MAE 263D

Final

Sepehr Ghassemi

Professor: Dr. Jacob Rosen

TA: Te-Kang Chow

INDEX

Newton Euler *3-18*

Lagrangian *18-30*

Tool Space Trajectory *31-37*

Joint Space Trajectory *37-44*

Extra *44-46*

Other *Electronically attached*

Dynamics-Newton Euler:

The calculations are done in MATLAB. The Code is attached.

All the steps are explained in the comments of the code to make it easy to flow and understand. The final result is published and available below.

Note that some animations and motion cannot be included in the paper. I have also tried not to make it short since it would distract the understanding and flow of document.

The tau terms are separated by order and simplified (as seen by the script, however the output is still very large.

```
%Final 263D
%Newton-Euler Method
%Sepehr Ghassemi


%Initial Clearing
clc
clear all
close all

%syms Defined

syms alpha1 alpha2 alpha3 alpha4 real
syms a1 a2 a3 a4 real
syms d1 d2 d3 d4 real
syms theta1 theta2 theta3 theta4 real
syms thetadot1 thetadot2 thetadot3 thetadot4 real
syms thetadotdot1 thetadotdot2 thetadotdot3 thetadotdot4 real
syms Ixx1 Ixx2 Ixx3 Iyy1 Iyy2 Iyy3 Izz1 Izz2 Izz3 real
syms m1 m2 m3 m4 real
syms POS1 POS2 POS3 POS4 real
syms COM1 COM2 COM3 COM4 real
syms cx1 cx2 cx3 cx4  cy1 cy2 cy3 cy4 cz1 cz2 cz3 cz4 real
syms avel1 avel2 avel3 avel4 real
syms aacc1 aacc2 aacc3 aacc4 real
syms acc1 acc2 acc3 acc4 real
syms accm1 accm2 accm3 accm4 real
syms gravity real
syms fx fy fz real
syms nx ny nz real
```

```
%DH Table and Transformations (with rotation and position matrices)

dhtable=[0 0 0 theta1;-pi/2 0 0 theta2;0 a2 d3 theta3;-pi/2 a3 d4 theta4]

for counter1=1:4
T(:,:,counter1)=[cos(dhtable(counter1,4)),-
sin(dhtable(counter1,4)),0,dhtable(counter1,2);sin(dhtable(counter1,4))*cos(dhtable(co
unter1,1)),cos(dhtable(counter1,4))*cos(dhtable(counter1,1)),-
sin(dhtable(counter1,1)),-
sin(dhtable(counter1,1))*dhtable(counter1,3);sin(dhtable(counter1,4))*sin(dhtable(coun
ter1,1)),cos(dhtable(counter1,4))*sin(dhtable(counter1,1)),cos(dhtable(counter1,1)),co
s(dhtable(counter1,1))*dhtable(counter1,3);0,0,0,1];
end

T01=T(:,:,1)
T12=T(:,:,2)
T23=T(:,:,3)
T34=T(:,:,4)

T02=T01*T12
T03=T01*T12*T23
T04=T01*T12*T23*T34

R01=T(1:3,1:3);
R12=T(1:3,1:3);
R23=T(1:3,1:3);
R34=T(1:3,1:3);

POS1=T01(1:3,4);
POS2=T02(1:3,4);
POS3=T03(1:3,4);
POS4=T04(1:3,4);


%Mass and Inertia
%With respect to frame
COM1=[cx1;cy1;cz1];
COM2=[cx2;cy2;cz2];
COM3=[cx3;cy3;cz3];
COM4=POS4;

I1=[Ixx1 0 0;0 Iyy1 0;0 0 Izz1]
I2=[Ixx2 0 0;0 Iyy2 0;0 0 Izz2]
I3=[Ixx3 0 0;0 Iyy3 0;0 0 Izz3]


%Newton Euler Approach

%angular velocities and acceleration

avel1=[0;0;thetadot1];
avel2=R12'*avel1+[0;0;thetadot2];
```

```
avel3=R23'*avel2+[0;0;thetadot3];
aacc1=[0;0;thetadotdot1];
aacc2=R12'*aacc1+[0;0;thetadotdot2];
aacc3=R23'*aacc2+[0;0;thetadotdot3];

%acceleration (gravity is expressed as [0;0;gravity])

acc1=R01'*[0;0;gravity];
    variable1=cross(aacc1,POS1);
    variable2=cross(avel1,(cross(avel1,POS1)));
acc2=R12'*((variable1)+(variable2)+acc1);
    variable3=cross(aacc2,POS2);
    variable4=cross(avel2,(cross(avel2,POS2)));
acc3=R23'*((variable3)+(variable4)+acc2);

accm1=cross(aacc1,COM1)+cross(avel1,(cross(avel1,COM1)))+acc1;
accm2=cross(aacc2,COM2)+cross(avel2,(cross(avel2,COM2)))+acc2;
accm3=cross(aacc3,COM3)+cross(avel3,(cross(avel3,COM3)))+acc3;
accm4=cross(aacc3,COM4)+cross(avel3,(cross(avel3,COM4)))+acc3;

%iterate forward

F1=m1*accm1;
F2=m2*accm2;
F3=m3*accm3;
F4=m4*accm4;

N1=I1*aacc1+cross(avel1,(I1*avel1));
N2=I2*aacc2+cross(avel2,(I1*avel2));
N3=I3*aacc3+cross(avel3,(I1*avel3));

N4=[0;0;0]; % no inertia as said by document

%external force
%No external force case when fx, fy, and fz are equal to 0
Fexternal=[fx;fy;fz];
Nexternal=[nx;ny;nz];

%iterate backward

f4=Fexternal+F4;
f3=R34'*f4+F3;
f2=R23'*f3+F2;
f1=R12'*f2+F1;

n4=N4+Nexternal+cross(POS4,F3);
n3=N3+R34'*n4+cross(COM3,F3)+cross(POS3,R34'*f4);
n2=N2+R23'*n3+cross(COM2,F2)+cross(POS2,R23'*f3);
n1=N1+R12'*n2+cross(COM1,F1)+cross(POS1,R12'*f2);

%Joint torque with simplification

tau =[n1(3);n2(3);n3(3)];
```

```
tau =simplify(expand(tau),10);
tau =simplify(tau);
tau = collect(tau,[thetadotdot1 thetadotdot2 thetadotdot3 thetadot1^2 thetadot2^2
thetadot3^2 gravity fx fy fz nx ny nz])

%EXTRA CREDIT numerical torque
%The same script is used with the values provided in place of syms.
%no external force will be needed. (output would be too long and cluttered
%to be useful.
```

```
dhtable =


[     0,  0,  0, theta1]

[ -pi/2,  0,  0, theta2]

[     0, a2, d3, theta3]

[ -pi/2, a3, d4, theta4]


T01 =


[ cos(theta1), -sin(theta1), 0, 0]

[ sin(theta1),  cos(theta1), 0, 0]

[           0,            0, 1, 0]

[           0,            0, 0, 1]


T12 =
```

```
[  cos(theta2), -sin(theta2), 0, 0]

[            0,            0, 1, 0]

[ -sin(theta2), -cos(theta2), 0, 0]

[            0,            0, 0, 1]



T23 =


[ cos(theta3), -sin(theta3), 0, a2]

[ sin(theta3),  cos(theta3), 0,  0]

[           0,            0, 1, d3]

[           0,            0, 0,  1]



T34 =


[  cos(theta4), -sin(theta4), 0, a3]

[            0,            0, 1, d4]

[ -sin(theta4), -cos(theta4), 0,  0]

[            0,            0, 0,  1]



T02 =
```

```
[ cos(theta1)*cos(theta2), -cos(theta1)*sin(theta2), -sin(theta1), 0]

[ cos(theta2)*sin(theta1), -sin(theta1)*sin(theta2),  cos(theta1), 0]

[            -sin(theta2),             -cos(theta2),            0, 0]

[                       0,                        0,            0, 1]




T03 =



[ cos(theta1)*cos(theta2)*cos(theta3) - cos(theta1)*sin(theta2)*sin(theta3), -
cos(theta1)*cos(theta2)*sin(theta3) - cos(theta1)*cos(theta3)*sin(theta2), -
sin(theta1), a2*cos(theta1)*cos(theta2) - d3*sin(theta1)]

[ cos(theta2)*cos(theta3)*sin(theta1) - sin(theta1)*sin(theta2)*sin(theta3), -
cos(theta2)*sin(theta1)*sin(theta3) - cos(theta3)*sin(theta1)*sin(theta2),
cos(theta1), d3*cos(theta1) + a2*cos(theta2)*sin(theta1)]

[                       - cos(theta2)*sin(theta3) - cos(theta3)*sin(theta2),
sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3),            0,
-a2*sin(theta2)]

[                                                                          0,
0,            0,                                                 1]




T04 =



[   sin(theta1)*sin(theta4) - cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)), cos(theta4)*sin(theta1) +
sin(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)), - cos(theta1)*cos(theta2)*sin(theta3) -
cos(theta1)*cos(theta3)*sin(theta2), a2*cos(theta1)*cos(theta2) -
d4*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2)) -
d3*sin(theta1) - a3*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3))]
```

```
[ - cos(theta1)*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)), sin(theta4)*(sin(theta1)*sin(theta2)*sin(theta3)
- cos(theta2)*cos(theta3)*sin(theta1)) - cos(theta1)*cos(theta4), -
cos(theta2)*sin(theta1)*sin(theta3) - cos(theta3)*sin(theta1)*sin(theta2),
d3*cos(theta1) - d4*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - a3*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + a2*cos(theta2)*sin(theta1)]

[                                                        -
cos(theta4)*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)),
sin(theta4)*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)),
sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3),
- a3*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)) - d4*(cos(theta2)*cos(theta3)
- sin(theta2)*sin(theta3)) - a2*sin(theta2)]

[
0,
0,                                                                                  0,
1]





I1 =



[ Ixx1,    0,    0]

[    0, Iyy1,    0]

[    0,    0, Izz1]





I2 =



[ Ixx2,    0,    0]

[    0, Iyy2,    0]

[    0,    0, Izz2]
```

```
I3 =


[ Ixx3,    0,    0]

[    0, Iyy3,    0]

[    0,    0, Izz3]


tau =


 (Izz1 + Izz2 + Izz3 + cx1^2*m1 + cx2^2*m2 + cx3^2*m3 + cy1^2*m1 + cy2^2*m2 + cy3^2*m3
+ d3^2*m4*cos(theta1) + cy3*d3*m3*cos(theta1) - cx3*d3*m3*sin(theta1) +
a2^2*m4*cos(theta1)*cos(theta2)^2 + a2*cx3*m3*cos(theta1)*cos(theta2) +
a2*cy3*m3*cos(theta2)*sin(theta1) + a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) -
a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) -
cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) -
cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) -
a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) +
d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) +
d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) -
a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot1 + (Izz2 + Izz3
+ cx2^2*m2 + cx3^2*m3 + cy2^2*m2 + cy3^2*m3 + d3^2*m4*cos(theta1) +
cy3*d3*m3*cos(theta1) - cx3*d3*m3*sin(theta1) + a2^2*m4*cos(theta1)*cos(theta2)^2 +
a2*cx3*m3*cos(theta1)*cos(theta2) + a2*cy3*m3*cos(theta2)*sin(theta1) +
a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) -
a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) -
cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) -
cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) -
a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) +
d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) +
```

```
d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) -
a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot2 + (Izz3 +
cx3^2*m3 + cy3^2*m3 + d3^2*m4*cos(theta1) + cy3*d3*m3*cos(theta1) -
cx3*d3*m3*sin(theta1) + a2^2*m4*cos(theta1)*cos(theta2)^2 +
a2*cx3*m3*cos(theta1)*cos(theta2) + a2*cy3*m3*cos(theta2)*sin(theta1) +
a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) -
a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) - cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) -
cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) - cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) +
d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) + d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) -
