ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

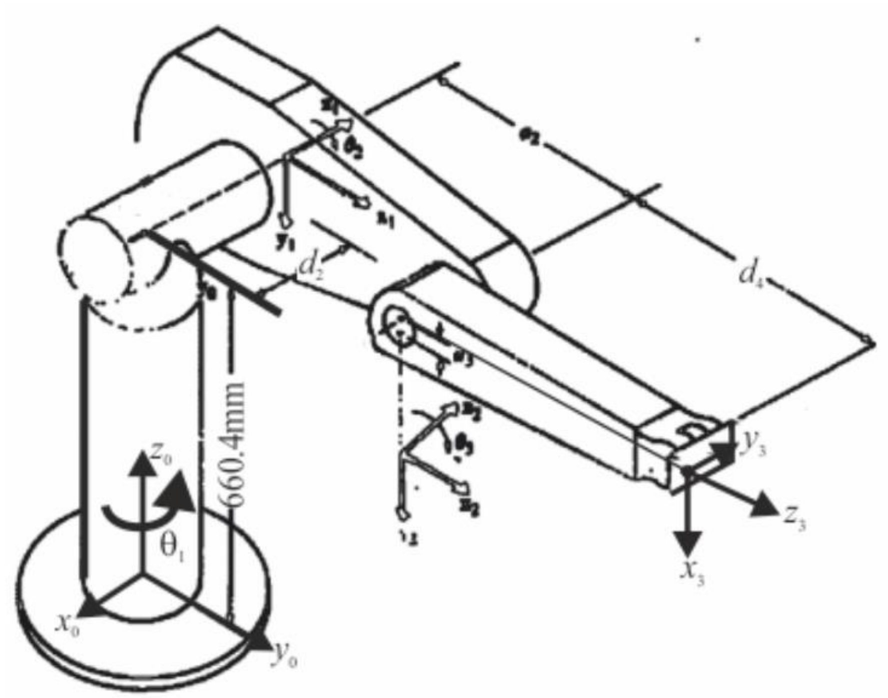
Τμήμα Ηλεκτρολόγων Μηχανικών και
Τεχνολογίας Υπολογιστών.

ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΡΟΜΠΟΤΙΚΗ
2016-17

PROJECT 2

ΓΙΩΡΓΟΣ ΜΠΟΛΑΤΟΓΛΟΥ 228424

1. Compute in symbolic form the matrix $A_0^3(\theta_1, \theta_2, \theta_3) = A_0^3(\bar{\theta})$



Αρχικά, υπολογίζονται οι D-H parameters απ' το παραπάνω σχήμα και με τα αντίστοιχα διανύσματα όπως αυτά παρουσιάζονται.

D-H parameters				
joint i	Θi*	di(in mm)	ai(in mm)	αi
1	90	d1+r	0	-90
2	0	d2+r	d2	0
3	90	0	-a3	90

Και η παράλληλη μετατόπιση απ' το Ο2 στο Ο3 κατά d4 στον z άξονα.

,όπου r=0.02m, d1=0.6604m, a2=0.4318, d2=0.14909, a3=-0.02032, d4=0.4337.

Για το εν λόγω project χρησιμοποιήθηκε το symbolic της matlab και το robotic toolbox του Peter Corke.
Μετατρέπουμε όλες τις γωνίες που μεταβάλλονται σε symbolic με την εντολή sym.

```
th1 = sym('th1');
th2 = sym('th2');
th3 = sym('th3');
```

Καθορίζουμε τις παραμέτρους που μας δίνονται.

```
r=0.020;
d1=0.6604;
a2=0.4318;
d2=0.14909;
```

```
a3=-0.02032;
d4=0.43307;
```

Στην συνέχεια χρησιμοποιώντας τις εντολές trot(x,y ή z) και transl από το robotic toolbox του peter corke δημιουργούμε τους πίνακες μετασχηματισμού διαδοχικών συστημάτων αξόνων A_{i-1}^{i} και εν τέλη τον ζητούμενο ολικό πίνακα μετασχηματισμού A_0^6.

T = trotz(**θ**) θα δώσει τον ομογενή μετασχηματισμό(πίνακα 4x4) που εκφράζει την περιστροφή κατά theta ακτίνια γύρω από τον z άξονα.

```
T=
[ cos(θ),   -sin(θ),  0, 0]
[ sin(θ),    cos(θ),  0, 0]
[        0,          0, 1, 0]
[        0,          0, 0, 1]
```

T = trotx(**θ**) θα δώσει τον αντίστοιχο μετασχηματισμό περιστροφής γύρω από τον x άξονα.

```
T =
[ 1,         0,          0, 0]
[ 0, cos(θ),   -sin(θ), 0]
[ 0,  sin(θ),   cos(θ), 0]
[ 0,         0,          0, 1]
```

T = transl(**x**, **y**, **z**) θα δώσει τον ομογενή μετασχηματισμό(πίνακα 4x4) που εκφράζει την μετατόπιση κατά x,y και z.

```
T =
[ 1, 0, 0, x]
[ 0, 1, 0, y]
[ 0, 0, 1, z]
[ 0, 0, 0, 1]
```

Άρα, όλοι οι πίνακες μετασχηματισμού διαδοχικών συστημάτων αξόνων μπορουν να περιγραφούν από την έκφραση:

trotz(θ1)*transl(x1,y1,z1)*transl(x2,y2,z2)*trotx(θ2)

Καλείται η συνάρτηση DH_parameters που επιστρέφει του ομογενείς πίνακες A_0^1 A_0^2 και A_0^3.

```
function [A01,A02,A03] = DH_parameters(th)

r=0.020;d1=0.6604;a2=0.4318;d2=0.14909;a3=-0.02032;d4=0.43307;

A01 = trotz(th(1))*transl(0,0,d1+r)*transl(0,0,0)*trotx(-pi/2);
A12 = trotz(th(2))*transl(0,0,d2+r)*transl(a2,0,0)*trotx(0);
A23 = trotz(th(3))*transl(0,0,0)*transl(a3,0,0)*trotx(pi/2);
A23=[1 0 0 d4;0 1 0 0;0 0 1 0;0 0 0 1]*A23;

A02=simplify(vpa(A01*A12));
A03=simplify(vpa(A01*A12*A23));

end
```

A_0^3=
[cos(th2 + th3)*cos(th1), -1.0*sin(th1), sin(th2 + th3)*cos(th1), 0.865*cos(th1)*cos(th2) - 0.169*sin(th1) + 0.0203*cos(th1)*sin(th2)*sin(th3) - 0.0203*cos(th1)*cos(th2)*cos(th3)]
[cos(th2 + th3)*sin(th1), cos(th1), sin(th2 + th3)*sin(th1), 0.169*cos(th1) + 0.865*cos(th2)*sin(th1) + 0.0203*sin(th1)*sin(th2)*sin(th3) - 0.0203*cos(th2)*cos(th3)*sin(th1)]
[-1.0*sin(th2 + th3), 0, cos(th2 + th3), 0.0203*sin(th2 + th3) - 0.865*sin(th2) + 0.68]
[0, 0, 0, 1.0]

