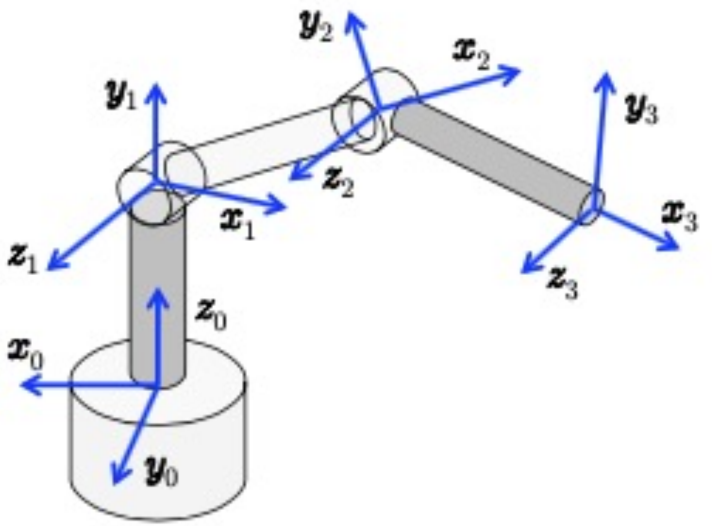
# Standard robot model

a file for the 3R spatial (elbow-type) robot made available in the repository




| $i$ | $\alpha_i$ | $d_i$ | $a_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | $\pi/2$ | $L_1$ | 0 | $q_1$ |
| 2 | 0 | 0 | $L_2$ | $q_2$ |
| 3 | 0 | 0 | $L_3$ | $q_3$ |

- center of masses on link axes
- diagonal link inertias
- gravity is present along $\boldsymbol{z}_0$
- same numerical data for kinematic and dynamic parameters to be used by all Short Projects referring to this model

```
% dynamic model of a spatial 3R elbow-type robot
% using a Lagrangian formulation in symbolic form
%
% A. De Luca
%
% distributed to the students of the
% pHRI module of the 2022-23 course on EiR/CPR
% on June 26, 2023

% assumptions:
% - frames assigned according to standard DH (see figure in Short Projects)
% - center of masses on link axes
% - diagonal link inertias (with Iyy=Izz (≠ Ixx) for links 2 and 3)
% - gravity is present along -z0
% numerical data at the end of the file

clear all
close all
clc

% kinematics (limited to rotation matrices used for angular velocity
% in recursive algorithm for kinetic energy computation, if used)

syms alpha d a theta real
syms L1 L2 L3 real

disp('**** dynamic model of 3R spatial elbow-type robot ****')
disp(' ')
disp('[press return to proceed at every pause (or comment them all)]')
```

```
pause

N=3;  % number of joints

DHTABLE = [ pi/2      0      sym('L1') sym('q1');
             0     sym('L2')    0      sym('q2');
             0     sym('L3')    0      sym('q3')];

TDH = [ cos(theta) -sin(theta)*cos(alpha)  sin(theta)*sin(alpha) a*cos(theta);
        sin(theta)  cos(theta)*cos(alpha) -cos(theta)*sin(alpha) a*sin(theta);
          0             sin(alpha)             cos(alpha)            d;
          0               0                       0                  1];

for i = 1:N
    alpha = DHTABLE(i,1);
    a = DHTABLE(i,2);
    d = DHTABLE(i,3);
    theta = DHTABLE(i,4);
    A{i} = subs(TDH);
end

disp(' ')
disp('* DH rotation matrices *')


R1=A{1}(1:3,1:3)
R2=A{2}(1:3,1:3)
R3=A{3}(1:3,1:3)
```

```
pause

% dynamics

% all symbolic variabel are defined as real
syms m1 m2 m3 dc1 dc2 dc3 real % m1 is unnecessary
syms I1zz I2xx I2yy I2xx I2yy I2zz I3xx I3yy I3zz real
% I1xx, I1yy are unnecessary; assumed I2zz=I2yy and I3zz=I3yy
syms q1 q2 q3 dq1 dq2 dq3 ddq1 ddq2 ddq3 u1 u2 u3 g0 real

% dynamic coefficients (in linear parametrization)
syms a1 a2 a3 a4 a5 a6 a7 a8 real

disp('* kinetic energy of link 1 *')

T1=(1/2)*I1zz*dq1^2

pause

disp('* kinetic energy of link 2 - linear part *')

pc2=[dc2*cos(q2)*cos(q1) dc2*cos(q2)*sin(q1) L1+dc2*sin(q2)]'
vc2=simplify(diff(pc2,q1)*dq1+diff(pc2,q2)*dq2)
T2c=(1/2)*simplify(m2*vc2'*vc2)

%pause

% initialization of recursion (when used)
% om0=0
```

```
% z0=[0 0 1]';
% om1=R1'*(om0+dq1*z0);

disp(['* kinetic energy of link 2 - angular part ' ...