a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) + a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) - a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot3 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot1^2 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot2^2 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot3^2 + (d3 -
2*a2*cos(theta2)*sin(theta1)*cos(theta1) - 2*d3*cos(theta1)^2)*fx +
(2*a2*cos(theta2)*cos(theta1)^2 - d3*sin(2*theta1) - a2*cos(theta2))*fy + nz +
```

```
2*d3^2*m4*thetadot1*thetadot2*sin(theta1) + 2*d3^2*m4*thetadot1*thetadot3*sin(theta1) +
2*d3^2*m4*thetadot2*thetadot3*sin(theta1) + 2*cx3*d3*m3*thetadot1*thetadot2*cos(theta1) +
2*cx3*d3*m3*thetadot1*thetadot3*cos(theta1) + 2*cx3*d3*m3*thetadot2*thetadot3*cos(theta1) +
2*cy3*d3*m3*thetadot1*thetadot2*sin(theta1) + 2*cy3*d3*m3*thetadot1*thetadot3*sin(theta1) +
2*cy3*d3*m3*thetadot2*thetadot3*sin(theta1) +
2*a2^2*m4*thetadot1*thetadot2*cos(theta2)^2*sin(theta1) +
2*a2^2*m4*thetadot1*thetadot3*cos(theta2)^2*sin(theta1) +
2*a2^2*m4*thetadot2*thetadot3*cos(theta2)^2*sin(theta1) -
2*a2*cy3*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2) -
2*a2*cy3*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2) -
2*a2*cy3*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2) +
2*a2*cx3*m3*thetadot1*thetadot2*cos(theta2)*sin(theta1) +
2*a2*cx3*m3*thetadot1*thetadot3*cos(theta2)*sin(theta1) +
2*a2*cx3*m3*thetadot2*thetadot3*cos(theta2)*sin(theta1) -
2*a3*cy3*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2)*cos(theta3) -
2*a3*cy3*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) -
2*a3*cy3*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot1*thetadot2*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot1*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot2*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*cx3*m3*thetadot1*thetadot2*cos(theta2)*cos(theta3)*sin(theta1) +
2*a3*cx3*m3*thetadot1*thetadot3*cos(theta2)*cos(theta3)*sin(theta1) +
2*a3*cx3*m3*thetadot2*thetadot3*cos(theta2)*cos(theta3)*sin(theta1) +
2*cy3*d4*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2)*sin(theta3) +
2*cy3*d4*m3*thetadot1*thetadot2*cos(theta1)*cos(theta3)*sin(theta2) +
2*cy3*d4*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) +
2*cy3*d4*m3*thetadot1*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) +
2*cy3*d4*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) +
2*cy3*d4*m3*thetadot2*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot1*thetadot2*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot1*thetadot2*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot1*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot1*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot2*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot2*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) +
2*a3*cy3*m3*thetadot1*thetadot2*cos(theta1)*sin(theta2)*sin(theta3) +
2*a3*cy3*m3*thetadot1*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) +
2*a3*cy3*m3*thetadot2*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot1*thetadot2*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot1*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot2*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot2*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot2*cos(theta3)*sin(theta1)*sin(theta2) -
2*cx3*d4*m3*thetadot1*thetadot3*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot3*cos(theta3)*sin(theta1)*sin(theta2) -
2*cx3*d4*m3*thetadot2*thetadot3*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot2*thetadot3*cos(theta3)*sin(theta1)*sin(theta2) -
2*a3*cx3*m3*thetadot1*thetadot2*sin(theta1)*sin(theta2)*sin(theta3) -
2*a3*cx3*m3*thetadot1*thetadot3*sin(theta1)*sin(theta2)*sin(theta3) -
2*a3*cx3*m3*thetadot2*thetadot3*sin(theta1)*sin(theta2)*sin(theta3) +
2*a2*a3*m4*thetadot1*thetadot2*cos(theta2)^2*cos(theta3)*sin(theta1) +
2*a2*a3*m4*thetadot1*thetadot3*cos(theta2)^2*cos(theta3)*sin(theta1) +
2*a2*a3*m4*thetadot2*thetadot3*cos(theta2)^2*cos(theta3)*sin(theta1) -
2*a2*d4*m4*thetadot1*thetadot2*cos(theta2)^2*sin(theta1)*sin(theta3) -
```

```
2*a2*d4*m4*thetadot1*thetadot3*cos(theta2)^2*sin(theta1)*sin(theta3) -
2*a2*d4*m4*thetadot2*thetadot3*cos(theta2)^2*sin(theta1)*sin(theta3) -
2*a2*a3*m4*thetadot1*thetadot2*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*a3*m4*thetadot1*thetadot3*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*a3*m4*thetadot2*thetadot3*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*d4*m4*thetadot1*thetadot2*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
2*a2*d4*m4*thetadot1*thetadot3*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
2*a2*d4*m4*thetadot2*thetadot3*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)

                            (Izz2 + Izz3 + cx2^2*m2 + cx3^2*m3 + cy2^2*m2 + cy3^2*m3 +
d3^2*m4*cos(theta1) + cy3*d3*m3*cos(theta1) - cx3*d3*m3*sin(theta1) +
a2^2*m4*cos(theta1)*cos(theta2)^2 + a2*cx3*m3*cos(theta1)*cos(theta2) +
a2*cy3*m3*cos(theta2)*sin(theta1) + a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) - a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) -
cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) - cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) -
a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) - cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) + d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) +
d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) - a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) + a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot1 + (Izz2 + Izz3 + cx2^2*m2
+ cx3^2*m3 + cy2^2*m2 + cy3^2*m3 + d3^2*m4*cos(theta1) + cy3*d3*m3*cos(theta1) -
cx3*d3*m3*sin(theta1) + a2^2*m4*cos(theta1)*cos(theta2)^2 + a2*cx3*m3*cos(theta1)*cos(theta2) +
a2*cy3*m3*cos(theta2)*sin(theta1) + a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) - a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) -
cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) - cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) -
a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) - cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) + d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) +
d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) - a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) + a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot2 + (Izz3 + cx3^2*m3 +
cy3^2*m3 + d3^2*m4*cos(theta1) + cy3*d3*m3*cos(theta1) - cx3*d3*m3*sin(theta1) +
a2^2*m4*cos(theta1)*cos(theta2)^2 + a2*cx3*m3*cos(theta1)*cos(theta2) +
a2*cy3*m3*cos(theta2)*sin(theta1) + a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) - a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) -
cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) - cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) -
a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) - cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) + d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) +
d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) - a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) + a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot3 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
```

```
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot1^2 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot2^2 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot3^2 + (d3 -
2*a2*cos(theta2)*sin(theta1)*cos(theta1) - 2*d3*cos(theta1)^2)*fx +
(2*a2*cos(theta2)*cos(theta1)^2 - d3*sin(2*theta1) - a2*cos(theta2))*fy + nz +
2*d3^2*m4*thetadot1*thetadot2*sin(theta1) + 2*d3^2*m4*thetadot1*thetadot3*sin(theta1) +
2*d3^2*m4*thetadot2*thetadot3*sin(theta1) + 2*cx3*d3*m3*thetadot1*thetadot2*cos(theta1) +
2*cx3*d3*m3*thetadot1*thetadot3*cos(theta1) + 2*cx3*d3*m3*thetadot2*thetadot3*cos(theta1) +
2*cy3*d3*m3*thetadot1*thetadot2*sin(theta1) + 2*cy3*d3*m3*thetadot1*thetadot3*sin(theta1) +
2*cy3*d3*m3*thetadot2*thetadot3*sin(theta1) +
2*a2^2*m4*thetadot1*thetadot2*cos(theta2)^2*sin(theta1) +
2*a2^2*m4*thetadot1*thetadot3*cos(theta2)^2*sin(theta1) +
2*a2^2*m4*thetadot2*thetadot3*cos(theta2)^2*sin(theta1) -
2*a2*cy3*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2) -
2*a2*cy3*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2) -
2*a2*cy3*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2) +
2*a2*cx3*m3*thetadot1*thetadot2*cos(theta2)*sin(theta1) +
2*a2*cx3*m3*thetadot1*thetadot3*cos(theta2)*sin(theta1) +
2*a2*cx3*m3*thetadot2*thetadot3*cos(theta2)*sin(theta1) -
2*a3*cy3*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2)*cos(theta3) -
2*a3*cy3*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) -
2*a3*cy3*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot1*thetadot2*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot1*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot2*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*cx3*m3*thetadot1*thetadot2*cos(theta2)*cos(theta3)*sin(theta1) +
2*a3*cx3*m3*thetadot1*thetadot3*cos(theta2)*cos(theta3)*sin(theta1) +
2*a3*cx3*m3*thetadot2*thetadot3*cos(theta2)*cos(theta3)*sin(theta1) +
2*cy3*d4*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2)*sin(theta3) +
2*cy3*d4*m3*thetadot1*thetadot2*cos(theta1)*cos(theta3)*sin(theta2) +
2*cy3*d4*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) +
```

```
2*cy3*d4*m3*thetadot1*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) +
2*cy3*d4*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) +
2*cy3*d4*m3*thetadot2*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot1*thetadot2*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot1*thetadot2*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot1*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot1*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot2*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot2*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) +
2*a3*cy3*m3*thetadot1*thetadot2*cos(theta1)*sin(theta2)*sin(theta3) +
2*a3*cy3*m3*thetadot1*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) +