2. Compute the 4×4 inertia matrices J_i of each link $i=1,2,3$

Οι inertia matrices είναι της μορφής:

$$\mathbf{J}_i = \int {}^i\mathbf{r}_i\, {}^i\mathbf{r}_i^T dm = \begin{bmatrix} \int x_i^2\, dm & \int x_i y_i\, dm & \int x_i z_i\, dm & \int x_i\, dm \\ \int x_i y_i\, dm & \int y_i^2\, dm & \int y_i z_i\, dm & \int y_i\, dm \\ \int x_i z_i\, dm & \int y_i z_i\, dm & \int z_i^2\, dm & \int z_i\, dm \\ \int x_i\, dm & \int y_i\, dm & \int z_i\, dm & \int dm \end{bmatrix}$$

,όπου $dm = \iiint p dx dy dz$

Αρκεί λοιπόν, να προσδιορίσουμε τα όρια κάθε συνδέσμου τα οποία αντιπροσωπεύουν τις αποστάσεις σε κάθε άξονα από το αμέσως επόμενο σημείο συντεταγμένων(Oi i=1,2,3):

Link 1:
Z: -(r.^2-x.^2).^(1/2),(r.^2-x.^2).^(1/2)
X:-r,r
Y:r,r+d1

Link 2:
Ο σύνδεσμος 2 αποτελείται από το άθροισμα ενός μικρού κυλίνδρου με μήκος d2 και ενός μεγάλου κυλίνδρου με μήκος α2+80mm. Ο inertia matrix αποτελείται από το άθροισμα των 2 επιμέρους 4x4 inertia matrix.

Για τον μικρό κύλινδρο έχουμε:
Z:-r-d2, -r
X:-a2-(r.^2-y.^2).^(1/2), -a2+(r.^2-y.^2).^(1/2)
Y:-r, r

Για τον μεγάλο κύλινδρο έχουμε:
Z:-r, r
X:-a2-0.08, 0
Y:-(r.^2-z.^2).^(1/2), (r.^2-z.^2).^(1/2))

Link 3:
Z:-d4 , 0
X:-(r.^2-y.^2).^(1/2), (r.^2-y.^2).^(1/2)
Y:-r, r

Η συνάρτηση που υλοποιείται:

Παίρνει σαν όρισμα τα άκρα των ολοκληρωμάτων και επιστρέφει τον 4x4 inertia matrix.

```
function [J] = inertial(z1,z2,x1,x2,y1,y2)
y=sym('y');z=sym('z');x=sym('x');r=sym('r');d2=sym('l');m=sym('m');
th=sym('th');R=sym('R');
r=0.020; %aktina
d1=0.6604;d2=0.14909; %d se metra
p=10^4;

xyz=[x y z 1];
for i=1:4
    for j=i:4
        fun=(p*xyz(i)*xyz(j));
        mesa=int(fun,z,z1,z2);
        meseo=int(mesa,x,x1,x2);
        ekso=int(meseo,y,y1,y2);
        J(i,j)=ekso ;
        if i~=j
            J(j,i)=J(i,j); %summetrikos
        end
    end
end

end
```

Και έχουμε τους 3 inertia matrixes:

```
J(:,:,1) =

[ 0.00082988,      0,          0,      0]
[          0, 1.3194,          0, 2.9063]
[          0,      0, 0.00082988,      0]
[          0, 2.9063,          0, 8.2988]


J(:,:,2) =

[  0.91106,         0, 0.076486,  -2.4548]
[        0, 0.0008305,        0,        0]
[ 0.076486,         0,  0.02086, -0.17713]
[  -2.4548,         0, -0.17713,    8.305]


J(:,:,3) =

[ 0.00054421,          0,        0,       0]
[          0, 0.00054421,        0,       0]
[          0,          0, 0.34022, -1.1784]
[          0,          0, -1.1784,  5.4421]
```

3. Compute in symbolic form the dynamics of the 3DoF manipulator

$$D_{3\times3}\left(\overline{\theta}\right)\ddot{\overline{\theta}} + C_{3\times1}\left(\overline{\theta},\dot{\overline{\theta}}\right) + G_{3\times1}\left(\overline{\theta}\right) = \overline{\tau}_{3\times1}$$

Για τον υπολογισμού του $\mathbf{D}_{\mathbf{3X3}}$ όρου:

$$D_{ik} = \sum_{j=\max(i,k)}^{n} \mathrm{Tr}\ (\mathbf{U}_{jk}\,\mathbf{J}_j\,\mathbf{U}_{ji}^T)$$

, όπου n=3

, J_i οι inertia matrixes που υπολογίσαμε παραπάνω και

$$U_{ij} = \frac{\partial A_0^{\,i}}{\partial \theta_j}$$

Αρχικά, υλοποιούμε την συνάρτηση που υπολογίζει την παράγωγο κάθε $A_0^{\,i}$ ως προς θ και την αποθηκεύει σε έναν 4 διαστάσεων πίνακα, όπως φαίνεται παρακάτω.

$\bar{\theta}=[\theta_1\ \theta_2\ \theta_3]$

```
function U = derivatives(th,A)

for i=1:3
    for j=1:i
        U(:,:,i,j)=simplify(diff(A(:,:,i),th(j)));
        if i~=j
            U(:,:,j,i)=U(:,:,i,j); %summetrikos

        end
    end
end


end
```

Στην συνέχεια υπολογίζουμε τον D, χρησιμοποιώντας την παραπάνω μεθοδολογία.

```
D=sym(zeros(3));
for i=1:3
    for k=1:3
        for j=max(i,k):3
            D(i,k)=D(i,k)+trace(U(:,:,j,k)*J(:,:,j)*transpose(U(:,:,j,i)));
        end
        if i~=j
            D(k,i)=D(i,k); %summetrikos
        end
    end
end
```

Για τον υπολογισμό του πίνακα $\mathbf{C}_{\mathbf{3X1}}$:

C = h = coriolis + centrifugal force

$$h_i = \sum_{k=1}^{n} \sum_{m=1}^{n} h_{ikm} \dot{q}_k \dot{q}_m \quad , \qquad h_{ikm} = \sum_{j=\max(i,k,m)}^{n} \mathbf{Tr}\ (\mathbf{U}_{jkm}\, \mathbf{J}_j\, \mathbf{U}_{ji}^T)$$

,όπου n=3 , $\dot{q} = \dot{\bar{\theta}} = [\ \dot{\theta}_1\ \ \dot{\theta}_2\ \ \dot{\theta}_3\ ]$

Υπολογίζουμε τον C, χρησιμοποιώντας την παραπάνω μεθοδολογία.

```
h=sym([0 0 0]);
for i=1:3
    for k=1:3
        for m=1:3
            h1=sym(0);
            j=sort([i k m]);
            for j=j(1):3
                dU=diff(U(:,:,j,k),th(m));
                h1=h1+trace(dU*J(:,:,j)*transpose(U(:,:,j,i)));
            end
            h(i)=h(i)+h1*thdot(k)*thdot(m);


        end
    end
end
h=h.';
```

Για τον υπολογισμό του πίνακα $\mathbf{G_{3X1}}$:

G = c = gravity loading force

$$c_i = \sum_{j=i}^{n} \left(-m_j \mathbf{g} \mathbf{U}_{ji} \, {}^j\bar{\mathbf{r}}_j \right)$$

Το g αποτελεί το διάνυσμα βαρύτητας και έχει την μορφή $\bar{g}$ = [0 0 -9.81 0]

Το (4,4) στοιχείο των inertia matrix είναι η μάζα του κάθε συνδέσμου (m_j).

