroes = zero(Dthdot)
k = dcgain(Dthdot)
```

```
Dx =

          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
```

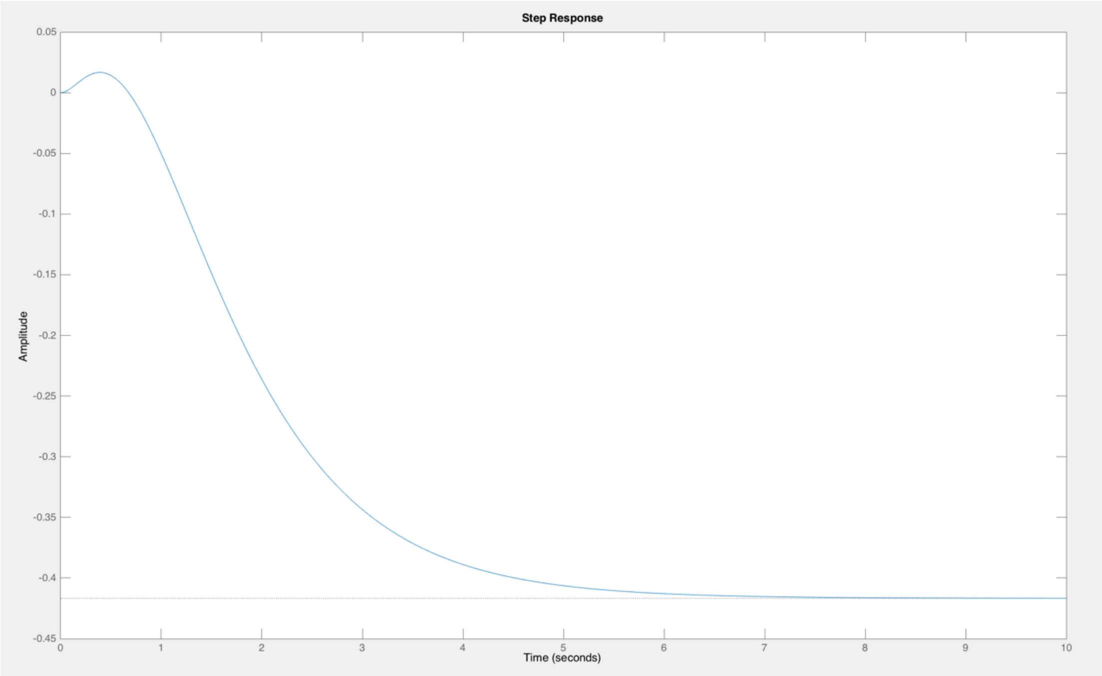
```
Dxdot =

        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 =

          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 =

        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:

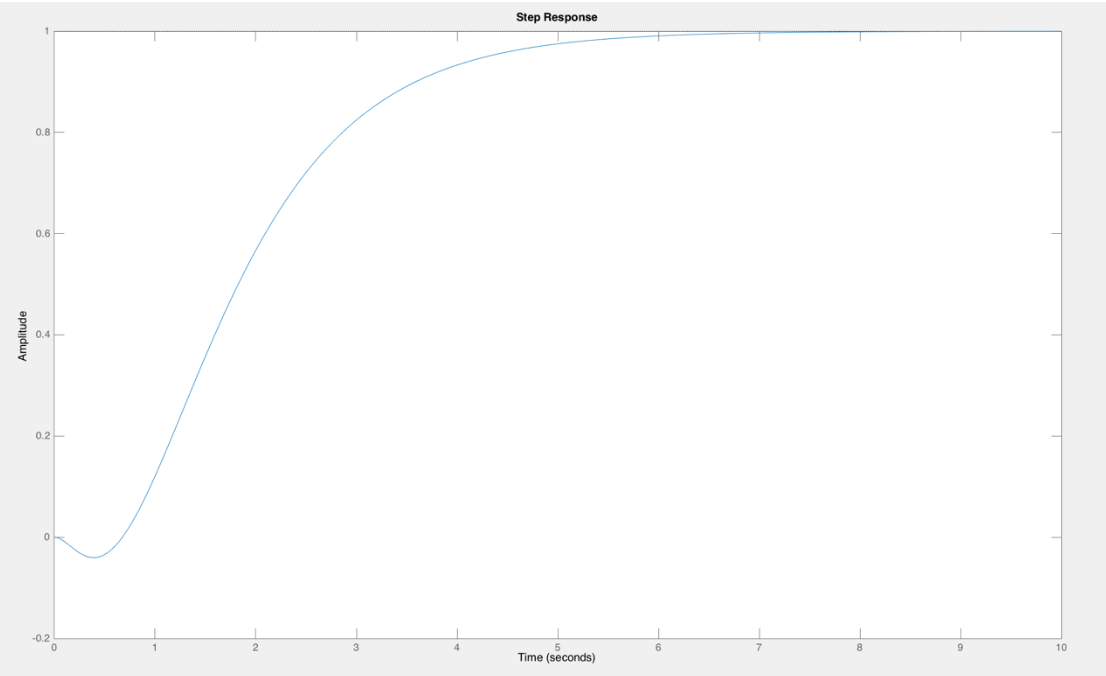


Παρατηρούμε πως έχουμε μεγάλο σφάλμα. Θα χρησιμοποιήσουμε έναν προαντισταθμιστή για να έχουμε μηδενικό.

```
%παίρνουμε την Dx
n1=n(1,:);
Dx = tf(n1,d);
%βρίσκουμε dcgain
k = dcgain(Dx)
%βρίσκομου πόσο πρέπει να γίνει η ανύψωση
prelifter=1/k
% το πολλαπλασιάζουμε με την συν. μεταφοράς
Output=prelifter*Dx

step(Output,t);
```

```
k =

   -0.4167


prelifter =

   -2.4000


Output =

         -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!