2*a3*cy3*m3*thetadot2*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot1*thetadot2*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot1*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot2*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot2*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot2*cos(theta3)*sin(theta1)*sin(theta2) -
2*cx3*d4*m3*thetadot1*thetadot3*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot3*cos(theta3)*sin(theta1)*sin(theta2) -
2*cx3*d4*m3*thetadot2*thetadot3*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot2*thetadot3*cos(theta3)*sin(theta1)*sin(theta2) -
2*a3*cx3*m3*thetadot1*thetadot2*sin(theta1)*sin(theta2)*sin(theta3) -
2*a3*cx3*m3*thetadot1*thetadot3*sin(theta1)*sin(theta2)*sin(theta3) -
2*a3*cx3*m3*thetadot2*thetadot3*sin(theta1)*sin(theta2)*sin(theta3) +
2*a2*a3*m4*thetadot1*thetadot2*cos(theta2)^2*cos(theta3)*sin(theta1) +
2*a2*a3*m4*thetadot1*thetadot3*cos(theta2)^2*cos(theta3)*sin(theta1) +
2*a2*a3*m4*thetadot2*thetadot3*cos(theta2)^2*cos(theta3)*sin(theta1) -
2*a2*d4*m4*thetadot1*thetadot2*cos(theta2)^2*sin(theta1)*sin(theta3) -
2*a2*d4*m4*thetadot1*thetadot3*cos(theta2)^2*sin(theta1)*sin(theta3) -
2*a2*d4*m4*thetadot2*thetadot3*cos(theta2)^2*sin(theta1)*sin(theta3) -
2*a2*a3*m4*thetadot1*thetadot2*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*a3*m4*thetadot1*thetadot3*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*a3*m4*thetadot2*thetadot3*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*d4*m4*thetadot1*thetadot2*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
2*a2*d4*m4*thetadot1*thetadot3*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
2*a2*d4*m4*thetadot2*thetadot3*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)

                                                                                (Izz3 +
cx3^2*m3 + cy3^2*m3 + d3^2*m4*cos(theta1) + cy3*d3*m3*cos(theta1) - cx3*d3*m3*sin(theta1) +
a2^2*m4*cos(theta1)*cos(theta2)^2 + a2*cx3*m3*cos(theta1)*cos(theta2) +
a2*cy3*m3*cos(theta2)*sin(theta1) + a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) - a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) -
cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) - cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) -
a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) - cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) + d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) +
d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) - a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) + a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot1 + (Izz3 + cx3^2*m3 +
cy3^2*m3 + d3^2*m4*cos(theta1) + cy3*d3*m3*cos(theta1) - cx3*d3*m3*sin(theta1) +
a2^2*m4*cos(theta1)*cos(theta2)^2 + a2*cx3*m3*cos(theta1)*cos(theta2) +
a2*cy3*m3*cos(theta2)*sin(theta1) + a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) +
```

```
a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) - a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) -
cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) - cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) -
a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) - cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) + d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) +
d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) - a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) + a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot2 + (Izz3 + cx3^2*m3 +
cy3^2*m3 + d3^2*m4*cos(theta1) + cy3*d3*m3*cos(theta1) - cx3*d3*m3*sin(theta1) +
a2^2*m4*cos(theta1)*cos(theta2)^2 + a2*cx3*m3*cos(theta1)*cos(theta2) +
a2*cy3*m3*cos(theta2)*sin(theta1) + a3*cx3*m3*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cy3*m3*cos(theta2)*cos(theta3)*sin(theta1) - a3*d3*m4*cos(theta2)*cos(theta3)*sin(theta1) -
cx3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) - cx3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) -
a3*cx3*m3*cos(theta1)*sin(theta2)*sin(theta3) - cy3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cy3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) + d3*d4*m4*cos(theta2)*sin(theta1)*sin(theta3) +
d3*d4*m4*cos(theta3)*sin(theta1)*sin(theta2) - a3*cy3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a3*d3*m4*sin(theta1)*sin(theta2)*sin(theta3) + a2*a3*m4*cos(theta1)*cos(theta2)^2*cos(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)^2*sin(theta3) -
a2*d4*m4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2) -
a2*a3*m4*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3))*thetadotdot3 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot1^2 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot2^2 + (d3^2*m4*sin(theta1) +
cx3*d3*m3*cos(theta1) + cy3*d3*m3*sin(theta1) + a2^2*m4*cos(theta2)^2*sin(theta1) -
a2*cy3*m3*cos(theta1)*cos(theta2) + a2*cx3*m3*cos(theta2)*sin(theta1) -
a3*cy3*m3*cos(theta1)*cos(theta2)*cos(theta3) + a3*d3*m4*cos(theta1)*cos(theta2)*cos(theta3) +
a3*cx3*m3*cos(theta2)*cos(theta3)*sin(theta1) + cy3*d4*m3*cos(theta1)*cos(theta2)*sin(theta3) +
cy3*d4*m3*cos(theta1)*cos(theta3)*sin(theta2) - d3*d4*m4*cos(theta1)*cos(theta2)*sin(theta3) -
d3*d4*m4*cos(theta1)*cos(theta3)*sin(theta2) + a3*cy3*m3*cos(theta1)*sin(theta2)*sin(theta3) -
a3*d3*m4*cos(theta1)*sin(theta2)*sin(theta3) - cx3*d4*m3*cos(theta2)*sin(theta1)*sin(theta3) -
cx3*d4*m3*cos(theta3)*sin(theta1)*sin(theta2) - a3*cx3*m3*sin(theta1)*sin(theta2)*sin(theta3) +
a2*a3*m4*cos(theta2)^2*cos(theta3)*sin(theta1) - a2*d4*m4*cos(theta2)^2*sin(theta1)*sin(theta3) -
a2*d4*m4*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
a2*a3*m4*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3))*thetadot3^2 + (d3 -
```

```
2*a2*cos(theta2)*sin(theta1)*cos(theta1) - 2*d3*cos(theta1)^2)*fx +
(2*a2*cos(theta2)*cos(theta1)^2 - d3*sin(2*theta1) - a2*cos(theta2))*fy + nz +
2*d3^2*m4*thetadot1*thetadot2*sin(theta1) + 2*d3^2*m4*thetadot1*thetadot3*sin(theta1) +
2*d3^2*m4*thetadot2*thetadot3*sin(theta1) + 2*cx3*d3*m3*thetadot1*thetadot2*cos(theta1) +
2*cx3*d3*m3*thetadot1*thetadot3*cos(theta1) + 2*cx3*d3*m3*thetadot2*thetadot3*cos(theta1) +
2*cy3*d3*m3*thetadot1*thetadot2*sin(theta1) + 2*cy3*d3*m3*thetadot1*thetadot3*sin(theta1) +
2*cy3*d3*m3*thetadot2*thetadot3*sin(theta1) +
2*a2^2*m4*thetadot1*thetadot2*cos(theta2)^2*sin(theta1) +
2*a2^2*m4*thetadot1*thetadot3*cos(theta2)^2*sin(theta1) +
2*a2^2*m4*thetadot2*thetadot3*cos(theta2)^2*sin(theta1) -
2*a2*cy3*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2) -
2*a2*cy3*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2) -
2*a2*cy3*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2) +
2*a2*cx3*m3*thetadot1*thetadot2*cos(theta2)*sin(theta1) +
2*a2*cx3*m3*thetadot1*thetadot3*cos(theta2)*sin(theta1) +
2*a2*cx3*m3*thetadot2*thetadot3*cos(theta2)*sin(theta1) -
2*a3*cy3*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2)*cos(theta3) -
2*a3*cy3*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) -
2*a3*cy3*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot1*thetadot2*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot1*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*d3*m4*thetadot2*thetadot3*cos(theta1)*cos(theta2)*cos(theta3) +
2*a3*cx3*m3*thetadot1*thetadot2*cos(theta2)*cos(theta3)*sin(theta1) +
2*a3*cx3*m3*thetadot1*thetadot3*cos(theta2)*cos(theta3)*sin(theta1) +
2*a3*cx3*m3*thetadot2*thetadot3*cos(theta2)*cos(theta3)*sin(theta1) +
2*cy3*d4*m3*thetadot1*thetadot2*cos(theta1)*cos(theta2)*sin(theta3) +
2*cy3*d4*m3*thetadot1*thetadot2*cos(theta1)*cos(theta3)*sin(theta2) +
2*cy3*d4*m3*thetadot1*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) +
2*cy3*d4*m3*thetadot1*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) +
2*cy3*d4*m3*thetadot2*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) +
2*cy3*d4*m3*thetadot2*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot1*thetadot2*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot1*thetadot2*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot1*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot1*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) -
2*d3*d4*m4*thetadot2*thetadot3*cos(theta1)*cos(theta2)*sin(theta3) -
2*d3*d4*m4*thetadot2*thetadot3*cos(theta1)*cos(theta3)*sin(theta2) +
2*a3*cy3*m3*thetadot1*thetadot2*cos(theta1)*sin(theta2)*sin(theta3) +
2*a3*cy3*m3*thetadot1*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) +
2*a3*cy3*m3*thetadot2*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot1*thetadot2*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot1*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*a3*d3*m4*thetadot2*thetadot3*cos(theta1)*sin(theta2)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot2*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot2*cos(theta3)*sin(theta1)*sin(theta2) -
2*cx3*d4*m3*thetadot1*thetadot3*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot1*thetadot3*cos(theta3)*sin(theta1)*sin(theta2) -
2*cx3*d4*m3*thetadot2*thetadot3*cos(theta2)*sin(theta1)*sin(theta3) -
2*cx3*d4*m3*thetadot2*thetadot3*cos(theta3)*sin(theta1)*sin(theta2) -
2*a3*cx3*m3*thetadot1*thetadot2*sin(theta1)*sin(theta2)*sin(theta3) -
2*a3*cx3*m3*thetadot1*thetadot3*sin(theta1)*sin(theta2)*sin(theta3) -
2*a3*cx3*m3*thetadot2*thetadot3*sin(theta1)*sin(theta2)*sin(theta3) +
2*a2*a3*m4*thetadot1*thetadot2*cos(theta2)^2*cos(theta3)*sin(theta1) +
2*a2*a3*m4*thetadot1*thetadot3*cos(theta2)^2*cos(theta3)*sin(theta1) +
```

```
2*a2*a3*m4*thetadot2*thetadot3*cos(theta2)^2*cos(theta3)*sin(theta1) -
2*a2*d4*m4*thetadot1*thetadot2*cos(theta2)^2*sin(theta1)*sin(theta3) -
2*a2*d4*m4*thetadot1*thetadot3*cos(theta2)^2*sin(theta1)*sin(theta3) -
2*a2*d4*m4*thetadot2*thetadot3*cos(theta2)^2*sin(theta1)*sin(theta3) -
2*a2*a3*m4*thetadot1*thetadot2*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*a3*m4*thetadot1*thetadot3*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*a3*m4*thetadot2*thetadot3*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) -
2*a2*d4*m4*thetadot1*thetadot2*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
2*a2*d4*m4*thetadot1*thetadot3*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) -
2*a2*d4*m4*thetadot2*thetadot3*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)
```