Η 4η στήλη(και γραμμή αφού είναι συμμετρικός) διαιρεμένη με την μάζα του κάθε συνδέσμου είναι το κέντρο βάρους του ($\tilde{r}_j$).

Υπολογίζουμε τον G, χρησιμοποιώντας την παραπάνω μεθοδολογία.

```
g=sym([ 0 0 -9.81 0]); %dianusma varhthtas
m=sym([J(4,4,1) J(4,4,2) J(4,4,3)]); %dianusma mazas
rp=sym([J(:,4,1)/m(1) J(:,4,2)/m(2) J(:,4,3)/m(3)]);
c=sym([0; 0; 0]);

for i=1:3
    for j=i:3
    c(i)= c(i)-m(j)*g*U(:,:,i,j)*rp(:,j);
    end
end


end
```

Τέλος, υπολογίζουμε την δυναμικη:

$$D_{3\times3}\left(\overline{\theta}\right)\ddot{\overline{\theta}} + C_{3\times1}\left(\overline{\theta},\dot{\overline{\theta}}\right) + G_{3\times1}\left(\overline{\theta}\right) = \overline{\tau}_{3\times1}$$

,όπου $\ddot{\overline{\theta}}$ =[$\ddot{\theta}_1$ $\ddot{\theta}_2$ $\ddot{\theta}_3$]

```
%% 3) COMPUTES IN SYMBOLIC FORM THE DYNAMICS

U=derivatives(th,A);

[D C G]=compute(J,U,th,thdot);

torque=D*thdotdot.'+G+C;
torque=vpa(torque,4)
```

Στην συνάρτηση compute τρέχουν οι παραπάνω αλγόριθμοι και επιστρέφουν τους αντίστοιχους πίνακες. Στην συνέχεια υπολογίζεται η ροπή από τον παραπάνω τύπο.

4. Plot $\tau(t), t \in [0,12]$ for joint trajectories (expressed in degrees)

$$\bar{\theta}^d(t) = \begin{bmatrix} A_1 \sin\left(\frac{2\pi}{T_1}t\right) \\ A_2 \sin\left(\frac{2\pi}{T_2}t\right) \\ 40^\circ + A_3 \sin\left(\frac{2\pi}{T_3}t\right) \end{bmatrix}$$

,where $A_1 = 2$ $A_2 = 2$ $A_3 = 80$ and $T_1 1 = 2$ $T_2 = 3$ $T_3 = 6$ seconds.

Record

$$\bar{\tau}_{\max} = \begin{bmatrix} \tau_1^{\max} \\ \tau_2^{\max} \\ \tau_3^{\max} \end{bmatrix} = \begin{bmatrix} \max_t \tau_1(t) \\ \max_t \tau_2(t) \\ \max_t \tau_3(t) \end{bmatrix}$$

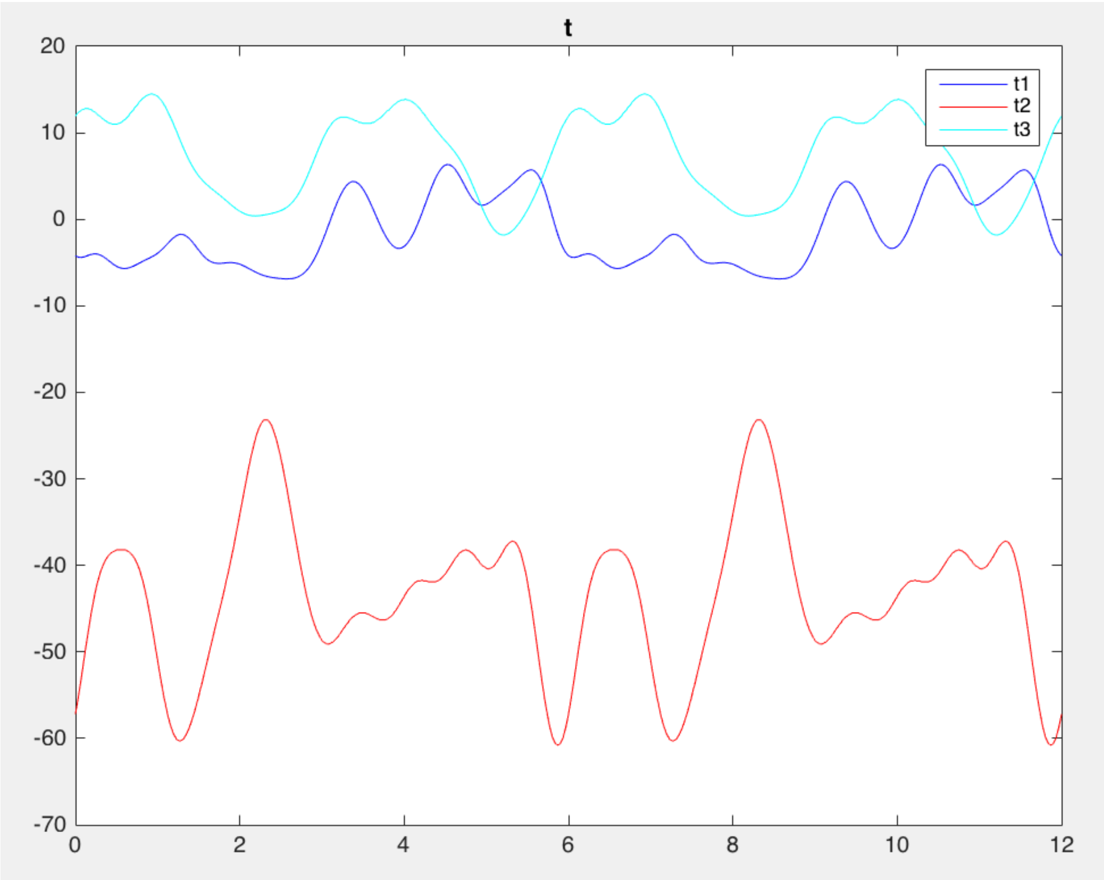
Στο αρχικό πρόγραμμα έχουμε τον εξής κώδικα:

```
%% 4) PLOT TORQUES

t=0:0.1:12; %dianusma xronou

A1=80*pi/180;A2=40*pi/180;A3=40*pi/180; %metatroph se rad
T1=6;T2=3;T3=2;

[tdesired thdesired thdotdesired thdotdotdesired] = er4(th,thdot,thdotdot,torque,t);

figure , plot(t, tdesired(1,:), '-b')
hold on, plot(t, tdesired(2,:), '-r')
hold on, plot(t, tdesired(3,:), '-c')
legend('t1', 't2', 't3')
title('t')
```