     '(in local coordinates to frame 2) *'])

%om2=simplify(R2'*(om1+dq2*z0))
om2=[dq1*sin(q2) dq1*cos(q2) dq2]'
T2a=(1/2)*om2'*diag([I2xx I2yy I2yy])*om2

T2=simplify(T2c+T2a)

pause

disp('*kinetic energy of link 3 - linear part*')

pc3=[(L2*cos(q2)+dc3*cos(q2+q3))*cos(q1) (L2*cos(q2)+dc3*cos(q2+q3))*sin(q1) L1+L2*sin(q2)↙
+dc3*sin(q2+q3)]'
vc3=simplify(diff(pc3,q1)*dq1+diff(pc3,q2)*dq2+diff(pc3,q3)*dq3)
T3c=(1/2)*simplify(m3*vc3'*vc3)

%pause

disp(['* kinetic energy of link 3 - angular part ' ...
      '(in local coordinates to frame 3) *'])

%om3=simplify(R3'*(om2+dq3*z0))
om3=[dq1*sin(q2+q3) dq1*cos(q2+q3) dq2+dq3]'
T3a=(1/2)*om3'*diag([I3xx I3yy I3yy])*om3
```

```
T3=simplify(T3c+T3a)

pause

disp('*** robot kinetic energy ***')

T=simplify(T1+T2+T3)
% T=collect(T,dq1^2);
% T=collect(T,dq2^2);
% T=collect(T,dq3^2);

%pause

disp('*** robot inertia matrix ***')

M(1,1)=diff(T,dq1,2);
TempM1=diff(T,dq1);
M(1,2)=diff(TempM1,dq2);
M(1,3)=diff(TempM1,dq3);
M(2,2)=diff(T,dq2,2);
TempM2=diff(T,dq2);
M(2,3)=diff(TempM2,dq3);
M(3,3)=diff(T,dq3,2);
M(2,1)=M(1,2);
M(3,1)=M(1,3);
M(3,2)=M(2,3);
M=simplify(M)
```

```
pause

disp('*** linear parametrization of inertia matrix ***')

%a1=I1zz+I2xx+I3xx
%a2=I2yy+m2*dc2^2+m3*L2^2-I2xx
%a3=I3yy+m3*dc3^2-I3xx
%a4=m3*L2*dc3
%a5=I2yy+m2*dc2^2+m3*dc3^2+I3yy+m3*L2^2
%a6=I3yy+m3*dc3^2

M(3,3)=subs(M(3,3),I3yy+m3*dc3^2,a6);
M(2,3)=subs(M(2,3),{m3*dc3^2+I3yy,m3*L2*dc3},{a6,a4});
M(3,2)=M(2,3);
M(2,2)=subs(M(2,2),{m3*dc3^2+I2yy+m2*dc2^2+I3yy+m3*L2^2,m3*L2*dc3,},{a5,a4});
%special treatment for element (1,1)
M11=M(1,1);
M11=subs(M11,{cos(2*q2 + 2*q3),cos(2*q2)},{2*(cos(q2+q3))^2-1,2*(cos(q2))^2-1});
M11=subs(M11,{sin(q2)^2},{1-cos(q2)^2});
M11=simplify(M11);
M11=collect(M11,'cos');
M11=subs(M11,{I1zz+I2xx+I3xx,m3*L2*dc3},{a1,a4});
M11=subs(M11,{I2yy+m2*dc2^2+m3*L2^2-I2xx,I3yy+m3*dc3^2-I3xx},{a2,a3});
M(1,1)=M11;
% display final result
M

pause
```

```
disp('* Christoffel matrices *')

q=[q1;q2;q3];
M1=M(:,1);
C1=(1/2)*(jacobian(M1,q)+jacobian(M1,q)'-diff(M,q1))
M2=M(:,2);
C2=(1/2)*(jacobian(M2,q)+jacobian(M2,q)'-diff(M,q2))
M3=M(:,3);
C3=(1/2)*(jacobian(M3,q)+jacobian(M3,q)'-diff(M,q3))

%pause

disp('*** robot centrifugal terms ***')

dq=[dq1;dq2;dq3];
c1=dq'*C1*dq;
c2=dq'*C2*dq;
c3=dq'*C3*dq;
c=[c1;c2;c3]

pause

disp('* check of skew-symmetry of S=dM-2C using with Christoffel symbols *')