Dynamics Lagrangian :

Note that although the methods agree, this cod gives a much more simplified output.

In addition, I have kept some of the repeating things such as transformations and inertia to help with the flow and grading.

```
%Final 263D
%Lagrangian Method
%Sepehr Ghassemi


%Initial Clearing
clc
clear all
close all

%syms Defined

syms alpha1 alpha2 alpha3 alpha4 real
syms a1 a2 a3 a4 real
syms d1 d2 d3 d4 real
syms theta1 theta2 theta3 theta4 real
syms thetadot1 thetadot2 thetadot3 thetadot4 real
syms thetadotdot1 thetadotdot2 thetadotdot3 thetadotdot4 real
syms Ixx1 Ixx2 Ixx3 Iyy1 Iyy2 Iyy3 Izz1 Izz2 Izz3 real
syms m1 m2 m3 m4 real
syms POS1 POS2 POS3 POS4 real
syms COM1 COM2 COM3 COM4 real
syms COMB1 COMB2 COMB3 COMB4 real
syms cx1 cx2 cx3 cx4  cy1 cy2 cy3 cy4 cz1 cz2 cz3 cz4 real
syms velc1 velc2 velc3 velc4 real
syms avel1 avel2 avel3 avel4 real
syms K1 K2 K3 K4 KTOT real
syms P1 P2 P3 P4 PTOT real
syms L real
syms DL1 DL2 DL3 real
syms DK1 DK2 DK3 real
syms DDK1 DDK2 DDK3 real
syms DDDK1 DDDK2 DDDK3 real
```

```
syms gravity real
syms fx fy fz real
syms nx ny nz real

%DH Table and Transformations (with rotation and position matrices)

dhtable=[0 0 0 theta1;-pi/2 0 0 theta2;0 a2 d3 theta3;-pi/2 a3 d4 theta4]

for counter1=1:4
T(:,:,counter1)=[cos(dhtable(counter1,4)),-
sin(dhtable(counter1,4)),0,dhtable(counter1,2);sin(dhtable(counter1,4))*cos(dhtable(co
unter1,1)),cos(dhtable(counter1,4))*cos(dhtable(counter1,1)),-
sin(dhtable(counter1,1)),-
sin(dhtable(counter1,1))*dhtable(counter1,3);sin(dhtable(counter1,4))*sin(dhtable(coun
ter1,1)),cos(dhtable(counter1,4))*sin(dhtable(counter1,1)),cos(dhtable(counter1,1)),co
s(dhtable(counter1,1))*dhtable(counter1,3);0,0,0,1];
end

T01=T(:,:,1)
T12=T(:,:,2)
T23=T(:,:,3)
T34=T(:,:,4)

T02=T01*T12
T03=T01*T12*T23
T04=T01*T12*T23*T34

R01=T(1:3,1:3);
R12=T(1:3,1:3);
R23=T(1:3,1:3);
R34=T(1:3,1:3);

POS1=T01(1:3,4);
POS2=T02(1:3,4);
POS3=T03(1:3,4);
POS4=T04(1:3,4);


%Mass and Inertia

%With respect to frame
COM1=[cx1;cy1;cz1];
COM2=[cx2;cy2;cz2];
COM3=[cx3;cy3;cz3];
COM4=POS4;
%With respect to Base
COMB1=POS1+R01*COM1;
COMB2=POS2+R12*COM3;
COMB3=POS2+R23*COM3;
COMB4=POS4;

%moment of inertia
I1=[Ixx1 0 0;0 Iyy1 0;0 0 Izz1]
```

```
I2=[Ixx2 0 0;0 Iyy2 0;0 0 Izz2]
I3=[Ixx3 0 0;0 Iyy3 0;0 0 Izz3]


%Lagrangian Approach

%velocitie based on COM
velc1=thetadot1*diff(COMB1,theta1);
velc2=thetadot1*diff(COMB2,theta1)+thetadot2*diff(COMB2,theta2);
velc3=thetadot1*diff(COMB3,theta1)+thetadot2*diff(COMB3,theta2)+thetadot3*diff(COMB3,t
heta3);
velc4=thetadot1*diff(COMB4,theta1)+thetadot2*diff(COMB4,theta2)+thetadot3*diff(COMB4,t
heta3);
%velocities angular
avel1=[0;0;thetadot1];
avel2=R12'*avel1+[0;0;thetadot2];
avel3=R23'*avel2+[0;0;thetadot3];

%Kinetic energy
K1=0.5*m1*velc1'*velc1+0.5*avel1'*I1*avel1;
K2=0.5*m2*velc2'*velc2+0.5*avel2'*I2*avel2;
K3=0.5*m3*velc3'*velc3+0.5*avel3'*I3*avel3;
K4=0.5*m4*velc4'*velc4+0;
KTOT=K1+K2+K3+K4

%Potential energy
P1=m1*gravity*COMB1;
P2=m2*gravity*COMB2;
P3=m3*gravity*COMB3;
P4=m4*gravity*COMB4;
PTOT=P1+P2+P3+P4

%Lagrangian
L=KTOT-PTOT

%Derivitive calculations to get tau

DL1=diff(L,theta1);
DL2=diff(L,theta2);
DL3=diff(L,theta3);


DK1=diff(KTOT,thetadot1);
DK2=diff(KTOT,thetadot2);
DK3=diff(KTOT,thetadot3);

DDK1=thetadot1*diff(DK1,theta1);
DDK2=thetadot2*diff(DK2,theta2);
DDK3=thetadot3*diff(DK3,theta3);

DDDK1=thetadotdot1*diff(DK1,thetadot1);
DDDK2=thetadotdot2*diff(DK2,thetadot2);
DDDK3=thetadotdot3*diff(DK3,thetadot3);
```

```
%adding one variable for mathematical simplicity
VARRR1=(DDK1+DDDK1)-DL1;
VARRR2=(DDK2+DDDK2)-DL2;
VARRR3=(DDK3+DDDK3)-DL3;

%note this yields almost same answer as euler with 0 external force. here.
%we add a -Jf to the torqu to account for the external force if any.

%finding torque

tau=[VARRR1;VARRR2;VARRR3];
tau =simplify(expand(tau),10);
tau =simplify(tau);
tau = collect(tau,[thetadotdot1 thetadotdot2 thetadotdot3 thetadot1^2 thetadot2^2
thetadot3^2 gravity fx fy fz nx ny nz])
```

```
dhtable =


[     0,  0,  0, theta1]

[ -pi/2,  0,  0, theta2]

[     0, a2, d3, theta3]

[ -pi/2, a3, d4, theta4]


T01 =


[ cos(theta1), -sin(theta1), 0, 0]

[ sin(theta1),  cos(theta1), 0, 0]

[           0,            0, 1, 0]

[           0,            0, 0, 1]
```

```
T12 =


[  cos(theta2), -sin(theta2), 0, 0]

[            0,            0, 1, 0]

[ -sin(theta2), -cos(theta2), 0, 0]

[            0,            0, 0, 1]




T23 =


[ cos(theta3), -sin(theta3), 0, a2]

[ sin(theta3),  cos(theta3), 0,  0]

[           0,            0, 1, d3]

[           0,            0, 0,  1]




T34 =


[  cos(theta4), -sin(theta4), 0, a3]

[            0,            0, 1, d4]

[ -sin(theta4), -cos(theta4), 0,  0]

[            0,            0, 0,  1]
```

```
T02 =



[ cos(theta1)*cos(theta2), -cos(theta1)*sin(theta2), -sin(theta1), 0]

[ cos(theta2)*sin(theta1), -sin(theta1)*sin(theta2),  cos(theta1), 0]

[            -sin(theta2),             -cos(theta2),            0, 0]

[                       0,                        0,            0, 1]





T03 =



[ cos(theta1)*cos(theta2)*cos(theta3) - cos(theta1)*sin(theta2)*sin(theta3), -
cos(theta1)*cos(theta2)*sin(theta3) - cos(theta1)*cos(theta3)*sin(theta2), -
sin(theta1), a2*cos(theta1)*cos(theta2) - d3*sin(theta1)]

[ cos(theta2)*cos(theta3)*sin(theta1) - sin(theta1)*sin(theta2)*sin(theta3), -
cos(theta2)*sin(theta1)*sin(theta3) - cos(theta3)*sin(theta1)*sin(theta2),
cos(theta1), d3*cos(theta1) + a2*cos(theta2)*sin(theta1)]

[                       - cos(theta2)*sin(theta3) - cos(theta3)*sin(theta2),
sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3),            0,
-a2*sin(theta2)]

[                                                                          0,
0,            0,                                                 1]





T04 =



[   sin(theta1)*sin(theta4) - cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)), cos(theta4)*sin(theta1) +
sin(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)), - cos(theta1)*cos(theta2)*sin(theta3) -
cos(theta1)*cos(theta3)*sin(theta2), a2*cos(theta1)*cos(theta2) -
d4*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2)) -
```

```
d3*sin(theta1) - a3*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3))]

[ - cos(theta1)*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)), sin(theta4)*(sin(theta1)*sin(theta2)*sin(theta3)
- cos(theta2)*cos(theta3)*sin(theta1)) - cos(theta1)*cos(theta4), -
cos(theta2)*sin(theta1)*sin(theta3) - cos(theta3)*sin(theta1)*sin(theta2),
d3*cos(theta1) - d4*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - a3*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + a2*cos(theta2)*sin(theta1)]

[                                                       -
cos(theta4)*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)),
sin(theta4)*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)),
sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3),
- a3*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)) - d4*(cos(theta2)*cos(theta3)
- sin(theta2)*sin(theta3)) - a2*sin(theta2)]

[
0,
0,                                                                                  0,
1]





I1 =



[ Ixx1,    0,    0]

[    0, Iyy1,    0]

[    0,    0, Izz1]





I2 =



[ Ixx2,    0,    0]

[    0, Iyy2,    0]
```

```
[    0,    0, Izz2]



I3 =


[ Ixx3,    0,    0]

[    0, Iyy3,    0]

[    0,    0, Izz3]



KTOT =


(m4*(thetadot2*(a3*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) -
d4*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)) + a2*cos(theta2)) +
thetadot3*(a3*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) -
d4*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2))))^2)/2 +
(m4*(thetadot3*(a3*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - d4*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3))) -
thetadot1*(a3*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + d4*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d3*cos(theta1) - a2*cos(theta2)*sin(theta1)) +
thetadot2*(a3*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - d4*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + a2*cos(theta1)*sin(theta2)))^2)/2 +
(Izz1*thetadot1^2)/2 + (m4*(thetadot2*(a3*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d4*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + a2*sin(theta1)*sin(theta2)) +
thetadot1*(a3*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + d4*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) + d3*sin(theta1) - a2*cos(theta1)*cos(theta2)) +
thetadot3*(a3*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d4*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1))))^2)/2 + Izz3*(thetadot1 + thetadot2 +
thetadot3)*(thetadot1/2 + thetadot2/2 + thetadot3/2) +
(m1*thetadot1^2*(cy1*cos(theta1) + cx1*sin(theta1))^2)/2 +
(m1*thetadot1^2*(cx1*cos(theta1) - cy1*sin(theta1))^2)/2 +
(m2*thetadot1^2*(cy3*cos(theta1) + cx3*sin(theta1))^2)/2 +
(m2*thetadot1^2*(cx3*cos(theta1) - cy3*sin(theta1))^2)/2 +
```

```
(m3*thetadot1^2*(cy3*cos(theta1) + cx3*sin(theta1))^2)/2 +
(m3*thetadot1^2*(cx3*cos(theta1) - cy3*sin(theta1))^2)/2 + Izz2*(thetadot1/2 +
thetadot2/2)*(thetadot1 + thetadot2)
```

```
PTOT =
```

```
 gravity*m1*(cx1*cos(theta1) - cy1*sin(theta1)) -
gravity*m4*(a3*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + d4*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) + d3*sin(theta1) - a2*cos(theta1)*cos(theta2)) +
gravity*m2*(cx3*cos(theta1) - cy3*sin(theta1)) + gravity*m3*(cx3*cos(theta1) -
cy3*sin(theta1))

 gravity*m1*(cy1*cos(theta1) + cx1*sin(theta1)) -
gravity*m4*(a3*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + d4*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d3*cos(theta1) - a2*cos(theta2)*sin(theta1)) +
gravity*m2*(cy3*cos(theta1) + cx3*sin(theta1)) + gravity*m3*(cy3*cos(theta1) +
cx3*sin(theta1))


cz1*gravity*m1 + cz3*gravity*m2 + cz3*gravity*m3 -
gravity*m4*(a3*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)) +
d4*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) + a2*sin(theta2))
```

```
L =
```

```
 (m4*(thetadot2*(a3*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) -
d4*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)) + a2*cos(theta2)) +
thetadot3*(a3*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) -
d4*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2))))^2)/2 +
(m4*(thetadot3*(a3*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - d4*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3))) -
thetadot1*(a3*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + d4*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d3*cos(theta1) - a2*cos(theta2)*sin(theta1)) +
thetadot2*(a3*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - d4*(cos(theta1)*sin(theta2)*sin(theta3) -
```

```
cos(theta1)*cos(theta2)*cos(theta3)) + a2*cos(theta1)*sin(theta2)))^2)/2 +
(Izz1*thetadot1^2)/2 + (m4*(thetadot2*(a3*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d4*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + a2*sin(theta1)*sin(theta2)) +