που καλεί την συνάρτηση:

```matlab
function [tdesired,thdesired,thdotdesired,thdotdotdesired] = er4(th,thdot,thdotdot,torque,t)

th1 = sym('th1');th2 = sym('th2');th3 = sym('th3');pi=sym('pi');
th1dot = sym('th1dot');th2dot = sym('th2dot');th3dot = sym('th3dot');
th1dotdot = sym('th1dotdot');th2dotdot = sym('th2dotdot');th3dotdot = sym('th3dotdot');
A1=80*pi/180;A2=40*pi/180;A3=40*pi/180;
T1=6;T2=3;T3=2;

thdesired=subs(th',{th1, th2, th3},{A1*sin(2*pi*t/T1) A2*sin(2*pi*t/T2) A2+A3*sin(2*pi*t/T3)});
thdotdesired=subs(thdot',{th1dot, th2dot, th3dot},{ (4*pi^2*cos((pi*t)/3))/27, ...
    (4*pi^2*cos((2*pi*t)/3))/27, (2*pi^2*cos(pi*t))/9});
thdotdotdesired=subs(thdotdot',{th1dotdot, th2dotdot, th3dotdot},...
{ -(4*pi^3*sin((pi*t)/3))/81, -(8*pi^3*sin((2*pi*t)/3))/81, -(2*pi^3*sin(pi*t))/9});

tdesired=vpa(subs(torque,{th1 th2,th3,th1dot,th2dot,th3dot,th1dotdot,th2dotdot,th3dotdot}...
,{thdesired(1,:) thdesired(2,:) thdesired(3,:) thdotdesired(1,:) thdotdesired(2,:)...
 thdotdesired(3,:) thdotdotdesired(1,:) thdotdotdesired(2,:) thdotdotdesired(3,:)}),3);



end
```

Για Ts=0.01s έχουμε:



Για τον υπολογισμό των μεγίστων:

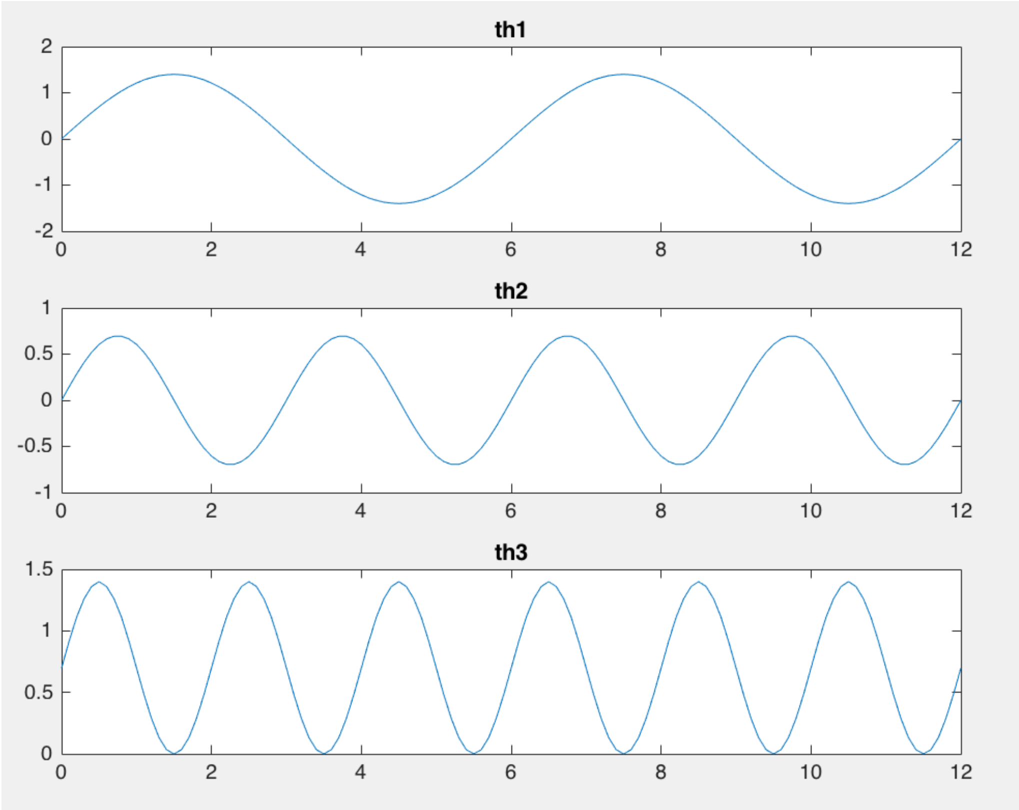
```
%% 4) DISPLAY MAX TORQUES

tsorted1=sort(tdesired(1,:),'descend');
tsorted2=sort(tdesired(2,:),'ascend');
tsorted3=sort(tdesired(3,:),'descend');
max_torque=[abs(tsorted1(1)) abs(tsorted2(1)) abs(tsorted3(1))]
```

$$\tau_{\max} = \begin{matrix} [6.34] \\ [60.8] \\ [14.5] \end{matrix}$$

5. Design a computed torque the robot forces its joints to track these desired angles for $\bar{\theta}(0) = \begin{bmatrix} -160^\circ \\ -225^\circ \\ -45^\circ \end{bmatrix}$.

Assume that K_d, K_p are diagonal positive matrices. Select their diagonal elements so that you have a satisfactory response, while the maximum torque applied to the motors does not exceed $2\tau_{\max}$

Τα θd απ' το προηγούμενο ερώτημα είναι τα εξης:



Νόμος ελέγχου που καλούμαστε να υλοποίησουμε:

$$\bar{\tau}_{3\times1} = D_{3\times3}\left(\bar{\theta}\right)\left[\ddot{\bar{\theta}}^{d} + K_{d}\left(\dot{\bar{\theta}}^{d} - \dot{\bar{\theta}}\right) + K_{p}\left(\bar{\theta}^{d} - \bar{\theta}\right)\right] + C_{3\times1}\left(\bar{\theta},\dot{\bar{\theta}}\right) + G_{3\times1}\left(\bar{\theta}\right)$$

Χρησιμοποιώντας την odefun θα προσδιορίσουμε τα $\dot{\bar{\theta}}$ και $\bar{\theta}$ για ένα μικρό step στον χρόνο λύνοντας την γραμμική πλέον διαφορική εξίσωση η οποία μας δίνει το σφάλμα $\bar{\varepsilon}$=$\bar{\theta}d - \bar{\theta}$ και $\dot{\bar{\varepsilon}}$=$\dot{\bar{\theta}}d - \dot{\bar{\theta}}$