dM=diff(M,q1)*dq+diff(M,q2)*dq2+diff(M,q3)*dq3;
C=[dq'*C1;dq'*C2;dq'*C3];
S=dM-2*C
check_S_plus_Stransp=simplify(S+S')
```

```
pause

g=[0;0;-g0];  % gravity acceleration along -z0

disp('* potential energy of link 1 *')

U1=0

%pause

disp('* potential energy of link 2 *')

U2=-m2*g'*pc2

%pause

disp('* potential energy of link 3 *')

U3=-m3*g'*pc3

%pause

disp('*** robot potential energy (due to gravity) ***')

U=simplify(U1+U2+U3)

pause

disp('*** robot gravity terms ***')
```

```
G=jacobian(U,q)'

pause

disp('*** linear parametrization of gravity vector ***')

%a7=(m2*dc2+m3*L2)*g0
%a8=m3*dc3*g0
G=collect(G,'cos');
G(3)=subs(G(3),{m3*dc3*g0},{a8});
G(2)=subs(G(2),{m2*dc2*g0+m3*L2*g0,m3*dc3*g0},{a7,a8});

% display final result
G

pause

disp('*** complete dynamic equations in symbolic form ***')

M*[ddq1; ddq2; ddq3]+c+G==[u1 u2 u3]'

pause

disp('*** regressor matrix Y in linear model parametrization Y*a=u ***')

a=[a1;a2;a3;a4;a5;a6;a7;a8];
ddq=[ddq1;ddq2;ddq3];
u=M*ddq+c+G;
```

```
Y=jacobian(u,a)

pause

disp('*** numerical evaluation of kinematic and dynamic coefficients ***')

L1=0.5 % link lengths [m]
L2=0.5
L3=0.4

dc1=L1/2; % link CoMs (on local x axis) [m]
dc2=L2/2;
dc3=L3/2;

m1=15; % link masses [kg]
m2=10;
m3=5;

r1=0.2;  % links as full cylinders with uniform mass of radius r [m]
I1zz=(1/2)*m1*r1^2; % [kg*m^2]
r2=0.1;
I2xx=(1/2)*m2*r2^2;
I2yy=(1/12)*m2*(3*r2^2+L2^2);
I2zz=I2yy;
r3=0.1;
I3xx=(1/2)*m3*r3^2;
I3yy=(1/12)*m3*(3*r3^2+L3^2);
I3zz=I3yy;
```

```
g0=9.81; % acceleration of gravity [m/s^2]

a1=I1zz+I2xx+I3xx % from a1 to a6 [kg*m^2]
a2=I2yy+m2*dc2^2+m3*L2^2-I2xx
a3=I3yy+m3*dc3^2-I3xx
a4=m3*L2*dc3
a5=I2yy+m2*dc2^2+m3*dc3^2+I3yy+m3*L2^2
a6=I3yy+m3*dc3^2
a7=(m2*dc2+m3*L2)*g0  % a7 and a8 [kg*m^2/s^2]
a8=m3*dc3*g0

disp('***end***')

% end
```

```
**** dynamic model of 3R spatial elbow-type robot ****

[press return to proceed at every pause (or comment them all)]

* DH rotation matrices *

R1 =

[cos(q1), 0,  sin(q1)]
[sin(q1), 0, -cos(q1)]
[      0, 1,        0]


R2 =

[cos(q2), -sin(q2), 0]
[sin(q2),  cos(q2), 0]
[      0,        0, 1]


R3 =

[cos(q3), -sin(q3), 0]
[sin(q3),  cos(q3), 0]
[      0,        0, 1]

* kinetic energy of link 1 *

T1 =
```

```
(I1zz*dq1^2)/2

* kinetic energy of link 2 - linear part *

pc2 =

dc2*cos(q1)*cos(q2)
dc2*cos(q2)*sin(q1)
   L1 + dc2*sin(q2)


vc2 =

- dc2*dq1*cos(q2)*sin(q1) - dc2*dq2*cos(q1)*sin(q2)
  dc2*dq1*cos(q1)*cos(q2) - dc2*dq2*sin(q1)*sin(q2)
                                  dc2*dq2*cos(q2)


T2c =

(dc2^2*m2*(dq1^2 - dq1^2*sin(q2)^2 + dq2^2))/2

* kinetic energy of link 2 - angular part (in local coordinates to frame 2) *

om2 =

dq1*sin(q2)
dq1*cos(q2)
```

```
        dq2


T2a =

(I2yy*dq2^2)/2 + (I2yy*dq1^2*cos(q2)^2)/2 + (I2xx*dq1^2*sin(q2)^2)/2


T2 =

(I2yy*dq2^2)/2 + (I2yy*dq1^2*cos(q2)^2)/2 + (I2xx*dq1^2*sin(q2)^2)/2 + (dc2^2*m2*(dq1^2 -↙
dq1^2*sin(q2)^2 + dq2^2))/2

*kinetic energy of link 3 - linear part*

pc3 =

cos(q1)*(dc3*cos(q2 + q3) + L2*cos(q2))
sin(q1)*(dc3*cos(q2 + q3) + L2*cos(q2))
     L1 + dc3*sin(q2 + q3) + L2*sin(q2)


vc3 =

- dq2*cos(q1)*(dc3*sin(q2 + q3) + L2*sin(q2)) - dq1*sin(q1)*(dc3*cos(q2 + q3) + L2*cos↙
(q2)) - dc3*dq3*sin(q2 + q3)*cos(q1)
  dq1*cos(q1)*(dc3*cos(q2 + q3) + L2*cos(q2)) - dq2*sin(q1)*(dc3*sin(q2 + q3) + L2*sin↙
(q2)) - dc3*dq3*sin(q2 + q3)*sin(q1)
                                                                      dq2*(dc3*cos(q2 + q3) +↙
```

```
L2*cos(q2)) + dc3*dq3*cos(q2 + q3)


T3c =

(m3*(L2^2*dq1^2 + 2*L2^2*dq2^2 + dc3^2*dq1^2 + 2*dc3^2*dq2^2 + 2*dc3^2*dq3^2 +↙
L2^2*dq1^2*cos(2*q2) + dc3^2*dq1^2*cos(2*q2 + 2*q3) + 4*dc3^2*dq2*dq3 + 2*L2*dc3*dq1^2*cos↙
(q3) + 4*L2*dc3*dq2^2*cos(q3) + 2*L2*dc3*dq1^2*cos(2*q2 + q3) + 4*L2*dc3*dq2*dq3*cos(q3)))↙
/4

* kinetic energy of link 3 - angular part (in local coordinates to frame 3) *

om3 =

dq1*sin(q2 + q3)
dq1*cos(q2 + q3)
       dq2 + dq3


T3a =

(I3yy*dq1^2*cos(q2 + q3)^2)/2 + (I3xx*dq1^2*sin(q2 + q3)^2)/2 + I3yy*(dq2 + dq3)*(dq2/2 +↙
dq3/2)


T3 =

(I3xx*dq1^2)/4 + (I3yy*dq1^2)/4 + (I3yy*dq2^2)/2 + (I3yy*dq3^2)/2 + (L2^2*dq1^2*m3)/4 +↙
(L2^2*dq2^2*m3)/2 + (dc3^2*dq1^2*m3)/4 + (dc3^2*dq2^2*m3)/2 + (dc3^2*dq3^2*m3)/2 +↙
```

```
I3yy*dq2*dq3 - (I3xx*dq1^2*cos(2*q2 + 2*q3))/4 + (I3yy*dq1^2*cos(2*q2 + 2*q3))/4 +↙
dc3^2*dq2*dq3*m3 + (L2^2*dq1^2*m3*cos(2*q2))/4 + (dc3^2*dq1^2*m3*cos(2*q2 + 2*q3))/4 +↙
(L2*dc3*dq1^2*m3*cos(q3))/2 + L2*dc3*dq2^2*m3*cos(q3) + (L2*dc3*dq1^2*m3*cos(2*q2 + q3))/2↙
+ L2*dc3*dq2*dq3*m3*cos(q3)

*** robot kinetic energy ***

T =

(I3xx*dq1^2)/4 + (I2yy*dq2^2)/2 + (I3yy*dq1^2)/4 + (I3yy*dq2^2)/2 + (I3yy*dq3^2)/2 +↙
(I1zz*dq1^2)/2 + (L2^2*dq1^2*m3)/4 + (L2^2*dq2^2*m3)/2 + (dc3^2*dq1^2*m3)/4 +↙
(dc3^2*dq2^2*m3)/2 + (dc3^2*dq3^2*m3)/2 + I3yy*dq2*dq3 + (I2yy*dq1^2*cos(q2)^2)/2 +↙
(I2xx*dq1^2*sin(q2)^2)/2 - (I3xx*dq1^2*cos(2*q2 + 2*q3))/4 + (I3yy*dq1^2*cos(2*q2 + 2*q3))↙
/4 + (dc2^2*m2*(dq1^2 - dq1^2*sin(q2)^2 + dq2^2))/2 + dc3^2*dq2*dq3*m3 +↙
(L2^2*dq1^2*m3*cos(2*q2))/4 + (dc3^2*dq1^2*m3*cos(2*q2 + 2*q3))/4 + (L2*dc3*dq1^2*m3*cos↙
(q3))/2 + L2*dc3*dq2^2*m3*cos(q3) + (L2*dc3*dq1^2*m3*cos(2*q2 + q3))/2 +↙
L2*dc3*dq2*dq3*m3*cos(q3)

*** robot inertia matrix ***

M =

[I3xx/2 + I3yy/2 + I1zz - (I3xx*cos(2*q2 + 2*q3))/2 + (I3yy*cos(2*q2 + 2*q3))/2 +↙
(L2^2*m3)/2 + (dc3^2*m3)/2 + I2yy*cos(q2)^2 + I2xx*sin(q2)^2 - (dc2^2*m2*(2*sin(q2)^2 -↙
2))/2 + (L2^2*m3*cos(2*q2))/2 + (dc3^2*m3*cos(2*q2 + 2*q3))/2 + L2*dc3*m3*cos(q3) +↙
L2*dc3*m3*cos(2*q2 + q3),↙
0,                                     0]
[↙
0, m3*L2^2 + 2*m3*cos(q3)*L2*dc3 + m2*dc2^2 + m3*dc3^2 + I2yy + I3yy, m3*dc3^2 + L2*m3*cos↙
```

```
(q3)*dc3 + I3yy]
[↙
0,                              m3*dc3^2 + L2*m3*cos(q3)*dc3 + I3yy,↙
m3*dc3^2 + I3yy]

*** linear parametrization of inertia matrix ***

M =

[a1 + a4*cos(q3) + a4*cos(2*q2 + q3) + a3*cos(q2 + q3)^2 + a2*cos(q2)^2,↙
0,              0]
[                                                                  0, a5 + 2*a4*cos↙
(q3), a6 + a4*cos(q3)]
[                                                                  0,   a6 + a4*cos↙
(q3),             a6]

* Christoffel matrices *

C1 =

[                                                                  0, - a4*sin(2*q2 +↙
q3) - a3*cos(q2 + q3)*sin(q2 + q3) - a2*cos(q2)*sin(q2), - (a4*sin(q3))/2 - (a4*sin(2*q2 +↙
q3))/2 - a3*cos(q2 + q3)*sin(q2 + q3)]
[- a4*sin(2*q2 + q3) - a3*cos(q2 + q3)*sin(q2 + q3) - a2*cos(q2)*sin(q2),↙
0,                                                                         0]
[- (a4*sin(q3))/2 - (a4*sin(2*q2 + q3))/2 - a3*cos(q2 + q3)*sin(q2 + q3),↙
0,                                                                         0]
```

```
C2 =

[a4*sin(2*q2 + q3) + a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2),             0,↙
0]
[                                                                     0,             0, -↙
a4*sin(q3)]
[                                                                     0, -a4*sin(q3), -↙
a4*sin(q3)]


C3 =

[(a4*sin(q3))/2 + (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3),            0, 0]
[                                                                     0, a4*sin(q3), 0]
[                                                                     0,            0, 0]

*** robot centrifugal terms ***

c =

- dq1*(dq2*(a4*sin(2*q2 + q3) + a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2)) + dq3*↙
((a4*sin(q3))/2 + (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3))) - dq1*dq3*↙
((a4*sin(q3))/2 + (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3)) - dq1*dq2*(a4*sin↙
(2*q2 + q3) + a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2))
↙
(a4*sin(2*q2 + q3) + a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2))*dq1^2 - dq3*↙
(a4*dq2*sin(q3) + a4*dq3*sin(q3)) - a4*dq2*dq3*sin(q3)
↙
((a4*sin(q3))/2 + (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3))*dq1^2 + a4*sin(q3)↙
```

```
*dq2^2

* check of skew-symmetry of S=dM-2C using with Christoffel symbols *

S =

[2*dq2*(a4*sin(2*q2 + q3) + a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2)) - dq2*↙
(2*a4*sin(2*q2 + q3) + 2*a3*cos(q2 + q3)*sin(q2 + q3) + 2*a2*cos(q2)*sin(q2)) - dq3*↙
(a4*sin(q3) + a4*sin(2*q2 + q3) + 2*a3*cos(q2 + q3)*sin(q2 + q3)) + 2*dq3*((a4*sin(q3))/2↙
+ (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3)), 2*dq1*(a4*sin(2*q2 + q3) + a3*cos↙
(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2)), 2*dq1*((a4*sin(q3))/2 + (a4*sin(2*q2 + q3))↙
/2 + a3*cos(q2 + q3)*sin(q2 + q3))]
[↙
-2*dq1*(a4*sin(2*q2 + q3) + a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2)),↙
0,                                              2*a4*dq2*sin(q3) + a4*dq3*sin(q3)]
[↙
-2*dq1*((a4*sin(q3))/2 + (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3)),↙
- 2*a4*dq2*sin(q3) - a4*dq3*sin(q3),↙
0]


check_S_plus_Stransp =

[0, 0, 0]
[0, 0, 0]
[0, 0, 0]

* potential energy of link 1 *
```

```
U1 =
 
     0
 
* potential energy of link 2 *
 
U2 =
 
g0*m2*(L1 + dc2*sin(q2))
 
* potential energy of link 3 *
 
U3 =
 
g0*m3*(L1 + dc3*sin(q2 + q3) + L2*sin(q2))
 
*** robot potential energy (due to gravity) ***
 
U =
 
g0*m2*(L1 + dc2*sin(q2)) + g0*m3*(L1 + dc3*sin(q2 + q3) + L2*sin(q2))
 
*** robot gravity terms ***
 
G =
 
                                                        0
g0*m3*(dc3*cos(q2 + q3) + L2*cos(q2)) + dc2*g0*m2*cos(q2)
                                    dc3*g0*m3*cos(q2 + q3)
```

```
*** linear parametrization of gravity vector ***

G =

                                   0
a8*cos(q2 + q3) + a7*cos(q2)
             a8*cos(q2 + q3)

*** complete dynamic equations in symbolic form ***

ans =

ddq1*(a1 + a4*cos(q3) + a4*cos(2*q2 + q3) + a3*cos(q2 + q3)^2 + a2*cos(q2)^2) - dq1*(dq2*↙
(a4*sin(2*q2 + q3) + a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2)) + dq3*((a4*sin↙
(q3))/2 + (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3))) - dq1*dq3*((a4*sin(q3))/2↙
+ (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3)) - dq1*dq2*(a4*sin(2*q2 + q3) +↙
a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2)) == u1
↙
(a4*sin(2*q2 + q3) + a3*cos(q2 + q3)*sin(q2 + q3) + a2*cos(q2)*sin(q2))*dq1^2 + ddq2*(a5 +↙
2*a4*cos(q3)) + ddq3*(a6 + a4*cos(q3)) - dq3*(a4*dq2*sin(q3) + a4*dq3*sin(q3)) + a8*cos(q2↙
+ q3) + a7*cos(q2) - a4*dq2*dq3*sin(q3) == u2
↙
((a4*sin(q3))/2 + (a4*sin(2*q2 + q3))/2 + a3*cos(q2 + q3)*sin(q2 + q3))*dq1^2 + a4*sin(q3)↙
*dq2^2 + ddq2*(a6 + a4*cos(q3)) + a6*ddq3 + a8*cos(q2 + q3) == u3

*** regressor matrix Y in linear model parametrization Y*a=u ***

Y =
```

```
[ddq1, ddq1*cos(q2)^2 - 2*dq1*dq2*cos(q2)*sin(q2), ddq1*cos(q2 + q3)^2 - dq1*(dq2*cos(q2 +↙
q3)*sin(q2 + q3) + dq3*cos(q2 + q3)*sin(q2 + q3)) - dq1*dq2*cos(q2 + q3)*sin(q2 + q3) -↙
dq1*dq3*cos(q2 + q3)*sin(q2 + q3), ddq1*(cos(2*q2 + q3) + cos(q3)) - dq1*(dq2*sin(2*q2 +↙
q3) + dq3*(sin(2*q2 + q3)/2 + sin(q3)/2)) - dq1*dq2*sin(2*q2 + q3) - dq1*dq3*(sin(2*q2 +↙
q3)/2 + sin(q3)/2),    0,            0,       0,             0]
[   0,                       dq1^2*cos(q2)*sin(q2),↙
dq1^2*cos(q2 + q3)*sin(q2 + q3),↙
sin(2*q2 + q3)*dq1^2 - dq3*(dq2*sin(q3) + dq3*sin(q3)) + 2*ddq2*cos(q3) + ddq3*cos(q3) -↙
dq2*dq3*sin(q3), ddq2,        ddq3, cos(q2), cos(q2 + q3)]
[   0,                                            0,↙
dq1^2*cos(q2 + q3)*sin(q2 + q3),↙
(sin(2*q2 + q3)/2 + sin(q3)/2)*dq1^2 + sin(q3)*dq2^2 + ddq2*cos(q3),    0, ddq2 + ddq3,↙
0, cos(q2 + q3)]

*** numerical evaluation of kinematic and dynamic coefficients ***

L1 =

    0.5000


L2 =

    0.5000


L3 =
```

```
    0.4000


a1 =

    0.3750


a2 =

    2.0583


a3 =

    0.2542


a4 =

    0.5000


a5 =

    2.3875


a6 =
```

```
    0.2792


a7 =

   49.0500


a8 =

    9.8100

***end***
>>
```