thetadot1*(a3*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + d4*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) + d3*sin(theta1) - a2*cos(theta1)*cos(theta2)) +
thetadot3*(a3*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d4*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1))))^2)/2 + Izz3*(thetadot1 + thetadot2 +
thetadot3)*(thetadot1/2 + thetadot2/2 + thetadot3/2) +
(m1*thetadot1^2*(cy1*cos(theta1) + cx1*sin(theta1))^2)/2 +
(m1*thetadot1^2*(cx1*cos(theta1) - cy1*sin(theta1))^2)/2 +
(m2*thetadot1^2*(cy3*cos(theta1) + cx3*sin(theta1))^2)/2 +
(m2*thetadot1^2*(cx3*cos(theta1) - cy3*sin(theta1))^2)/2 +
(m3*thetadot1^2*(cy3*cos(theta1) + cx3*sin(theta1))^2)/2 +
(m3*thetadot1^2*(cx3*cos(theta1) - cy3*sin(theta1))^2)/2 +
gravity*m4*(a3*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + d4*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) + d3*sin(theta1) - a2*cos(theta1)*cos(theta2)) -
gravity*m1*(cx1*cos(theta1) - cy1*sin(theta1)) - gravity*m2*(cx3*cos(theta1) -
cy3*sin(theta1)) - gravity*m3*(cx3*cos(theta1) - cy3*sin(theta1)) + Izz2*(thetadot1/2
+ thetadot2/2)*(thetadot1 + thetadot2)

 (m4*(thetadot2*(a3*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) -
d4*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)) + a2*cos(theta2)) +
thetadot3*(a3*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) -
d4*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2))))^2)/2 +
(m4*(thetadot3*(a3*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - d4*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3))) -
thetadot1*(a3*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + d4*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d3*cos(theta1) - a2*cos(theta2)*sin(theta1)) +
thetadot2*(a3*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - d4*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + a2*cos(theta1)*sin(theta2)))^2)/2 +
(Izz1*thetadot1^2)/2 + (m4*(thetadot2*(a3*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d4*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + a2*sin(theta1)*sin(theta2)) +
thetadot1*(a3*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + d4*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) + d3*sin(theta1) - a2*cos(theta1)*cos(theta2)) +
thetadot3*(a3*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d4*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1))))^2)/2 +
gravity*m4*(a3*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + d4*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d3*cos(theta1) - a2*cos(theta2)*sin(theta1)) +
Izz3*(thetadot1 + thetadot2 + thetadot3)*(thetadot1/2 + thetadot2/2 + thetadot3/2) +
(m1*thetadot1^2*(cy1*cos(theta1) + cx1*sin(theta1))^2)/2 +
(m1*thetadot1^2*(cx1*cos(theta1) - cy1*sin(theta1))^2)/2 +
(m2*thetadot1^2*(cy3*cos(theta1) + cx3*sin(theta1))^2)/2 +
```

```
(m2*thetadot1^2*(cx3*cos(theta1) - cy3*sin(theta1))^2)/2 +
(m3*thetadot1^2*(cy3*cos(theta1) + cx3*sin(theta1))^2)/2 +
(m3*thetadot1^2*(cx3*cos(theta1) - cy3*sin(theta1))^2)/2 - gravity*m1*(cy1*cos(theta1)
+ cx1*sin(theta1)) - gravity*m2*(cy3*cos(theta1) + cx3*sin(theta1)) -
gravity*m3*(cy3*cos(theta1) + cx3*sin(theta1)) + Izz2*(thetadot1/2 +
thetadot2/2)*(thetadot1 + thetadot2)


(m4*(thetadot2*(a3*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) -
d4*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2)) + a2*cos(theta2)) +
thetadot3*(a3*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) -
d4*(cos(theta2)*sin(theta3) + cos(theta3)*sin(theta2))))^2)/2 +
(m4*(thetadot3*(a3*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - d4*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3))) -
thetadot1*(a3*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + d4*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d3*cos(theta1) - a2*cos(theta2)*sin(theta1)) +
thetadot2*(a3*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - d4*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + a2*cos(theta1)*sin(theta2)))^2)/2 +
(Izz1*thetadot1^2)/2 + (m4*(thetadot2*(a3*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d4*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1)) + a2*sin(theta1)*sin(theta2)) +
thetadot1*(a3*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3)) + d4*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) + d3*sin(theta1) - a2*cos(theta1)*cos(theta2)) +
thetadot3*(a3*(cos(theta2)*sin(theta1)*sin(theta3) +
cos(theta3)*sin(theta1)*sin(theta2)) - d4*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1))))^2)/2 + Izz3*(thetadot1 + thetadot2 +
thetadot3)*(thetadot1/2 + thetadot2/2 + thetadot3/2) - cz1*gravity*m1 - cz3*gravity*m2
- cz3*gravity*m3 + (m1*thetadot1^2*(cy1*cos(theta1) + cx1*sin(theta1))^2)/2 +
(m1*thetadot1^2*(cx1*cos(theta1) - cy1*sin(theta1))^2)/2 +
(m2*thetadot1^2*(cy3*cos(theta1) + cx3*sin(theta1))^2)/2 +
(m2*thetadot1^2*(cx3*cos(theta1) - cy3*sin(theta1))^2)/2 +
(m3*thetadot1^2*(cy3*cos(theta1) + cx3*sin(theta1))^2)/2 +
(m3*thetadot1^2*(cx3*cos(theta1) - cy3*sin(theta1))^2)/2 + Izz2*(thetadot1/2 +
thetadot2/2)*(thetadot1 + thetadot2) + gravity*m4*(a3*(cos(theta2)*sin(theta3) +
cos(theta3)*sin(theta2)) + d4*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) +
a2*sin(theta2))




tau =



 (Izz1 + Izz2 + Izz3 + (a2^2*m4)/2 + (a3^2*m4)/2 + cx1^2*m1 + cx3^2*m2 + cx3^2*m3 +
cy1^2*m1 + cy3^2*m2 + cy3^2*m3 + d3^2*m4 + (d4^2*m4)/2 + (a2^2*m4*cos(2*theta2))/2 +
(a3^2*m4*cos(2*theta2 + 2*theta3))/2 - (d4^2*m4*cos(2*theta2 + 2*theta3))/2 -
```

```
a3*d4*m4*sin(2*theta2 + 2*theta3) + a2*a3*m4*cos(theta3) - a2*d4*m4*sin(theta3) +
a2*a3*m4*cos(2*theta2 + theta3) - a2*d4*m4*sin(2*theta2 + theta3))*thetadotdot1 +
((d4*m4*cos(theta1 - theta2 - theta3))/2 - cy3*m2*cos(theta1) - cy3*m3*cos(theta1) -
d3*m4*cos(theta1) - cy1*m1*cos(theta1) - cx1*m1*sin(theta1) - cx3*m2*sin(theta1) -
cx3*m3*sin(theta1) - (a3*m4*sin(theta1 - theta2 - theta3))/2 - (a2*m4*sin(theta1 -
theta2))/2 - (d4*m4*cos(theta1 + theta2 + theta3))/2 - (a3*m4*sin(theta1 + theta2 +
theta3))/2 - (a2*m4*sin(theta1 + theta2))/2)*gravity

 (Izz1 + Izz2 + Izz3 + (a2^2*m4)/2 + (a3^2*m4)/2 + cx1^2*m1 + cx3^2*m2 + cx3^2*m3 +
cy1^2*m1 + cy3^2*m2 + cy3^2*m3 + d3^2*m4 + (d4^2*m4)/2 + (a2^2*m4*cos(2*theta2))/2 +
(a3^2*m4*cos(2*theta2 + 2*theta3))/2 - (d4^2*m4*cos(2*theta2 + 2*theta3))/2 -
a3*d4*m4*sin(2*theta2 + 2*theta3) + a2*a3*m4*cos(theta3) - a2*d4*m4*sin(theta3) +
a2*a3*m4*cos(2*theta2 + theta3) - a2*d4*m4*sin(2*theta2 + theta3))*thetadotdot1 +
(cx1*m1*cos(theta1) + cx3*m2*cos(theta1) + cx3*m3*cos(theta1) + (a3*m4*cos(theta1 -
theta2 - theta3))/2 - cy1*m1*sin(theta1) - cy3*m2*sin(theta1) - cy3*m3*sin(theta1) -
d3*m4*sin(theta1) + (d4*m4*sin(theta1 - theta2 - theta3))/2 + (a2*m4*cos(theta1 -
theta2))/2 + (a3*m4*cos(theta1 + theta2 + theta3))/2 - (d4*m4*sin(theta1 + theta2 +
theta3))/2 + (a2*m4*cos(theta1 + theta2))/2)*gravity


(Izz1 + Izz2 + Izz3 + (a2^2*m4)/2 + (a3^2*m4)/2 + cx1^2*m1 + cx3^2*m2 + cx3^2*m3 +
cy1^2*m1 + cy3^2*m2 + cy3^2*m3 + d3^2*m4 + (d4^2*m4)/2 + (a2^2*m4*cos(2*theta2))/2 +
(a3^2*m4*cos(2*theta2 + 2*theta3))/2 - (d4^2*m4*cos(2*theta2 + 2*theta3))/2 -
a3*d4*m4*sin(2*theta2 + 2*theta3) + a2*a3*m4*cos(theta3) - a2*d4*m4*sin(theta3) +
a2*a3*m4*cos(2*theta2 + theta3) - a2*d4*m4*sin(2*theta2 + theta3))*thetadotdot1


(m4*a2^2 + 2*m4*cos(theta3)*a2*a3 - 2*m4*sin(theta3)*a2*d4 + m4*a3^2 + m4*d4^2 + Izz2
+ Izz3)*thetadotdot2 + ((m4*sin(2*theta2)*a2^2)/2 + m4*sin(2*theta2 + theta3)*a2*a3 +
d4*m4*cos(2*theta2 + theta3)*a2 + (m4*sin(2*theta2 + 2*theta3)*a3^2)/2 +
d4*m4*cos(2*theta2 + 2*theta3)*a3 - (d4^2*m4*sin(2*theta2 + 2*theta3))/2)*thetadot1^2
+ ((a3*m4*sin(theta1 - theta2 - theta3))/2 - (d4*m4*cos(theta1 - theta2 - theta3))/2 +
(a2*m4*sin(theta1 - theta2))/2 - (d4*m4*cos(theta1 + theta2 + theta3))/2 -
(a3*m4*sin(theta1 + theta2 + theta3))/2 - (a2*m4*sin(theta1 + theta2))/2)*gravity +
d3*d4*m4*thetadot1*thetadot3*sin(theta2 + theta3) -
a3*d3*m4*thetadot1*thetadot3*cos(theta2 + theta3)


(m4*a2^2 + 2*m4*cos(theta3)*a2*a3 - 2*m4*sin(theta3)*a2*d4 + m4*a3^2 + m4*d4^2 + Izz2
+ Izz3)*thetadotdot2 + ((m4*sin(2*theta2)*a2^2)/2 + m4*sin(2*theta2 + theta3)*a2*a3 +
d4*m4*cos(2*theta2 + theta3)*a2 + (m4*sin(2*theta2 + 2*theta3)*a3^2)/2 +
d4*m4*cos(2*theta2 + 2*theta3)*a3 - (d4^2*m4*sin(2*theta2 + 2*theta3))/2)*thetadot1^2
+ ((a3*m4*cos(theta1 + theta2 + theta3))/2 - (d4*m4*sin(theta1 - theta2 - theta3))/2 -
(a2*m4*cos(theta1 - theta2))/2 - (a3*m4*cos(theta1 - theta2 - theta3))/2 -
(d4*m4*sin(theta1 + theta2 + theta3))/2 + (a2*m4*cos(theta1 + theta2))/2)*gravity +
d3*d4*m4*thetadot1*thetadot3*sin(theta2 + theta3) -
a3*d3*m4*thetadot1*thetadot3*cos(theta2 + theta3)


(m4*a2^2 + 2*m4*cos(theta3)*a2*a3 - 2*m4*sin(theta3)*a2*d4 + m4*a3^2 + m4*d4^2 + Izz2
+ Izz3)*thetadotdot2 + ((m4*sin(2*theta2)*a2^2)/2 + m4*sin(2*theta2 + theta3)*a2*a3 +
d4*m4*cos(2*theta2 + theta3)*a2 + (m4*sin(2*theta2 + 2*theta3)*a3^2)/2 +
```

```
d4*m4*cos(2*theta2 + 2*theta3)*a3 - (d4^2*m4*sin(2*theta2 + 2*theta3))/2)*thetadot1^2
+ (d4*m4*sin(theta2 + theta3) - a3*m4*cos(theta2 + theta3) -
a2*m4*cos(theta2))*gravity + d3*d4*m4*thetadot1*thetadot3*sin(theta2 + theta3) -
a3*d3*m4*thetadot1*thetadot3*cos(theta2 + theta3)


(m4*a3^2 + m4*d4^2 + Izz3)*thetadotdot3 + ((a3^2*m4*sin(2*theta2 + 2*theta3))/2 -
(d4^2*m4*sin(2*theta2 + 2*theta3))/2 + (a2*d4*m4*cos(theta3))/2 +
(a2*a3*m4*sin(theta3))/2 + (a2*d4*m4*cos(2*theta2 + theta3))/2 +
(a2*a3*m4*sin(2*theta2 + theta3))/2 + a3*d4*m4*cos(2*theta2 + 2*theta3))*thetadot1^2 +
(a2*d4*m4*cos(theta3) + a2*a3*m4*sin(theta3))*thetadot2^2 + ((a3*m4*sin(theta1 -
theta2 - theta3))/2 - (d4*m4*cos(theta1 - theta2 - theta3))/2 - (d4*m4*cos(theta1 +
theta2 + theta3))/2 - (a3*m4*sin(theta1 + theta2 + theta3))/2)*gravity +
d3*d4*m4*thetadot1*thetadot2*sin(theta2 + theta3) -
a3*d3*m4*thetadot1*thetadot2*cos(theta2 + theta3)


(m4*a3^2 + m4*d4^2 + Izz3)*thetadotdot3 + ((a3^2*m4*sin(2*theta2 + 2*theta3))/2 -
(d4^2*m4*sin(2*theta2 + 2*theta3))/2 + (a2*d4*m4*cos(theta3))/2 +
(a2*a3*m4*sin(theta3))/2 + (a2*d4*m4*cos(2*theta2 + theta3))/2 +
(a2*a3*m4*sin(2*theta2 + theta3))/2 + a3*d4*m4*cos(2*theta2 + 2*theta3))*thetadot1^2 +
(a2*d4*m4*cos(theta3) + a2*a3*m4*sin(theta3))*thetadot2^2 + ((a3*m4*cos(theta1 +
theta2 + theta3))/2 - (d4*m4*sin(theta1 - theta2 - theta3))/2 - (a3*m4*cos(theta1 -
theta2 - theta3))/2 - (d4*m4*sin(theta1 + theta2 + theta3))/2)*gravity +
d3*d4*m4*thetadot1*thetadot2*sin(theta2 + theta3) -
a3*d3*m4*thetadot1*thetadot2*cos(theta2 + theta3)


(m4*a3^2 + m4*d4^2 + Izz3)*thetadotdot3 + ((a3^2*m4*sin(2*theta2 + 2*theta3))/2 -
(d4^2*m4*sin(2*theta2 + 2*theta3))/2 + (a2*d4*m4*cos(theta3))/2 +
(a2*a3*m4*sin(theta3))/2 + (a2*d4*m4*cos(2*theta2 + theta3))/2 +
(a2*a3*m4*sin(2*theta2 + theta3))/2 + a3*d4*m4*cos(2*theta2 + 2*theta3))*thetadot1^2 +
(a2*d4*m4*cos(theta3) + a2*a3*m4*sin(theta3))*thetadot2^2 + (d4*m4*sin(theta2 +
theta3) - a3*m4*cos(theta2 + theta3))*gravity +
d3*d4*m4*thetadot1*thetadot2*sin(theta2 + theta3) -
a3*d3*m4*thetadot1*thetadot2*cos(theta2 + theta3)
```