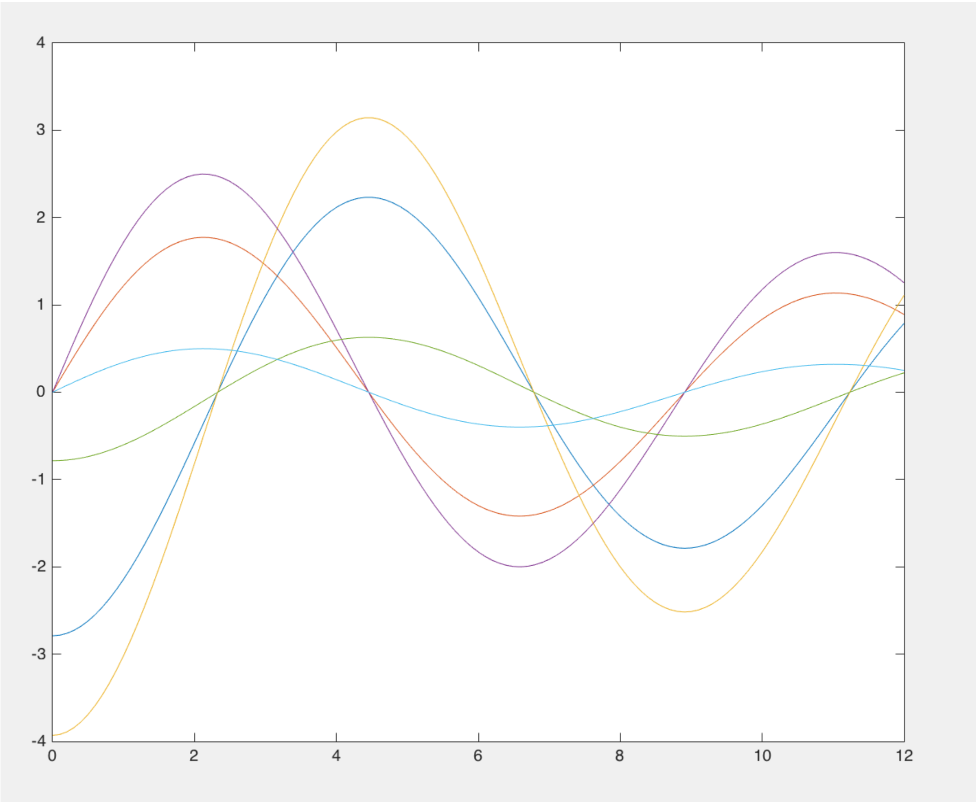
Η διαφορική λοιπόν είναι $\ddot{\bar{\varepsilon}} + Kd\,\dot{\bar{\varepsilon}}$+Kp $\bar{\varepsilon}$

Σαν αρχικές τιμές έχω για την θέση θ(0) και για ταχύτητα = 0

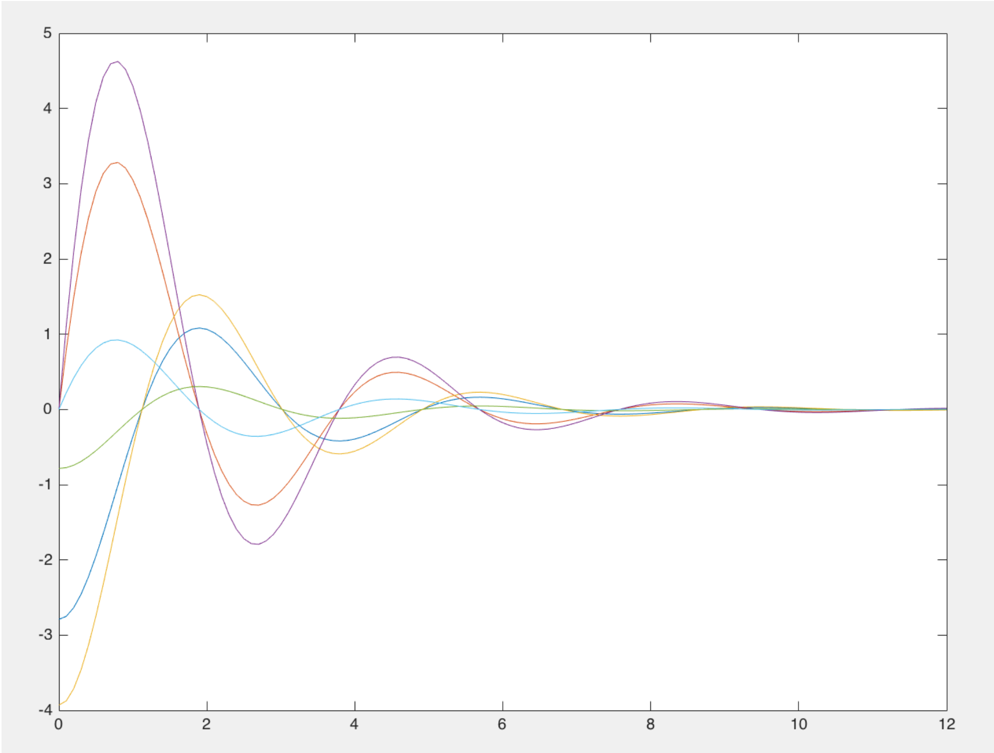
Για διάφορα Kp και Kd με όλα τα στοιχεία τους ίδια:

Διαγράμματα για $\bar{\theta}$=[θ_1 θ_2 θ_3] και $\dot{\bar{\theta}}$ =[$\dot{\theta}_1$ $\dot{\theta}_2$ $\dot{\theta}_3$]

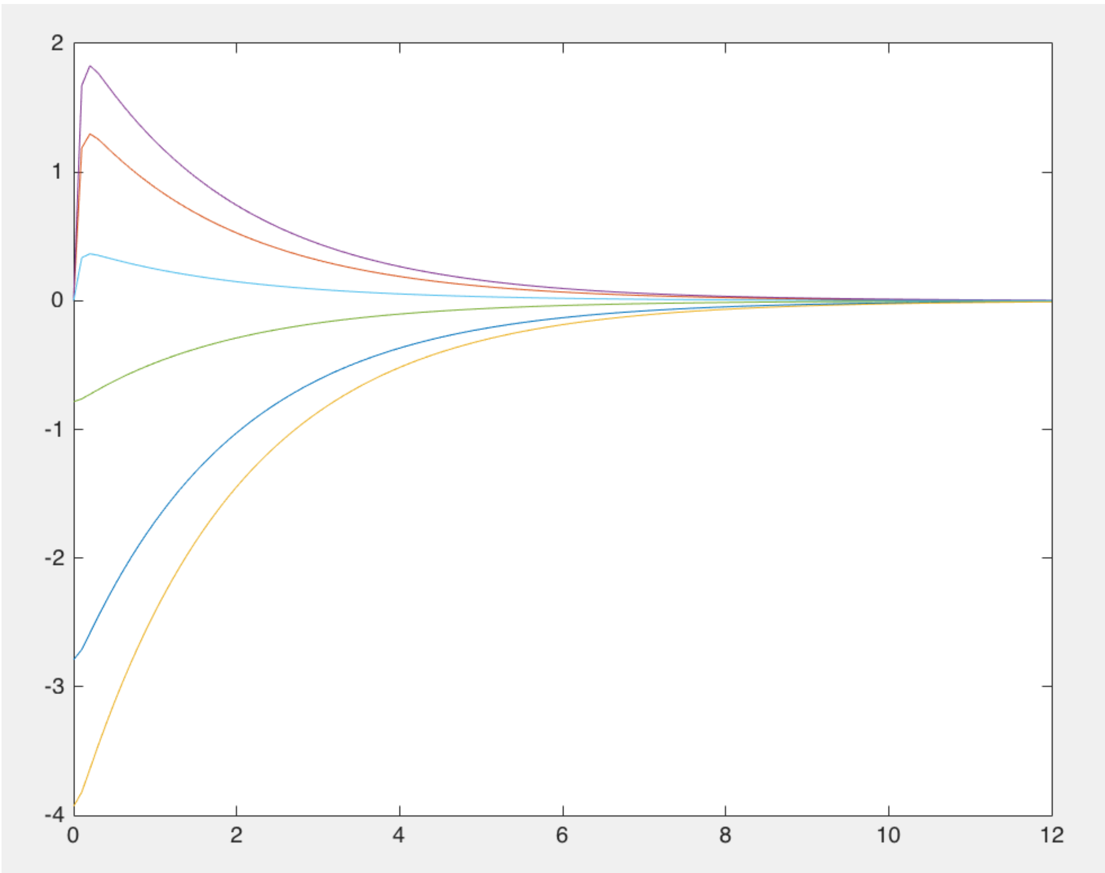
Για Kp=0.5, Kd=0.1:



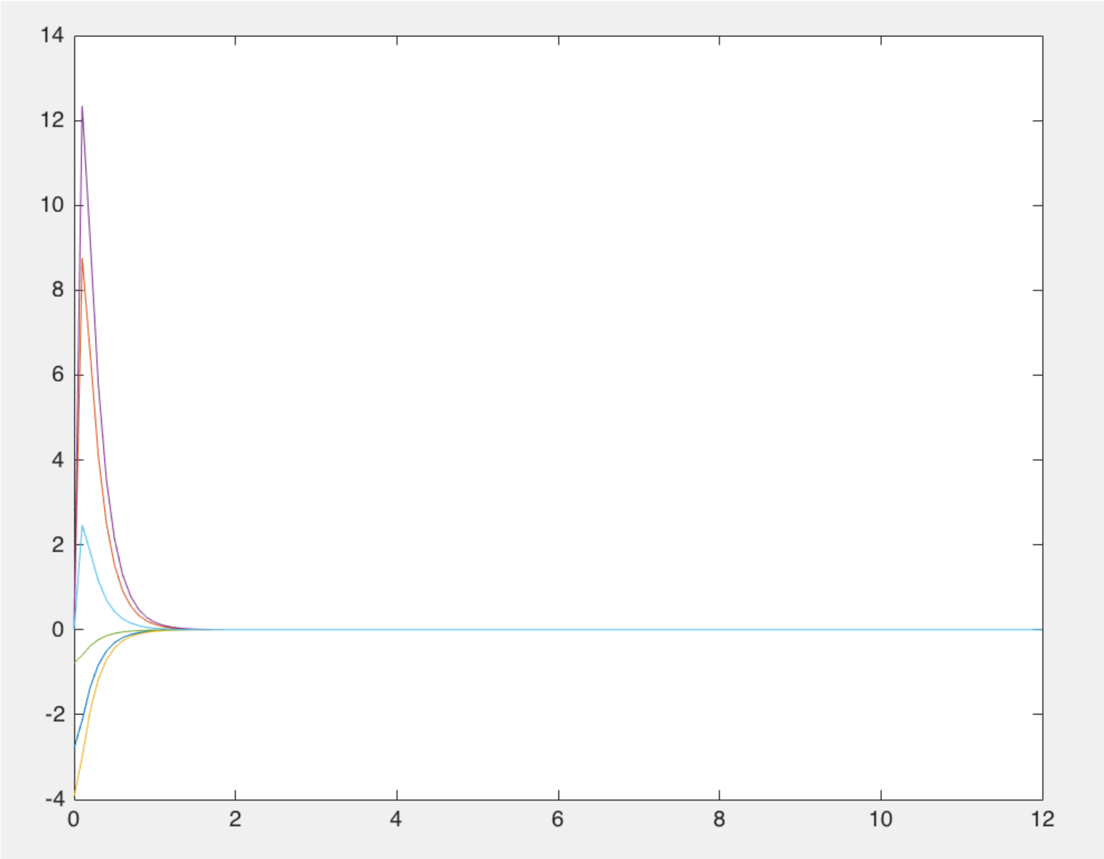
Για Kp=3, Kd=1:



Για Kp=10, Kd=20:



Για Kp=100, Kd=25:



Παρατηρούμε πως για μεγαλύτερα Kp και Kd μείωνεται πολύ πιο γρήγορα το σφάλμα με το πέρας του χρόνου. Άρα για μια βέλτιστη λύση θα μπορούσαμε να επιλέξουμε τεράστια K, όπως τότε θα έπρεπε να δώσουμε και πάρα πολύ μεγάλη ροπή στο σύστημα. Εδώ έχουμε περιορισμό 2τmax!

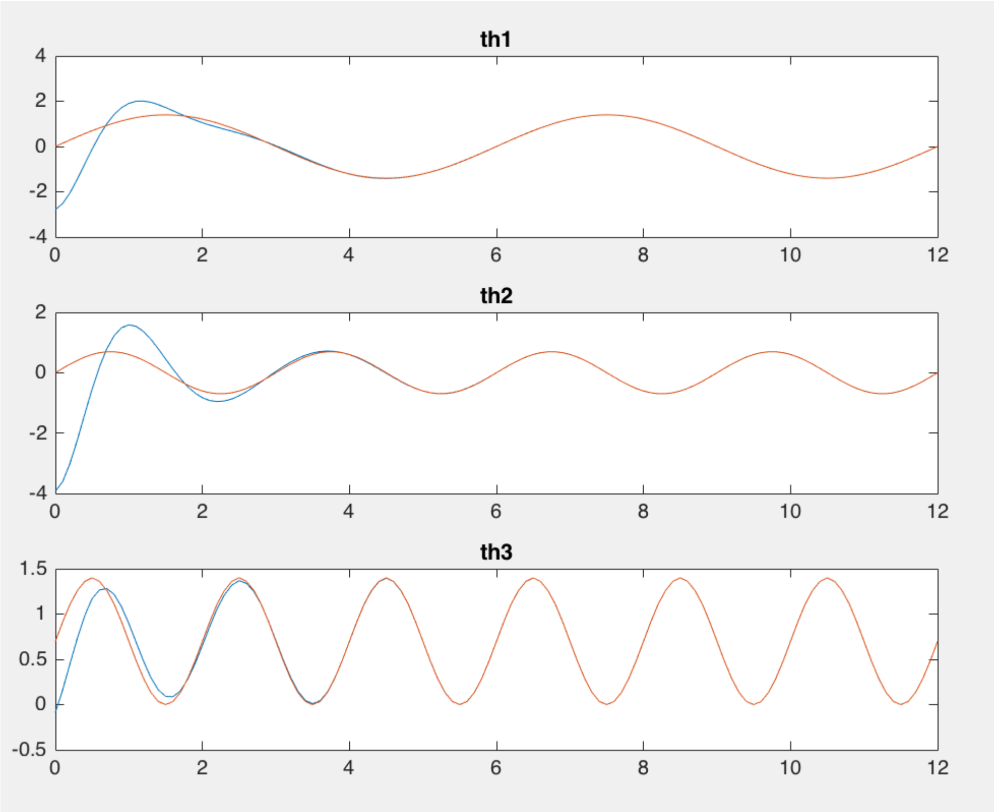
Άρα λοιπόν θα πρέπει για κάθε χρονική στιγμή να υπολογίζουμε την ροπη σύμφωνα με την εξίσωση

$$\overline{\tau}_{3\times1} = D_{3\times3}\left(\overline{\theta}\right)\left[\ddot{\overline{\theta}}^{d} + K_d\left(\dot{\overline{\theta}}^{d} - \dot{\overline{\theta}}\right) + K_p\left(\overline{\theta}^{d} - \overline{\theta}\right)\right] + C_{3\times1}\left(\overline{\theta}, \dot{\overline{\theta}}\right) + G_{3\times1}\left(\overline{\theta}\right)$$

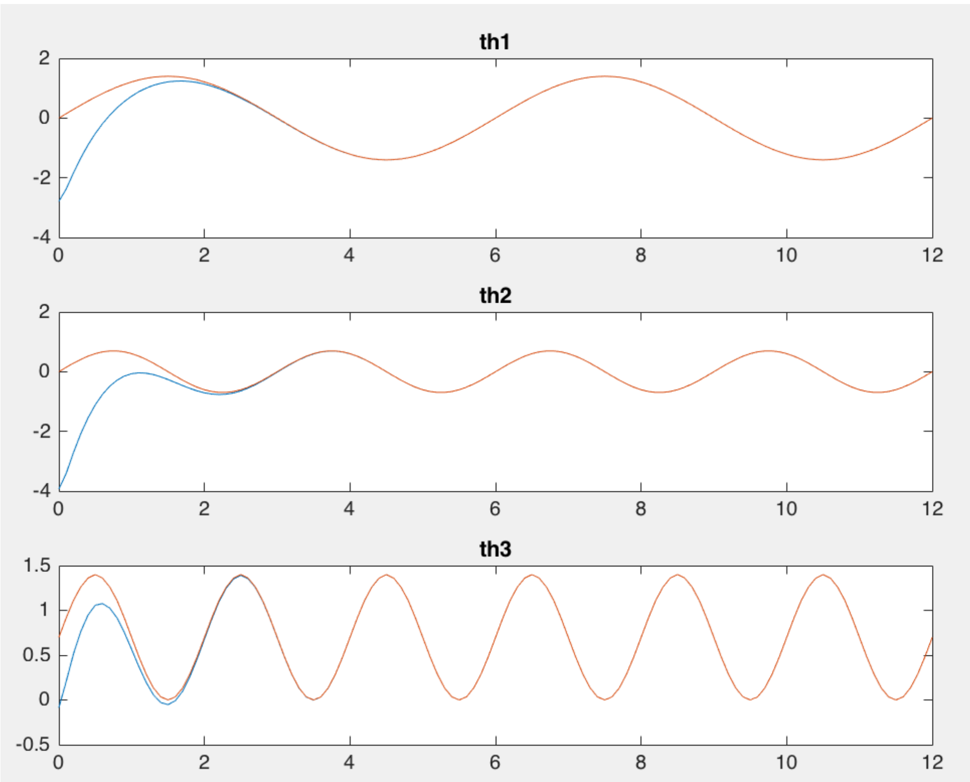
Και να συγκρίνουμε αν ξεπερνά τα 2τmax.

Για κάποιες random τιμές των Kp και Kd βλέπουμε πως μετά από κάποια δευτερόλεπτα προσεγγίζει την θdesired:

Για Kp=10, Kd=2.5:



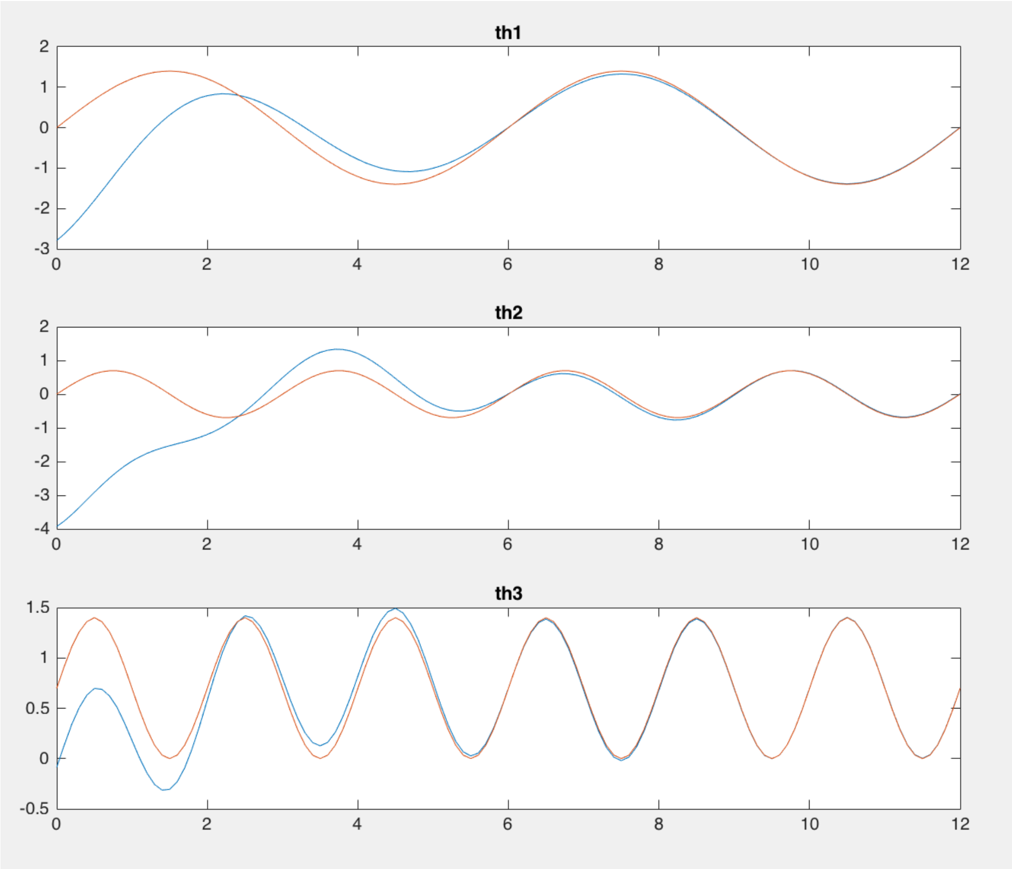
Για Kp=34, Kd=20:



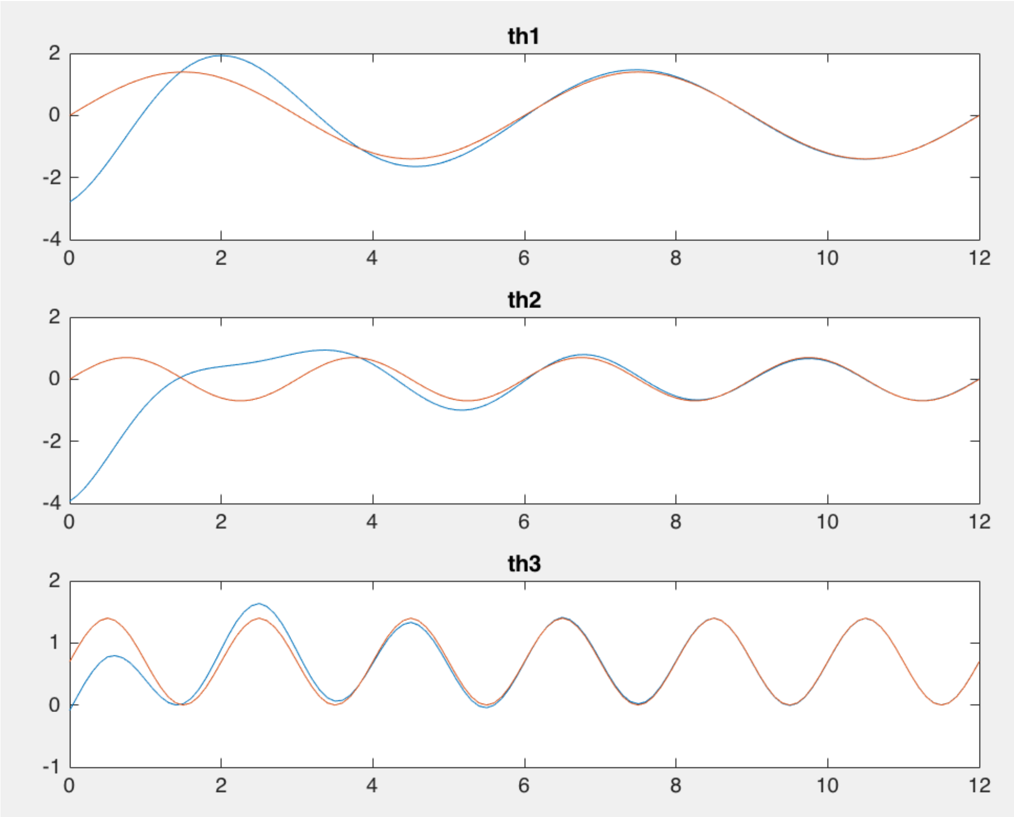
Συγκρίνοντας τώρα την ροπή για κάθε χρονική στιγμή με την τmax βρίσκουμε κάποιους συνδυασμούς που να ικανοποιούν αυτήν την συνθήκη.

Αυτοι οι συνδυασμοί είναι (Kp, Kd διαγώνιοι πίνακες παντα με στοιχεία= Kp kai Kd)

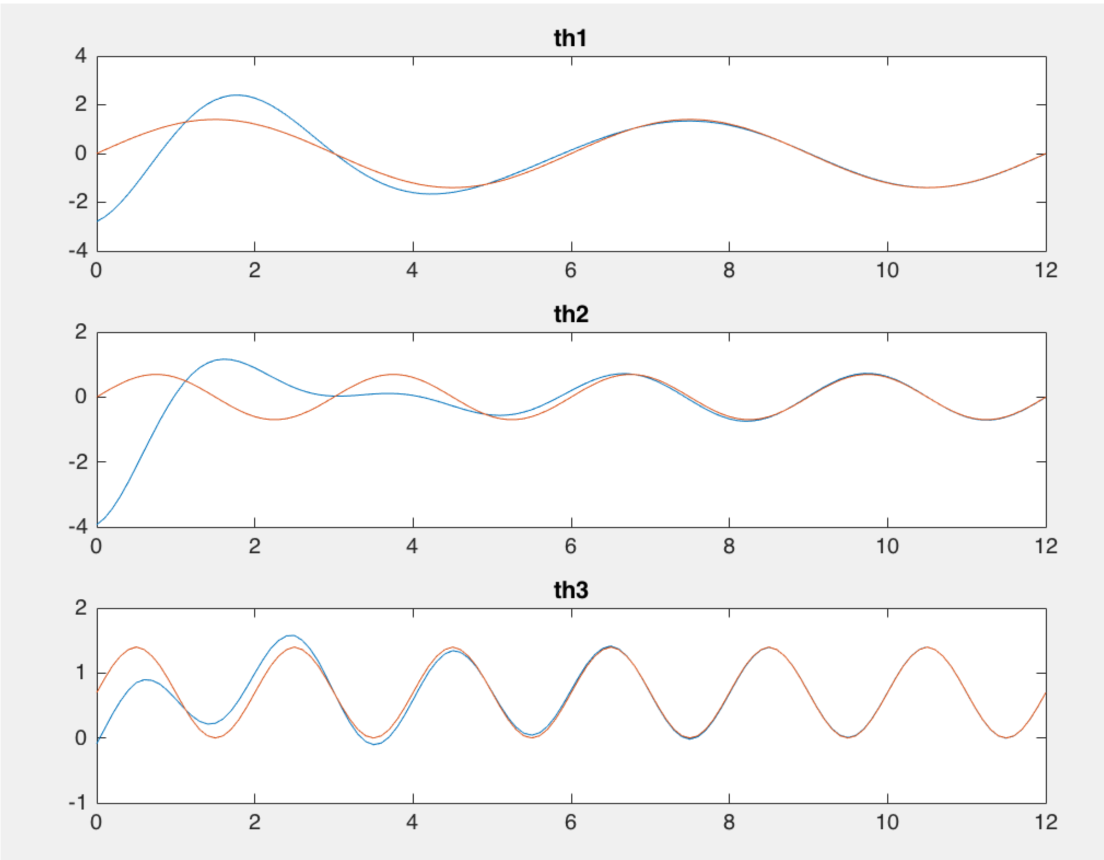
Για Kp=1 Kd=1



Για Kp=2 , Kd=1



Για Kp=2 , Kd=1



Στο αρχικό πρόγραμμα έχουμε τον εξής κώδικα:

```
q1=double([ -160*pi/180]);
q2=double([ -225*pi/180]);
q3=double([ -45*pi/180]);

u=0;
T_s=0.1;
z0=[q1 0 q2  0  q3  0]';
t_in=0;
t_fin=12;
cnt=1;
z=z0;
diaf=z0;
n=1;
Kp=3;Kd=1;

for t=t_in:T_s:t_fin-T_s
    cnt=cnt+1; %counter increment
    [tode,z]= ode45(@(t,z) F(t,z,D,C,G,th,thdot,thdotdot,thdesired,thdotdesired,thdotdotdesired,Kp,Kd), [t t+T_s]

    szx_ode=size(z);
    z=z(szx_ode(1),:)';
    diaf(:,cnt)=z;

end
t=t_in:T_s:t_fin;

Kp=eye(cnt)*Kp;
Kd=eye(cnt)*Kd;

plot(t,diaf)
```

που καλέι την function

```
function zdot=F(t,z,D,C,G,th,thdot,thdotdot,qdesired,qdotdesired,qdotdotdesired,Kp,Kd)
th1 = sym('th1');th2 = sym('th2');th3 = sym('th3');pi=sym('pi');
th1dot = sym('th1dot');th2dot = sym('th2dot');th3dot = sym('th3dot');

zdot=zeros(6,1); % since output must be a column vector

zdot(1)=z(2);
zdot(3)=z(4);
zdot(5)=z(6);

zdot(2)=-Kp*z(1)-Kd*zdot(1);
zdot(4)=-Kp*z(3)-Kd*zdot(3);
zdot(6)=-Kp*z(5)-Kd*zdot(5);


Dfound=subs(D,{th1 th2,th3},{z(1) z(3) z(5) });
Cfound=subs(C,{th1 th2,th3,th1dot,th2dot,th3dot},{z(1) z(3) z(5) z(2) z(4) z(6)}); %pinakas C
Gfound=subs(G,{th1, th2, th3},{z(1) z(3) z(5) }); %pinakas G
Dinv=inv(Dfound); %pinakas D^(-1)

torque1=Dinv*(qdotdotdesired(:,n)+(qdotdesired(:,n)-qdot(:,n))+(qdesired(:,n)-q(:,n)))+Gfound+Cfound;


end
```