Published with MATLAB® R2015a

Trajectory Generation_Tool Space :

All steps explained in the code.

```
%Final 263D
%Trajectory Generation Tool Space
%Sepehr Ghassemi


%Initial Clearing
clc
clear all
close all

%syms Defined

syms alpha1 alpha2 alpha3 alpha4 real
syms a1 a2 a3 a4 real
syms d1 d2 d3 d4 real
syms theta1 theta2 theta3 theta4 theta5 theta6 theta7 real
syms thetadot1 thetadot2 thetadot3 thetadot4 real
syms thetadotdot1 thetadotdot2 thetadotdot3 thetadotdot4 real

% %modified DH Table with tool
% % this is identical to the standard version however the ikine6s function
% % and other fuctons are only available for standard DH.
% clear L
% %              theta  d        a    alpha
% L(1) = Link([  0      0        0       0       0], 'modified');
% L(2) = Link([  0      0        0      -pi/2    0], 'modified');
% L(3) = Link([  0      0.1254   0.4318  0       0], 'modified');
% L(4) = Link([  0      0.4318   0.0191 -pi/2    0], 'modified');
% L(5) = Link([  0      0        0       pi/2    0], 'modified');
% L(6) = Link([  0      0        0      -pi/2    0], 'modified');
% tool=transl(-0.1,0,0.13625);

%standard DH Table with tool

clear L
%              theta       d        a    alpha
L(1) = Link([  0      0        0         -pi/2       0]);
L(2) = Link([  0      0        0.4318     0          0]);
L(3) = Link([  0      0.1254   0.0191    -pi/2       0]);
L(4) = Link([  0      0.4318   0          pi/2       0]);
L(5) = Link([  0      0        0         -pi/2       0]);
L(6) = Link([  0      0        0          0          0]);
tool=transl(-0.1,0,0.13625);
```

```
pum = SerialLink([L(1) L(2) L(3) L(4) L(5) L(6)],'tool',tool)
theta1=0;
theta2=0;
theta3=0;
theta4=0;
theta5=0;
theta6=0;
angles=[theta1 theta2 theta3 theta4 theta5 theta6];
%x=pum.fkine(angles)
%y=pum.ikine(x)
%pum.plot(angles)
pum.plot(angles)


L=0.8636;
segment1=0.15*L;
segment2=0.15*L;
segment3=pi*0.15*L;
segment4=0.15*L;
segment5=0.15*L;
totaldistance=segment1+segment2+segment3+segment4+segment5;
totaltime=20;
totalsteps=220; %chosen with a sample rate of around 10 hertz
%(we have 100 hertz capability to be in range. but the lower the better for
%this scenario.(not very low though)

averagevelocity=totaldistance/totaltime;
%chosen initial conditions
drawingplainz=0;
startingx=0.6000;
startingy=0.1254;
startingz=drawingplainz;

start=[startingx;startingy;startingz];

%viapoint distance for all except curve
viasize=0.05*L;
ymove=[0;1;0];
ymoveneg=[0;-1;0];
xmove=[1;0;0];
xmoveneg=[-1;0;0];

%chosen via points
%segment1
via1=start+ymove*viasize;
via2=via1+ymove*viasize;
via3=via2+ymove*viasize;
%segment2
via4=via3+xmove*viasize;
via5=via4+xmove*viasize;
via6=via5+xmove*viasize;
%sement3
%we devide the semicircle into 10 sections
```

```
via7=via6+(0.15*L*[cos(pi*4/10);-(1-sin(pi*4/10));0]);
via8=via6+(0.15*L*[cos(pi*3/10);-(1-sin(pi*3/10));0]);
via9=via6+(0.15*L*[cos(pi*2/10);-(1-sin(pi*2/10));0]);
via10=via6+(0.15*L*[cos(pi*1/10);-(1-sin(pi*1/10));0]);
via11=via6+(0.15*L*[1;-1;0]);
via12=via6+(0.15*L*[cos(pi*1/10);-(1+sin(pi*1/10));0]);
via13=via6+(0.15*L*[cos(pi*2/10);-(1+sin(pi*2/10));0]);
via14=via6+(0.15*L*[cos(pi*3/10);-(1+sin(pi*3/10));0]);
via15=via6+(0.15*L*[cos(pi*4/10);-(1+sin(pi*4/10));0]);
via16=via6+(0.15*L*[0;-2;0]);
%segment4
via17=via16+xmoveneg*viasize;
via18=via17+xmoveneg*viasize;
via19=via18+xmoveneg*viasize;
%segment5
via20=via19+ymove*viasize;
via21=via20+ymove*viasize;
via22=start;

allvias=[start via1 via2 via3 via4 via5 via6 via7 via8 via9 via10 via11 via12 via13
via14 via15 via16 via17 via18 via19 via20 via21 via22];
allviax=allvias(1,1:23);
allviay=allvias(2,1:23);

numviapoints=22;
totalsteps=220;
stepspervia=10;

%repeat operation for the 220 steps

%chosen initial conditions
drawingplainz=0;
startingx=0.6000;
startingy=0.1254;
startingz=drawingplainz;

start=[startingx;startingy;startingz];
nstep=220; %we are choosing 220 here
%viapoint distance for all except curve
viasize=(0.15/(nstep*3/22))*L;
ymove=[0;1;0];
ymoveneg=[0;-1;0];
xmove=[1;0;0];
xmoveneg=[-1;0;0];
il=nstep*3/22;

%chosen via points
count=1;
%segment1
vvia(:,1)=start;
for i=1:1:il
count=count+1;
vvia(:,count)=vvia(:,(count-1))+ymove*viasize;
```

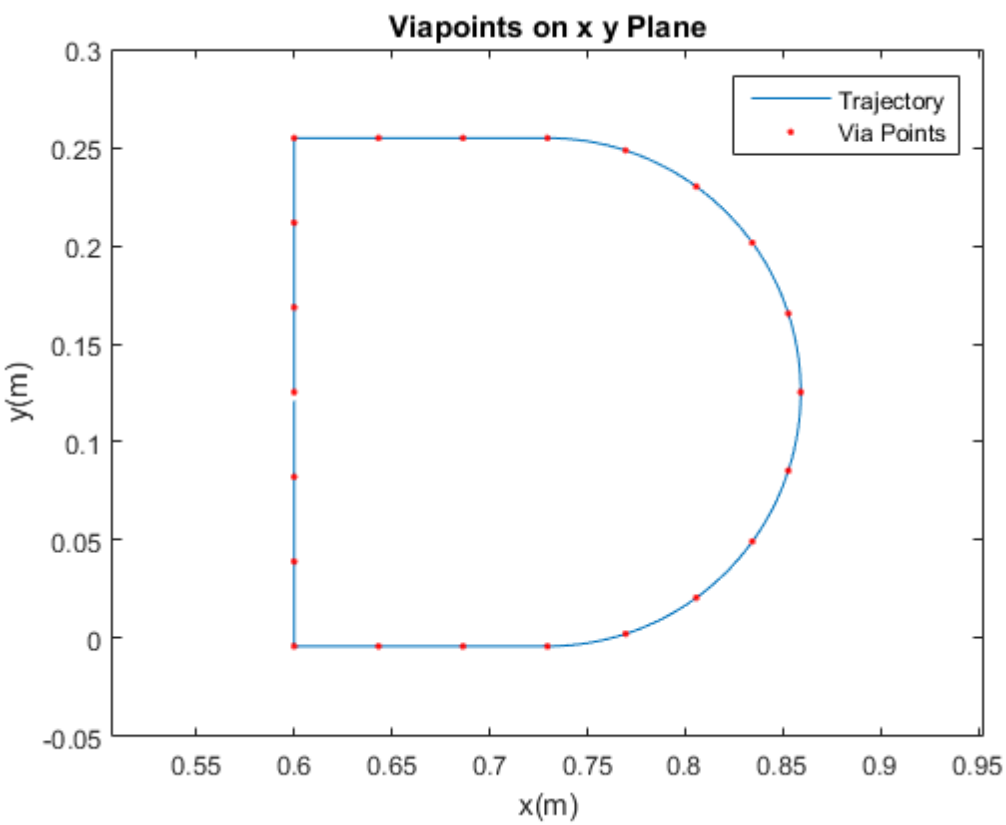
```
end
%segment2
for i=1:1:il
count=count+1;
vvia(:,count)=vvia(:,(count-1))+xmove*viasize;
end
vcount=count;
%sement3
%we devide the semicircle into 10 sections
for i=1:1:(nstep*5/22)
count=count+1;
vvia(:,count)=vvia(:,(vcount))+(0.15*L*[cos(pi*((nstep*5/22)-i)/(nstep*10/22));-(1-
sin(pi*((nstep*5/22)-(i))/(nstep*10/22)));0]);
end
for i=1:1:(nstep*5/22)
count=count+1;
vvia(:,count)=vvia(:,(vcount))+(0.15*L*[cos(pi*i/(nstep*10/22));-
(1+sin(pi*i/(nstep*10/22)));0]);
end
%segment4
for i=1:1:il
count=count+1;
vvia(:,count)=vvia(:,(count-1))+xmoveneg*viasize;
end
%segment5
for i=1:1:il
count=count+1;
vvia(:,count)=vvia(:,(count-1))+ymove*viasize;
end

sallviax=vvia(1,1:nstep);
sallviay=vvia(2,1:nstep);

%plotting the viapoints
plot(sallviax,sallviay);axis equal;hold on;
plot(allviax',allviay','r.');axis equal;
xlabel('x(m)');
ylabel('y(m)');
ylim([-0.05 0.3]);
title('Viapoints on x y Plane');
legend('Trajectory','Via Points');

%taking the joint position at the main via points (for delayissues a for
%loop was very inefficient while running (computer very slow. had to tweak
%some values for better covergence)

A1=wrapToPi(pum.ikine([0 0 1 (vvia(1,10));0 -1 0 (vvia(2,10));1 0 0 (vvia(3,10));0 0 0
1]));
A2=wrapToPi(pum.ikine([0 0 1 (vvia(1,22));0 -1 0 (vvia(2,22));1 0 0 (vvia(3,22));0 0 0
1]));
A3=wrapToPi(pum.ikine([0 0 1 (vvia(1,32));0 -1 0 (vvia(2,32));1 0 0 (vvia(3,32));0 0 0
1]));
```

```
A4=wrapToPi(pum.ikine([0 0 1 (vvia(1,40));0 -1 0 (vvia(2,40));1 0 0 (vvia(3,40));0 0 0
1]));
A5=wrapToPi(pum.ikine([0 0 1 (vvia(1,50));0 -1 0 (vvia(2,50));1 0 0 (vvia(3,50));0 0 0
1]));
A6=wrapToPi(pum.ikine([0 0 1 (vvia(1,60));0 -1 0 (vvia(2,60));1 0 0 (vvia(3,60));0 0 0
1]));
A7=wrapToPi(pum.ikine([0 0 1 (vvia(1,70));0 -1 0 (vvia(2,70));1 0 0 (vvia(3,70));0 0 0
1]));
A8=wrapToPi(pum.ikine([0 0 1 (vvia(1,80));0 -1 0 (vvia(2,80));1 0 0 (vvia(3,80));0 0 0
1]));
A9=wrapToPi(pum.ikine([0 0 1 (vvia(1,90));0 -1 0 (vvia(2,90));1 0 0 (vvia(3,90));0 0 0
1]));
A10=wrapToPi(pum.ikine([0 0 1 (vvia(1,100));0 -1 0 (vvia(2,100));1 0 0 (vvia(3,100));0
0 0 1]));
A11=wrapToPi(pum.ikine([0 0 1 (vvia(1,110));0 -1 0 (vvia(2,110));1 0 0 (vvia(3,110));0
0 0 1]));
A12=wrapToPi(pum.ikine([0 0 1 (vvia(1,120));0 -1 0 (vvia(2,120));1 0 0 (vvia(3,120));0
0 0 1]));
A13=wrapToPi(pum.ikine([0 0 1 (vvia(1,130));0 -1 0 (vvia(2,130));1 0 0 (vvia(3,130));0
0 0 1]));
A14=wrapToPi(pum.ikine([0 0 1 (vvia(1,140));0 -1 0 (vvia(2,140));1 0 0 (vvia(3,140));0
0 0 1]));
A15=wrapToPi(pum.ikine([0 0 1 (vvia(1,150));0 -1 0 (vvia(2,150));1 0 0 (vvia(3,150));0
0 0 1]));
A16=wrapToPi(pum.ikine([0 0 1 (vvia(1,160));0 -1 0 (vvia(2,160));1 0 0 (vvia(3,160));0
0 0 1]));
A17=wrapToPi(pum.ikine([0 0 1 (vvia(1,170));0 -1 0 (vvia(2,170));1 0 0 (vvia(3,170));0
0 0 1]));
A18=wrapToPi(pum.ikine([0 0 1 (vvia(1,180));0 -1 0 (vvia(2,180));1 0 0 (vvia(3,180));0
0 0 1]));
A19=wrapToPi(pum.ikine([0 0 1 (vvia(1,192));0 -1 0 (vvia(2,192));1 0 0 (vvia(3,192));0
0 0 1]));
A20=wrapToPi(pum.ikine([0 0 1 (vvia(1,200));0 -1 0 (vvia(2,200));1 0 0 (vvia(3,200));0
0 0 1]));
A21=wrapToPi(pum.ikine([0 0 1 (vvia(1,210));0 -1 0 (vvia(2,210));1 0 0 (vvia(3,210));0
0 0 1]));

cn=1;
theta1arr=[A1(:,cn);A2(:,cn);A3(:,cn);A4(:,cn);A5(:,cn);A6(:,cn);A7(:,cn);A8(:,cn);A9(
:,cn);A10(:,cn);A11(:,cn);A12(:,cn);A13(:,cn);A14(:,cn);A15(:,cn);A16(:,cn);A17(:,cn);
A18(:,cn);A19(:,cn);A20(:,cn);A21(:,cn)];
cn=2;
theta2arr=[A1(:,cn);A2(:,cn);A3(:,cn);A4(:,cn);A5(:,cn);A6(:,cn);A7(:,cn);A8(:,cn);A9(
:,cn);A10(:,cn);A11(:,cn);A12(:,cn);A13(:,cn);A14(:,cn);A15(:,cn);A16(:,cn);A17(:,cn);
A18(:,cn);A19(:,cn);A20(:,cn);A21(:,cn)];
cn=3;
theta3arr=[A1(:,cn);A2(:,cn);A3(:,cn);A4(:,cn);A5(:,cn);A6(:,cn);A7(:,cn);A8(:,cn);A9(
:,cn);A10(:,cn);A11(:,cn);A12(:,cn);A13(:,cn);A14(:,cn);A15(:,cn);A16(:,cn);A17(:,cn);
A18(:,cn);A19(:,cn);A20(:,cn);A21(:,cn)];
cn=4;
theta4arr=[A1(:,cn);A2(:,cn);A3(:,cn);A4(:,cn);A5(:,cn);A6(:,cn);A7(:,cn);A8(:,cn);A9(
:,cn);A10(:,cn);A11(:,cn);A12(:,cn);A13(:,cn);A14(:,cn);A15(:,cn);A16(:,cn);A17(:,cn);
A18(:,cn);A19(:,cn);A20(:,cn);A21(:,cn)];
```

```
cn=5;
theta5arr=[A1(:,cn);A2(:,cn);A3(:,cn);A4(:,cn);A5(:,cn);A6(:,cn);A7(:,cn);A8(:,cn);A9(
:,cn);A10(:,cn);A11(:,cn);A12(:,cn);A13(:,cn);A14(:,cn);A15(:,cn);A16(:,cn);A17(:,cn);
A18(:,cn);A19(:,cn);A20(:,cn);A21(:,cn)];
cn=6;
theta6arr=[A1(:,cn);A2(:,cn);A3(:,cn);A4(:,cn);A5(:,cn);A6(:,cn);A7(:,cn);A8(:,cn);A9(
:,cn);A10(:,cn);A11(:,cn);A12(:,cn);A13(:,cn);A14(:,cn);A15(:,cn);A16(:,cn);A17(:,cn);
A18(:,cn);A19(:,cn);A20(:,cn);A21(:,cn)];

t=[0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20];

%Extra Credit
%plotting the animation of arm at different poses during the operation
% qnt=50;
% T=pum.ikine([0 0 1 (via(1,qnt));0 -1 0 (via(2,qnt));1 0 0 (via(3,qnt));0 0 0 1]);

% figure(3)
% plot(allviax',allviay','r.');
% pum.plot(T);hold on;
```

```
pum =


noname (6 axis, RRRRRR, stdDH, fastRNE)


+---+-----------+-----------+-----------+-----------+-----------+

| j |     theta |         d |         a |     alpha |    offset |

+---+-----------+-----------+-----------+-----------+-----------+

|  1|         q1|          0|          0|     -1.571|          0|

|  2|         q2|          0|     0.4318|          0|          0|

|  3|         q3|     0.1254|     0.0191|     -1.571|          0|

|  4|         q4|     0.4318|          0|      1.571|          0|

|  5|         q5|          0|          0|     -1.571|          0|

|  6|         q6|          0|          0|          0|          0|

+---+-----------+-----------+-----------+-----------+-----------+
```

```
grav =     0  base = 1   0   0   0   tool =  1           0           0        -0.1

           0          0   1   0   0           0           1           0           0

        9.81          0   0   1   0           0           0           1     0.13625

                      0   0   0   1           0           0           0           1
```



Published with MATLAB® R2015a

Trajectory Generation_Joint Space :

All steps explained in the code. Including some explanations.

```
%Final 263D
%Trajectory Generation Joint Space
%Sepehr Ghassemi


%Initial Clearing
clc
clear all
close all

%syms Defined

syms alpha1 alpha2 alpha3 alpha4 real
syms a1 a2 a3 a4 real
syms d1 d2 d3 d4 real
syms theta1 theta2 theta3 theta4 theta5 theta6 theta7 real
syms thetadot1 thetadot2 thetadot3 thetadot4 real
syms thetadotdot1 thetadotdot2 thetadotdot3 thetadotdot4 real

% %modified DH Table with tool
% % this is identical to the standard version however the ikine6s function
% % and other fuctons are only available for standard DH.
% clear L
% %              theta  d        a    alpha
% L(1) = Link([  0      0        0       0       0], 'modified');
% L(2) = Link([  0      0        0      -pi/2    0], 'modified');
% L(3) = Link([  0      0.1254   0.4318  0       0], 'modified');
% L(4) = Link([  0      0.4318   0.0191 -pi/2    0], 'modified');
% L(5) = Link([  0      0        0       pi/2    0], 'modified');
% L(6) = Link([  0      0        0      -pi/2    0], 'modified');
% tool=transl(-0.1,0,0.13625);

%standard DH Table with tool

clear L
%              theta       d        a    alpha
L(1) = Link([  0      0        0         -pi/2       0]);
L(2) = Link([  0      0        0.4318     0          0]);
L(3) = Link([  0      0.1254   0.0191    -pi/2       0]);
L(4) = Link([  0      0.4318   0          pi/2       0]);
L(5) = Link([  0      0        0         -pi/2       0]);
L(6) = Link([  0      0        0          0          0]);
tool=transl(-0.1,0,0.13625);

pum = SerialLink([L(1) L(2) L(3) L(4) L(5) L(6)],'tool',tool)
theta1=0;
theta2=0;
theta3=0;
```

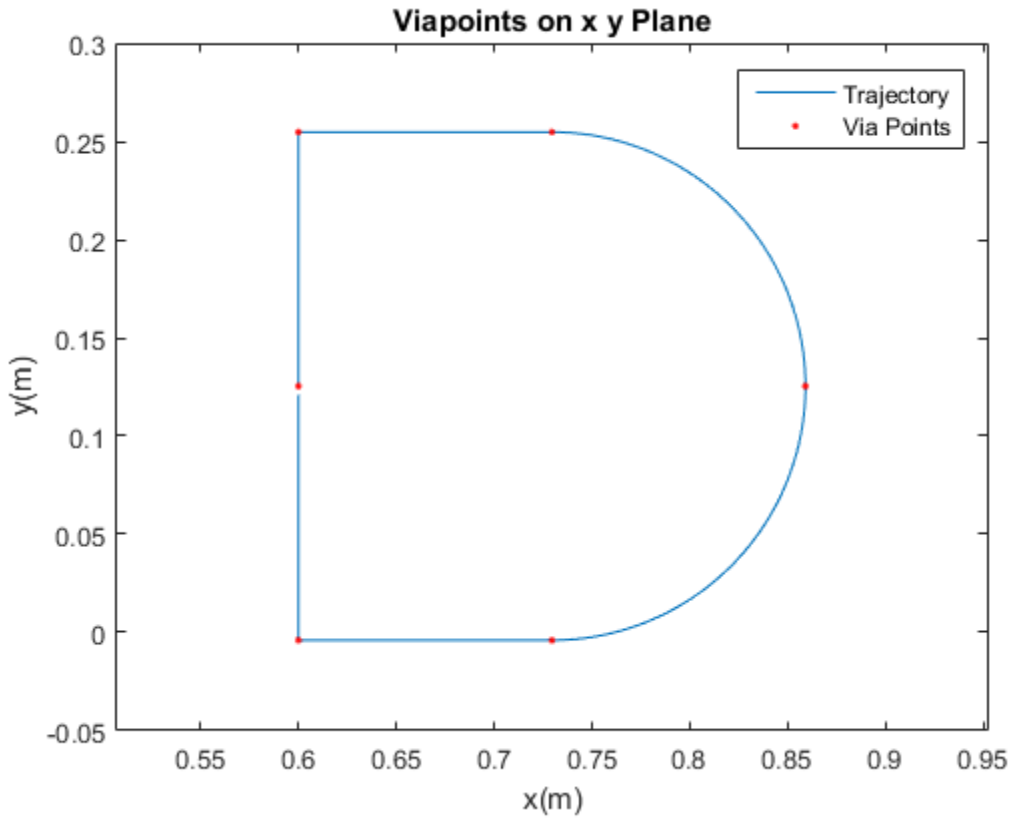
```
theta4=0;
theta5=0;
theta6=0;
angles=[theta1 theta2 theta3 theta4 theta5 theta6];
%x=pum.fkine(angles)
%y=pum.ikine(x)
%pum.plot(angles)
pum.plot(angles)


L=0.8636;
segment1=0.15*L;
segment2=0.15*L;
segment3=pi*0.15*L;
segment4=0.15*L;
segment5=0.15*L;
totaldistance=segment1+segment2+segment3+segment4+segment5;
totaltime=20;
totalsteps=220; %chosen with a sample rate of around 10 hertz
%(we have 100 hertz capability to be in range. but the lower the better for
%this scenario.(not very low though)

averagevelocity=totaldistance/totaltime;
%chosen initial conditions
drawingplainz=0;
startingx=0.6000;
startingy=0.1254;
startingz=drawingplainz;

start=[startingx;startingy;startingz];

%viapoint distance for all except curve
viasize=0.05*L;
ymove=[0;1;0];
ymoveneg=[0;-1;0];
xmove=[1;0;0];
xmoveneg=[-1;0;0];


% NOTE that this is based of my previous code (for tool space) and thus it has many
% redundancies
%chosen via points
%segment1
via1=start+ymove*viasize;
via2=via1+ymove*viasize;
via3=via2+ymove*viasize;
%segment2
via4=via3+xmove*viasize;
via5=via4+xmove*viasize;
via6=via5+xmove*viasize;
%sement3
%we devide the semicircle into 10 sections
via7=via6+(0.15*L*[cos(pi*4/10);-(1-sin(pi*4/10));0]);
```

```
via8=via6+(0.15*L*[cos(pi*3/10);-(1-sin(pi*3/10));0]);
via9=via6+(0.15*L*[cos(pi*2/10);-(1-sin(pi*2/10));0]);
via10=via6+(0.15*L*[cos(pi*1/10);-(1-sin(pi*1/10));0]);
via11=via6+(0.15*L*[1;-1;0]);
via12=via6+(0.15*L*[cos(pi*1/10);-(1+sin(pi*1/10));0]);
via13=via6+(0.15*L*[cos(pi*2/10);-(1+sin(pi*2/10));0]);
via14=via6+(0.15*L*[cos(pi*3/10);-(1+sin(pi*3/10));0]);
via15=via6+(0.15*L*[cos(pi*4/10);-(1+sin(pi*4/10));0]);
via16=via6+(0.15*L*[0;-2;0]);
%segment4
via17=via16+xmoveneg*viasize;
via18=via17+xmoveneg*viasize;
via19=via18+xmoveneg*viasize;
%segment5
via20=via19+ymove*viasize;
via21=via20+ymove*viasize;
via22=start;

%mainviapoints are via 3,6,11,16,19,22

allvias=[via3 via6 via11 via16 via19 via22];
allviax=allvias(1,1:6);
allviay=allvias(2,1:6);

numviapoints=6;


%********** toolspace work shown below is a refrence to show the path
%chosen initial conditions
drawingplainz=0;
startingx=0.6000;
startingy=0.1254;
startingz=drawingplainz;

start=[startingx;startingy;startingz];
nstep=220; %we are choosing 220 here
%viapoint distance for all except curve
viasize=(0.15/(nstep*3/22))*L;
ymove=[0;1;0];
ymoveneg=[0;-1;0];
xmove=[1;0;0];
xmoveneg=[-1;0;0];
il=nstep*3/22;

%chosen via points
count=1;
%segment1
vvia(:,1)=start;
for i=1:1:il
count=count+1;
vvia(:,count)=vvia(:,(count-1))+ymove*viasize;
end
%segment2
```

```
for i=1:1:il
count=count+1;
vvia(:,count)=vvia(:,(count-1))+xmove*viasize;
end
vcount=count;
%sement3
%we devide the semicircle into 10 sections
for i=1:1:(nstep*5/22)
count=count+1;
vvia(:,count)=vvia(:,(vcount))+(0.15*L*[cos(pi*((nstep*5/22)-i)/(nstep*10/22));-(1-
sin(pi*((nstep*5/22)-(i))/(nstep*10/22)));0]);
end
for i=1:1:(nstep*5/22)
count=count+1;
vvia(:,count)=vvia(:,(vcount))+(0.15*L*[cos(pi*i/(nstep*10/22));-
(1+sin(pi*i/(nstep*10/22)));0]);
end
%segment4
for i=1:1:il
count=count+1;
vvia(:,count)=vvia(:,(count-1))+xmoveneg*viasize;
end
%segment5
for i=1:1:il
count=count+1;
vvia(:,count)=vvia(:,(count-1))+ymove*viasize;
end

sallviax=vvia(1,1:nstep);
sallviay=vvia(2,1:nstep);
%**********

%plotting the viapoints
plot(sallviax,sallviay);axis equal;hold on;
plot(allviax',allviay','r.');axis equal;
xlabel('x(m)');
ylabel('y(m)');
ylim([-0.05 0.3]);
title('Viapoints on x y Plane');
legend('Trajectory','Via Points');

%taking the joint position at the main via points (for delayissues a for
%loop was very inefficient while running (computer very slow. had to tweak
%some values for better covergence)

A1=wrapToPi(pum.ikine([0 0 1 (vvia(1,10));0 -1 0 (vvia(2,10));1 0 0 (vvia(3,10));0 0 0
1]));
A3=wrapToPi(pum.ikine([0 0 1 (vvia(1,32));0 -1 0 (vvia(2,32));1 0 0 (vvia(3,32));0 0 0
1]));
A6=wrapToPi(pum.ikine([0 0 1 (vvia(1,60));0 -1 0 (vvia(2,60));1 0 0 (vvia(3,60));0 0 0
1]));
A11=wrapToPi(pum.ikine([0 0 1 (vvia(1,110));0 -1 0 (vvia(2,110));1 0 0 (vvia(3,110));0
0 0 1]));
```

```
A16=wrapToPi(pum.ikine([0 0 1 (vvia(1,160));0 -1 0 (vvia(2,160));1 0 0 (vvia(3,160));0
0 0 1]));
A19=wrapToPi(pum.ikine([0 0 1 (vvia(1,192));0 -1 0 (vvia(2,192));1 0 0 (vvia(3,192));0
0 0 1]));

cn=1;
theta1arr=[A1(:,cn);A3(:,cn);A6(:,cn);A11(:,cn);A16(:,cn);A19(:,cn)];
cn=2;
theta2arr=[A1(:,cn);A3(:,cn);A6(:,cn);A11(:,cn);A16(:,cn);A19(:,cn)];
cn=3;
theta3arr=[A1(:,cn);A3(:,cn);A6(:,cn);A11(:,cn);A16(:,cn);A19(:,cn)];
cn=4;
theta4arr=[A1(:,cn);A3(:,cn);A6(:,cn);A11(:,cn);A16(:,cn);A19(:,cn)];
cn=5;
theta5arr=[A1(:,cn);A3(:,cn);A6(:,cn);A11(:,cn);A16(:,cn);A19(:,cn)];
cn=6;
theta6arr=[A1(:,cn);A3(:,cn);A6(:,cn);A11(:,cn);A16(:,cn);A19(:,cn)];

t=[0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20];

%Linear interpolation from A1 to A6 and from A19 to A1
%Curve and polinomial from A6 to A11 and A11 to A16


%Now for the joint space work we start interpolating from the angles
%using polinomials along with linear interpolations


%then we can do fkine to check results and error.
%we use function (theta,thetadot,thetaf,thetafdot,t)

%both methods in the range we want

%both methods yield similar results.

%Extra Credit
% %plotting the animation of arm at different poses during the operation
% qnt=10;  %variable, see figures...
% T=pum.ikine([0 0 1 (via(1,qnt));0 -1 0 (via(2,qnt));1 0 0 (via(3,qnt));0 0 0 1]);
%
% figure(3)
% plot(allviax',allviay','r.');
% pum.plot(T);hold on;
```

```
pum =


noname (6 axis, RRRRRR, stdDH, fastRNE)


+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|     -1.571|          0|
|  2|         q2|          0|     0.4318|          0|          0|
|  3|         q3|     0.1254|     0.0191|     -1.571|          0|
|  4|         q4|     0.4318|          0|      1.571|          0|
|  5|         q5|          0|          0|     -1.571|          0|
|  6|         q6|          0|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+


grav =    0  base = 1  0  0  0   tool =  1           0           0        -0.1
          0         0  1  0  0           0           1           0           0
       9.81         0  0  1  0           0           0           1     0.13625
                    0  0  0  1           0           0           0           1
```

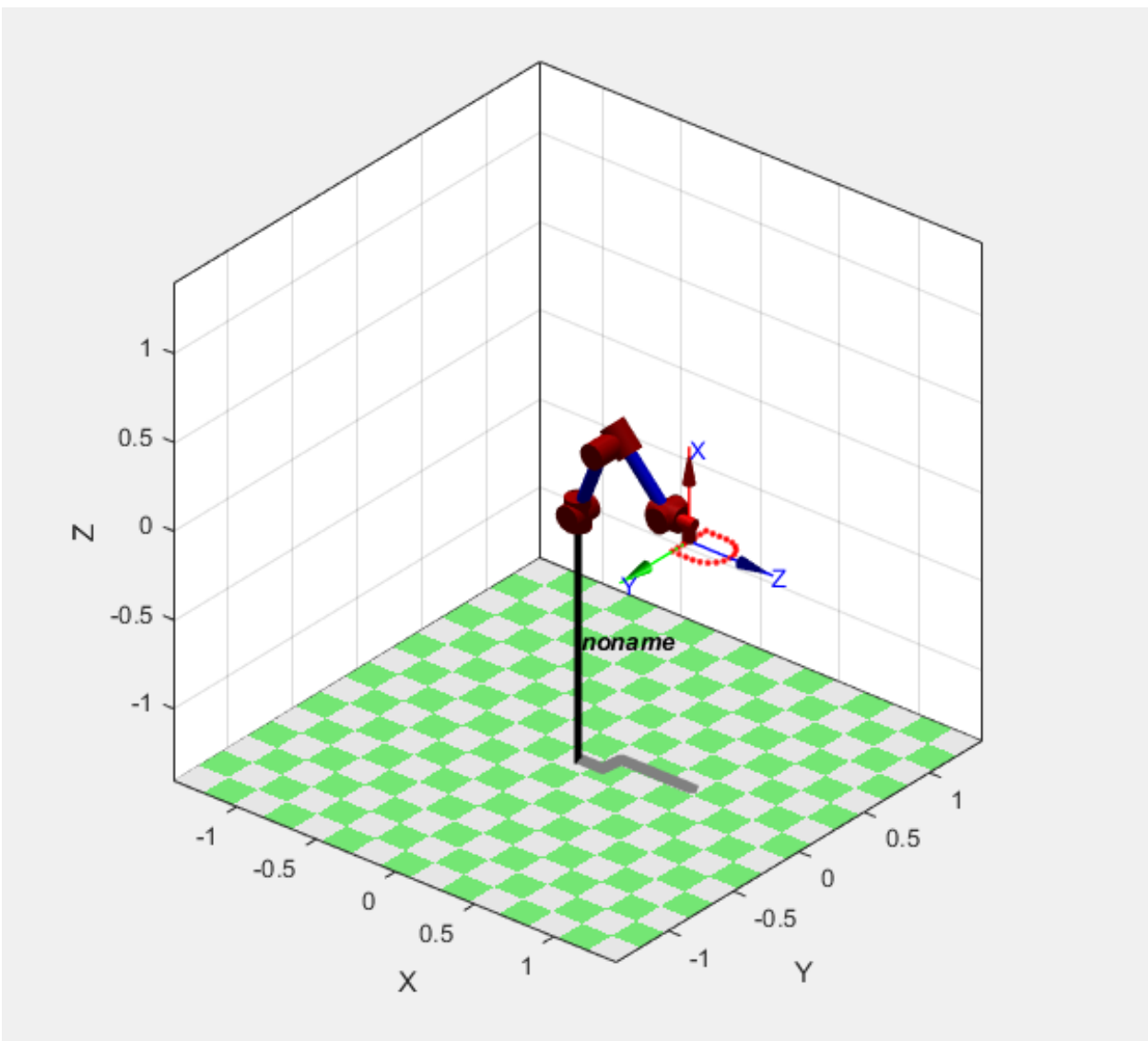
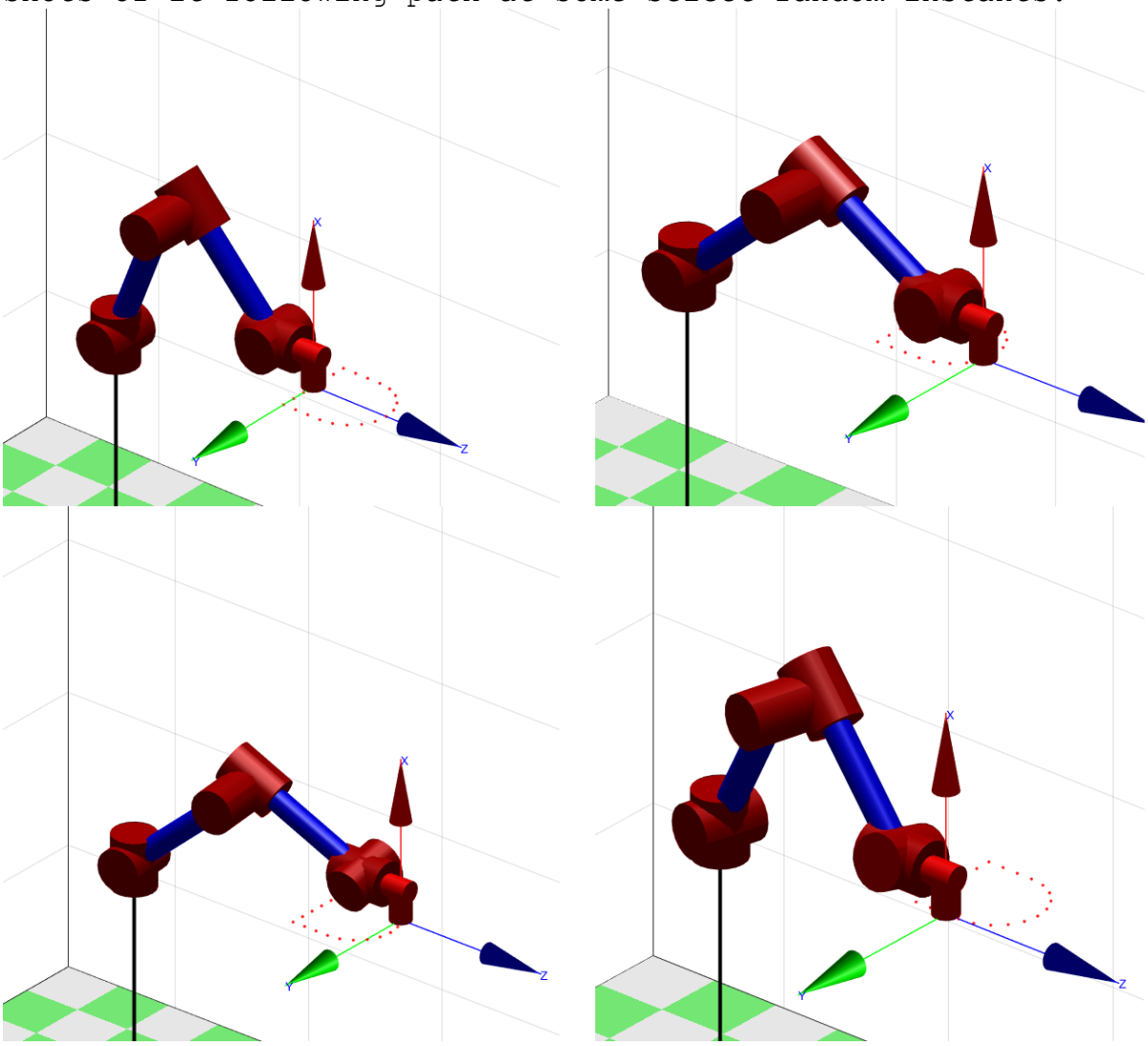


Published with MATLAB® R2015a

Simulation and following of path:

```
%plotting the animation of arm at different poses during the operation
qnt=1;    %a variable for instantaneous pose
T=pum.ikine([0 0 1 (vvia(1,qnt));0 -1 0 (vvia(2,qnt));1 0 0 (vvia(3,qnt));0 0
0 1]);

figure(3)
plot(allviax',allviay','r.');
pum.plot(T);hold on;
```

Overall look of graph:



Shots of it following path at some select random instants: