Contents

1 About the project

In this project, our aim is to implement kinematic analysis and several control approaches on 3 degrees of freedom(DoF) spatial robot. The robot utilizes a parallelogram linkage in its design to keep inertial contributions of the motors as low as possible. However, this mechanism ensures the rigidity of the mechanism to be high. Another benefit of such mechanism is high bandwith of the manipulator. This is a common kinematic arrangement which has been been used by many industrial robots such as ABB IRB460[1] and Motoman EPL300[2]. For the kinematic analysis, the

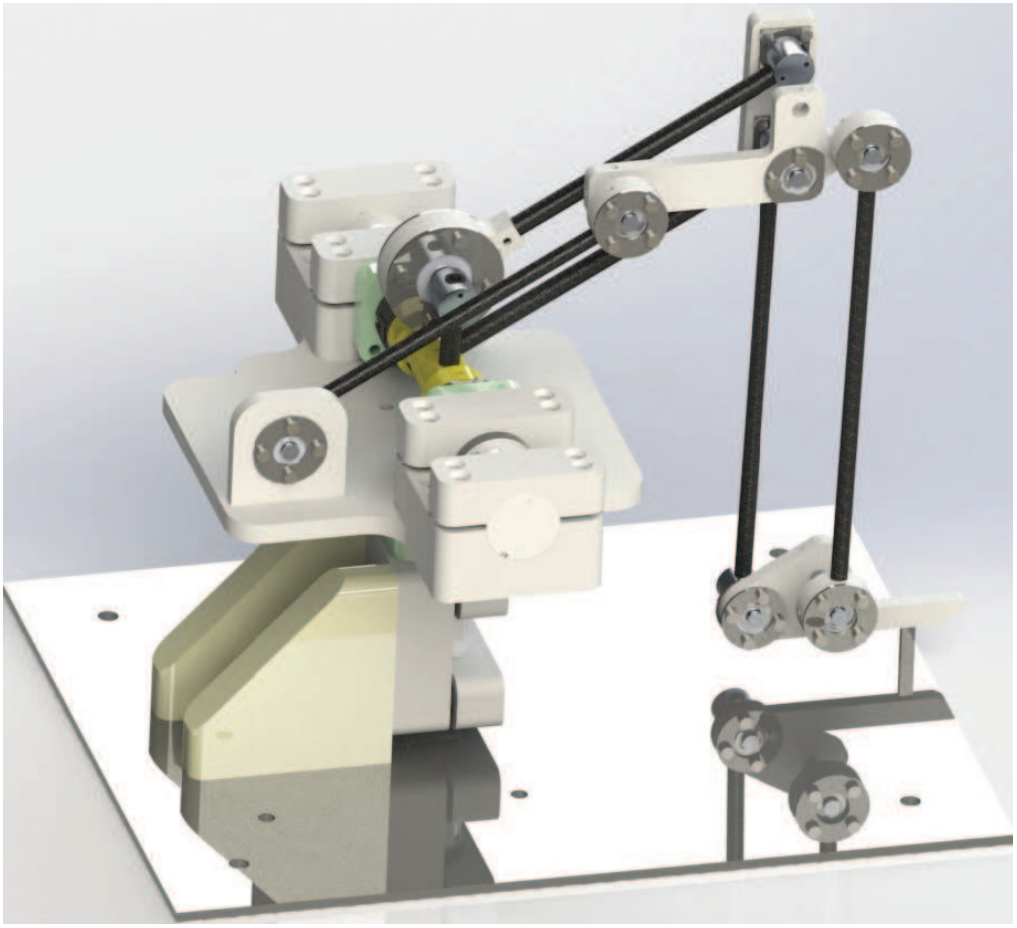


Figure 1: The solid model of the 3-DoF spatial manipulator

manipulator can be considered to consist of four rigid links: the fixed (Newtonian) link N, the base link A, the first link B, and the second link C. The links A, B and C are actuated with DC motors commanding torques τ_A, τ_B, and τ_C, respectively.

2 Denavit Hartenberg Parameters

A mathematical algorithm was created to obtain rotation and jacobian matrices with respect to given DH parameters and defined revolute/prismatic joints:

```
syms q1 q2 q3 i4 i5 i6 a1 a2 a3 l1 l2 l3 l4 l5 l6 d1 d2 d3 d4 d5 ...
    d6 fi

%% Configuration of denavit hartenberg parameters
d=[106;0;0] ;
theta=str2sym('[q1;q2;q3]');
a=[0;99;140];
alpha=str2sym('[pi/2;0;0]');

DH=str2sym('[d theta a alpha]');
numberoflinks=length(d);
%%
%%Transformation matrix declaration through link i to i+1
for i=1:numberoflinks
    H_{i}=[cos(theta(i)) -sin(theta(i))*cos(alpha(i)) ...
        sin(theta(i))*sin(alpha(i)) a(i)*cos(theta(i));
    sin(theta(i))     cos(theta(i))*cos(alpha(i))    ...
        -cos(theta(i))*sin(alpha(i)) a(i)*sin(theta(i));
    0 sin(alpha(i)) cos(alpha(i)) d(i);
    0 0 0 1];
end
%%
%%Transformation matrix through the end effector
Ha=1;
for i=1:numberoflinks
    Ha=Ha*cell2sym(H_(i));
    Hb{i}=Ha
end

%%Kinematic jacobian
links=[0 0 0]
J=sym(zeros(6,length(links)));
p0=[0;0;0]
p1=str2sym('[0 ; 0 ;l1]')
p2=str2sym('[l2*cos(q1)*cos(q2); l2*sin(q1)*cos(q2);l2*sin(q2)+l1]')
p3=str2sym('[l3*cos(q2+q3)*cos(q1)+l2*cos(q1)*cos(q2); ...
    l3*cos(q2+q3)*sin(q1)+l2*sin(q1)*cos(q2); ...
    l3*sin(q2+q3)+l2*sin(q2)+l1]')
P=[p0 p1 p2 p3];
for i=1:3%whole jacobian
    if links(i)==1 %prismatic
        H=cell2sym(Hb(i));
```

```
38          J(:,i)=[H(1:3,1:3)*[0;0;1];  [0;0;0]];
39      end
40      if links(i)==0%revolute
41          H=cell2sym(Hb(i));
42          J(:,i)=[cross((H(1:3,1:3)*[0;0;1]),(P(:,3)-P(:,i))) ;  ...
                H(1:3,1:3)*[0;0;1]];
43
44      end
45      J(4:6,1)=[0;0;1];
46
47  end
48  %%
```

In this algorithm by changing i value in the last for loop, one can obtain J_{v_1}, J_{v_2}, J_{v_3}, J_{w_1}, J_{w_2} and J_{w_3} respectively. Then those jacobians were used in derivation of equation of motion which is presented in the appendix.

3 Forward and Inverse Kinematics

The forward and inverse kinematics are defined by MATLAB, and the codes are given below

```
function q =Inverse_Kinematic_CL(x)

xe = x(1);
ye = x(2);
ze = x(3);
L = [105.8;98.7459;140.2516];
r = (xe^2+ye^2);

mag_C = (r+(ze-L(1))^2)^(1/2);

C_dir = -1*((r^(1/2))*[1; 0; 0] + (ze-L(1))*[0; 0; 1])/mag_C;
Δ = (mag_C^2+L(3)^2-L(2)^2)/(2*mag_C);

cxk = cross(C_dir, [0; 1; 0]);

A = -1*((L(3)^2-Δ^2))^(1/2)*cxk+(Δ-mag_C)*C_dir;
B = 1*((L(3)^2-Δ^2)^(1/2))*cxk-Δ*C_dir;

theta1 = atan2(ye, xe);
theta2 = atan2(A(3),A(1));
theta3 = atan2(B(3),B(1))-theta2;

q = [theta1; theta2; theta3];
```

```
function x = Forward_Kinematics_CL(q)
o1=q(1);
o2=q(2);
o3=q(3);
l1 = 105.8;
l2 = 98.7459;
l3 = 140.2516;
ht01 =htf(o1,l1,pi/2,0);

ht12=htf(o2,0,0,l2);

ht23=htf(o3,0,0,l3);

ht02=ht01*ht12;
ht03=ht02*ht23;
x=[ht03(1,4);ht03(2,4);ht03(3,4)];
```

```
function q_dots = Inverse_K_ML(q,x_dot)

q1=q(1);
q2=q(2);
q3=q(3);
l1 = 105.8;
l2 = 98.7459;
l3 = 140.2516;
ht01 =htf(q1,l1,pi/2,0);
ht12 =htf(q2,0,0,l2);
ht23 =htf(q3,0,0,l3);
ht02 =ht01*ht12;
ht03 =ht02*ht23;

R01=[ht01(1,1) ht01(1,2) ht01(1,3);ht01(2,1) ht01(2,2) ...
    ht01(2,3);ht01(3,1) ht01(3,2) ht01(3,3)];
R02=[ht02(1,1) ht02(1,2) ht02(1,3);ht02(2,1) ht02(2,2) ...
    ht02(2,3);ht02(3,1) ht02(3,2) ht02(3,3)];
R03=[ht03(1,1) ht03(1,2) ht03(1,3);ht03(2,1) ht03(2,2) ...
    ht03(2,3);ht03(3,1) ht03(3,2) ht03(3,3)];
o0=[0;0;0];

o1=[ht01(1,4);ht01(2,4);ht01(3,4)];
o2=[ht02(1,4);ht02(2,4);ht02(3,4)];
o3=[ht03(1,4);ht03(2,4);ht03(3,4)];
z=[0;0;1];
o3o0=(o3-o0);
o3o1=(o3-o1);
o3o2=(o3-o2);
K_Jac=[cross(z,o3o0) cross(R01*z,o3o1) cross(R02*z,o3o2);o0 ...
    R01*z R02*z];
K_Jac_T=pinv(K_Jac);

q_dot=K_Jac_T*x_dot;

q_dots=[q_dot(1);q_dot(2);q_dot(3)];
```

```
function x_dot =Forward_K_ML(q,q_dot)

q1=q(1);
q2=q(2);
q3=q(3);
l1 = 105.8;
l2 = 98.7459;
l3 = 140.2516;

ee_dot = [- l3*cos(q2 + q3)*sin(q1) - l2*cos(q2)*sin(q1),- ...
    l3*sin(q2 + q3)*cos(q1) - l2*cos(q1)*sin(q2),-l3*sin(q2 + ...
    q3)*cos(q1);...
             l3*cos(q2 + q3)*cos(q1) + l2*cos(q1)*cos(q2),- ...
                l3*sin(q2 + q3)*sin(q1) - ...
                l2*sin(q1)*sin(q2),-l3*sin(q2 + q3)*sin(q1);...
             0,l3*cos(q2 + q3) + l2*cos(q2),l3*cos(q2 + ...
                q3)]*[qdot1;qdot2;qdot3];
x_dot = ee_dot(1);
y_dot = ee_dot(2);
z_dot = ee_dot(3);

x_dot = [x_dot;y_dot;z_dot];
```

And the htf function used in this translations is;

```
1
2 function out= htf(o,d,a,ai)
3
4 out=[cos(o)      -sin(o)*cos(a)   sin(o)*sin(a)    ai*cos(o);
5       sin(o)      cos(o)*cos(a)   -cos(o)*sin(a)   ai*sin(o);
6       0           sin(a)          cos(a)           d        ;
7       0           0               0                1       ];
8 end
```

4 Simulation

Forward kinematics of the manipulator is implemented on MATLAB code to prove 2D and 3D plots of the mechanism. 2D and 3D plots of the mechanism for the constrained rotations of links, is given in the project folder.

```
q1i=180;
q1f=-180;
q2i=-45;
q2f=135;
q3i=-45;
q3f=-135;
q1 = deg2rad(linspace(q1i,q1f,200));
q2 = deg2rad(linspace(q2i,q2f,200));
q3 = deg2rad(linspace(q3i,q3f,200));
l1 = 106;
l2 = 98.7;
l3 = 140.25;
for i=1:length(q1)
    x1(i)=0;
    y1(i)=0;
    z1(i)=0;
    z2(i)=l1;
    x2(i)=0;
    y2(i)=0;
    x3(i)=l2*cos(q1(i))*cos(q2(i));
    y3(i)=l2*sin(q1(i))*cos(q2(i));
    z3(i)=l2*sin(q2(i))+l1;
    x4(i) =  l3*cos(q2(i)+q3(i))*cos(q1(i)) + ...
        l2*cos(q1(i))*cos(q2(i));

    y4(i) = l3*cos(q2(i)+q3(i))*sin(q1(i) ) + ...
        l2*sin(q1(i))*cos(q2(i));

    z4(i) = l3*sin(q2(i)+q3(i)) + l2*sin(q2(i)) + l1;

end
d=1; %defines velocity
j=1:d:length(q1);
for i=1:length(j)


hold off

%plot([x1(j(i)) x2(j(i))],[y1(j(i)) y2(j(i))],'o',[x2(j(i)) ...
    x3(j(i))],[y2(j(i)) y3(j(i))],'--rs',[x3(j(i)) ...
    x4(j(i))],[y3(j(i)) y4(j(i))],'k')
```

```
plot3([x1(j(i)) x2(j(i))],[y1(j(i)) y2(j(i))],[z1(j(i)) ...
    z2(j(i))],'--b',[x2(j(i)) x3(j(i))],[y2(j(i)) ...
    y3(j(i))],[z2(j(i)) z3(j(i))],'-k',[x3(j(i)) ...
    x4(j(i))],[y3(j(i)) y4(j(i))],[z3(j(i)) z4(j(i))],'-r')
title('Motion of 3DoF Spatial Manipulator')

axis([-300 300 -300 300 -300 300]); % last -300 300 will be ...
    added when 2D plot is desired
xlabel('x position(mm)')
ylabel('y position(mm)')
zlabel('z position(mm)')
legend('Show')
grid

MM(i)=getframe(gcf);
end

drawnow;

v = VideoWriter('Robotic Movie.mp4','MPEG-4');
open(v)
writeVideo(v,MM)
close(v)
```

5 Singularities and Manipulability

Possible singularities and manipulability ellipsoids are achieved by the following matlab code:

```
clc
clear
q1i=180;
q1f=-180;
q2i=-45;
q2f=135;
q3i=-135;
q3f=-45;
q1 = deg2rad(linspace(q1i,q1f,20));
q2 = deg2rad(linspace(q2i,q2f,20));
q3 = deg2rad(linspace(q3i,q3f,20));
l1 = 106;
l2 = 98.7;
l3 = 140.25;
a=1;
```

```
for i=1:length(q1)
    for j=1:length(q2)
        for k=1:length(q3)

        x(a,1) =  l3*cos(q2(j)+q3(k))*cos(q1(i)) + ...
            l2*cos(q1(i))*cos(q2(j));

        y(a,1) = l3*cos(q2(j)+q3(k))*sin(q1(i) ) + ...
            l2*sin(q1(i))*cos(q2(j));

        z(a,1) = l3*sin(q2(j)+q3(k)) + l2*sin(q2(j)) + l1;
        a=a+1;
        end
    end
end
scatter3(x,y,z)
xyz=[x,y,z];
B = unique(round(xyz,4),'rows');
if length(xyz)==length(B)
    singularity=0 %there is no singularity
else
    singularity=length(xyz)-length(B) %there are singularity at ...
       infinity point. this number shows number of singularity ...
       for pre-defined angles
end
clear q1 q2 q3
syms q1 q2 q3
J=str2sym('[- l3*cos(q2 + q3)*sin(q1) - l2*cos(q2)*sin(q1),- ...
    l3*sin(q2 + q3)*cos(q1) - l2*cos(q1)*sin(q2),-l3*sin(q2 + ...
    q3)*cos(q1); l3*cos(q2 + q3)*cos(q1) + l2*cos(q1)*cos(q2),- ...
    l3*sin(q2 + q3)*sin(q1) - l2*sin(q1)*sin(q2),-l3*sin(q2 + ...
    q3)*sin(q1);0,l3*cos(q2 + q3) + l2*cos(q2),l3*cos(q2 + q3)]');
M=inv(J*transpose(J));
q1i=150;
q1f=-150;
q2i=-45;
q2f=125;
q3i=-125;
q3f=-45;
q1l = deg2rad(linspace(q1i,q1f,3));
q2l = deg2rad(linspace(q2i,q2f,3));
q3l = deg2rad(linspace(q3i,q3f,3));
a=1;
for i=1:length(q1l)
    for j=1:length(q2l)
        for k=1:length(q3l)
            q1=q1l(i);
            q2=q2l(j);
            q3=q3l(k);
```

```
57              Man(a,:)=eig(eval(M));
58              thet=eig(eval(M)).*[1 0 0]';
59              theta(a)=acos(thet(1));
60              a=a+1;
61          end
62      end
63  end
64  for i=1:length(Man);
65      ra(i)=min(Man(i,:));
66      rb(i)=max(Man(i,:));
67  end
68  % figure
69  % hold
70  % for i=1:length(ra)
71  % [x, y, z] = ellipsoid(0,0,0,ra(i),rb(i),0,30);
72  % aa(:,:,i)=x;
73  % ab(:,:,i)=y;
74  % ac(:,:,i)=z;
75  % S=surfl(x,y,z)
76  %
77  % end
78  clear x y z
79  w=1;
80  t=linspace(1,100,1000);
81  k=zeros(1,length(t));
82  l=k;
83  for i=1:length(ra)
84  k(i,:)=ra(i)*sin(w*t);
85  l(i,:)=rb(i)*cos(w*t);
86  plot(k(i,:),l(i,:))
87  hold on
88  end
```

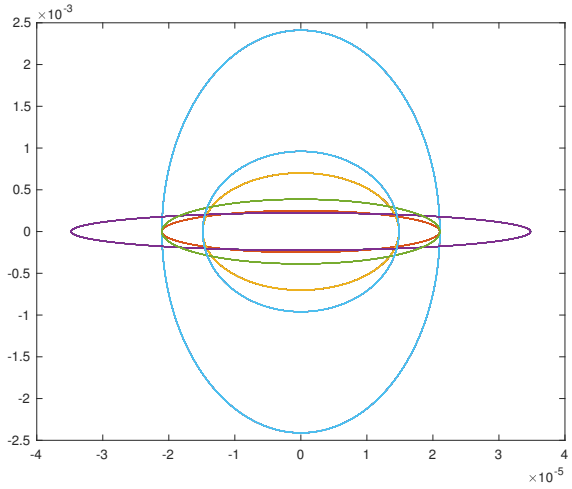


Figure 2: The manipulability ellipsoids of the 3-DoF spatial manipulator for 20 distinct locations

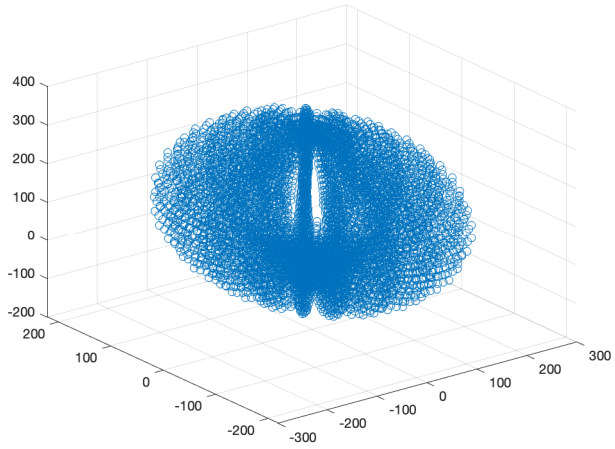


Figure 3: The workspace of the 3-DoF spatial manipulator for constrained rotations

6 Maximum force exerted by the end-effector for 0.5 Nm torque at each link

Maximum force exerted by the end-effector is calculated through the matlab code presented below and 5.9194 N is calculated by this code.

```
clc
clear
q1i=180;
q1f=-180;
q2i=-45;
q2f=135;
q3i=-135;
q3f=-45;
q1 = deg2rad(linspace(q1i,q1f,60));
q2 = deg2rad(linspace(q2i,q2f,60));
q3 = deg2rad(linspace(q3i,q3f,60));
l1 = 106;
l2 = 98.7;
l3 = 140.25;
a=1;
for i=1:length(q1)
    for j=1:length(q2)
        for k=1:length(q3)

o1=q1(i);
```

```
21 o2=q2(i);
22 o3=q3(i);
23
24
25
26 K_Jac=[-cos(q1(i))*(l1 + l3*sin(q2(i) + q3(i)) + l2*sin(q2(i))), ...
      -cos(q1(i))*(l3*sin(q2(i) + q3(i)) + ...
      l2*sin(q2(i))),-l3*sin(q2(i) + ...
      q3(i))*cos(q1(i));-sin(q1(i))*(l1 + l3*sin(q2(i) + q3(i)) + ...
      l2*sin(q2(i))),-sin(q1(i))*(l3*sin(q2(i) + q3(i)) + ...
      l2*sin(q2(i))),-l3*sin(q2(i) + ...
      q3(i))*sin(q1(i));cos(q1(i))*(l3*cos(q2(i) + ...
      q3(i))*cos(q1(i)) + l2*cos(q1(i))*cos(q2(i))) + ...
      sin(q1(i))*(l3*cos(q2(i) + q3(i))*sin(q1(i)) + ...
      l2*cos(q2(i))*sin(q1(i))), cos(q1(i))*(l3*cos(q2(i) + ...
      q3(i))*cos(q1(i)) + l2*cos(q1(i))*cos(q2(i))) + ...
      sin(q1(i))*(l3*cos(q2(i) + q3(i))*sin(q1(i)) + ...
      l2*cos(q2(i))*sin(q1(i))), l3*cos(q2(i) + q3(i))*cos(q1(i))^2 ...
      + l3*cos(q2(i) + q3(i))*sin(q1(i))^2;0,sin(q1(i)), ...
                                                  sin(q1(i)); ...

      0, ...

      -cos(q1(i)),                                              ...
      -cos(q1(i)); ...

      1, ...

      0,                                                    0];
27
28 T= [500;500;500];
29 JT = transpose(K_Jac);
30
31 F=pinv(JT)*T; %since JT is not nxn matrix
32 Force(a)=sqrt(F(1)^2+F(2)^2+F(3)^2);
33         a=a+1;
34         end
35     end
36 end
37 max(Force)
```

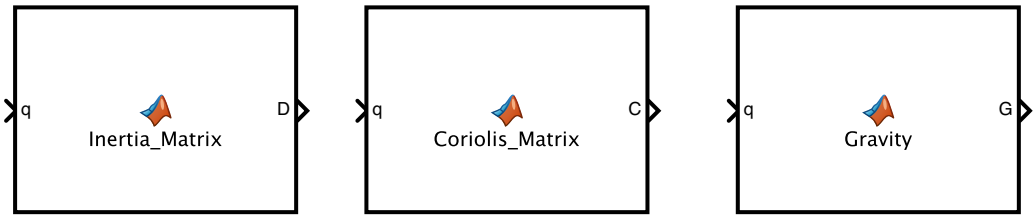
7 Controls

In this project, we are asked to design the following controlers

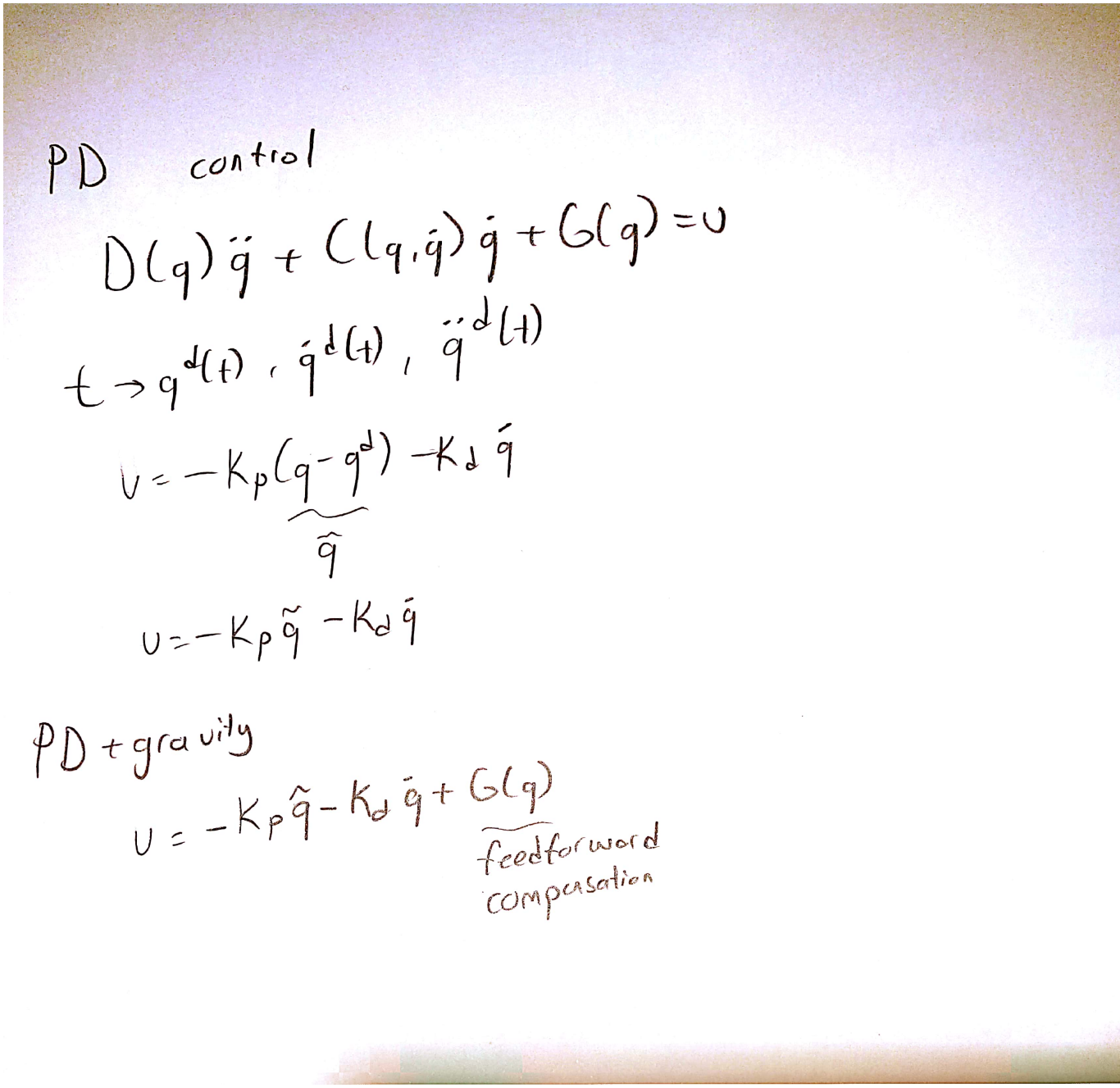
- PD + PD gravity.

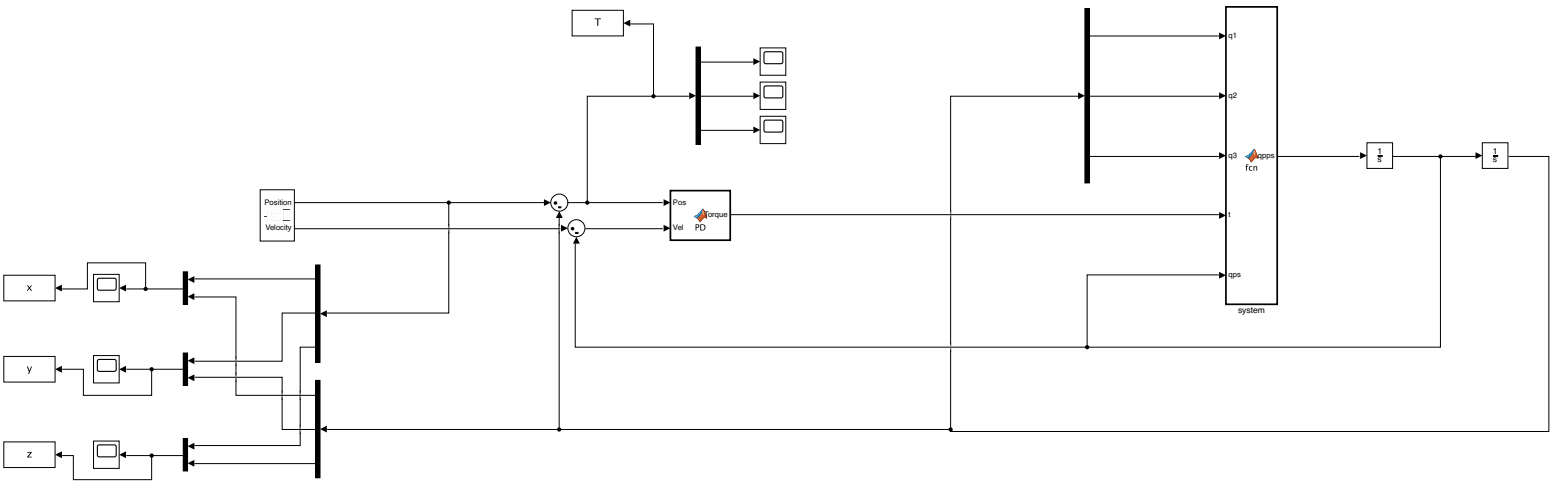
- PID + PID gravity.
- Inverse dynamics motion controller in joint and task space.
- Passivity based controller
- Computed Torque Task and Joint space controller

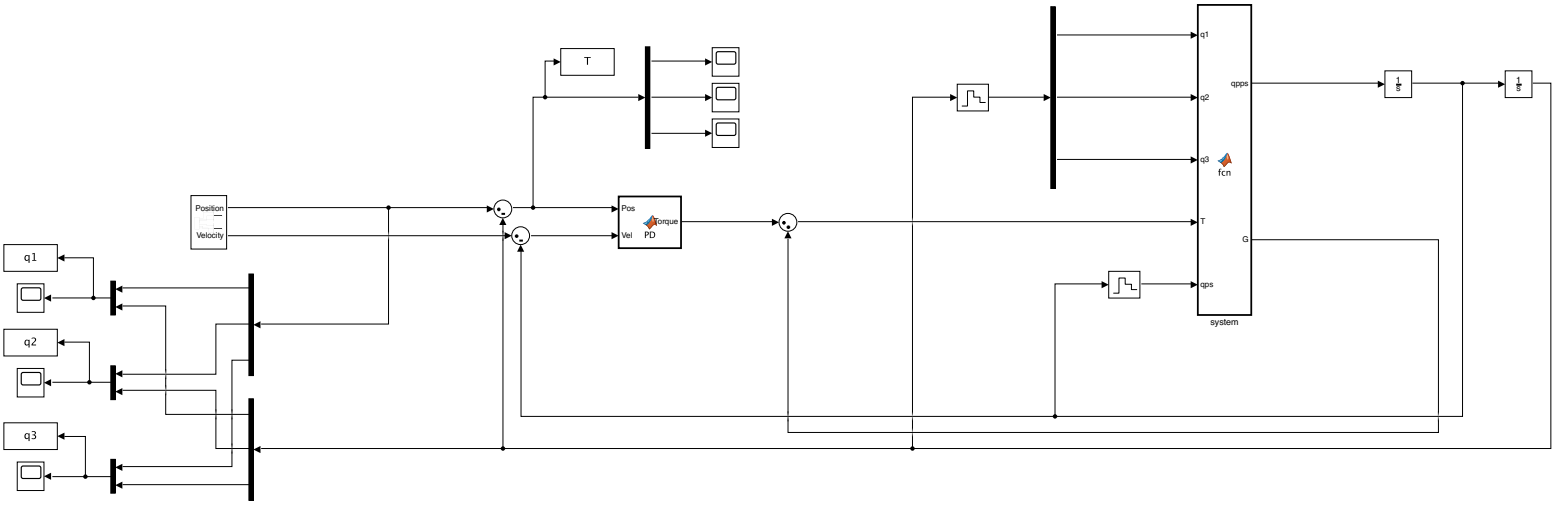
7.1 Dynamic Blocks


q
Inertia_Matrix
D
q
Coriolis_Matrix
C
q
Gravity
G

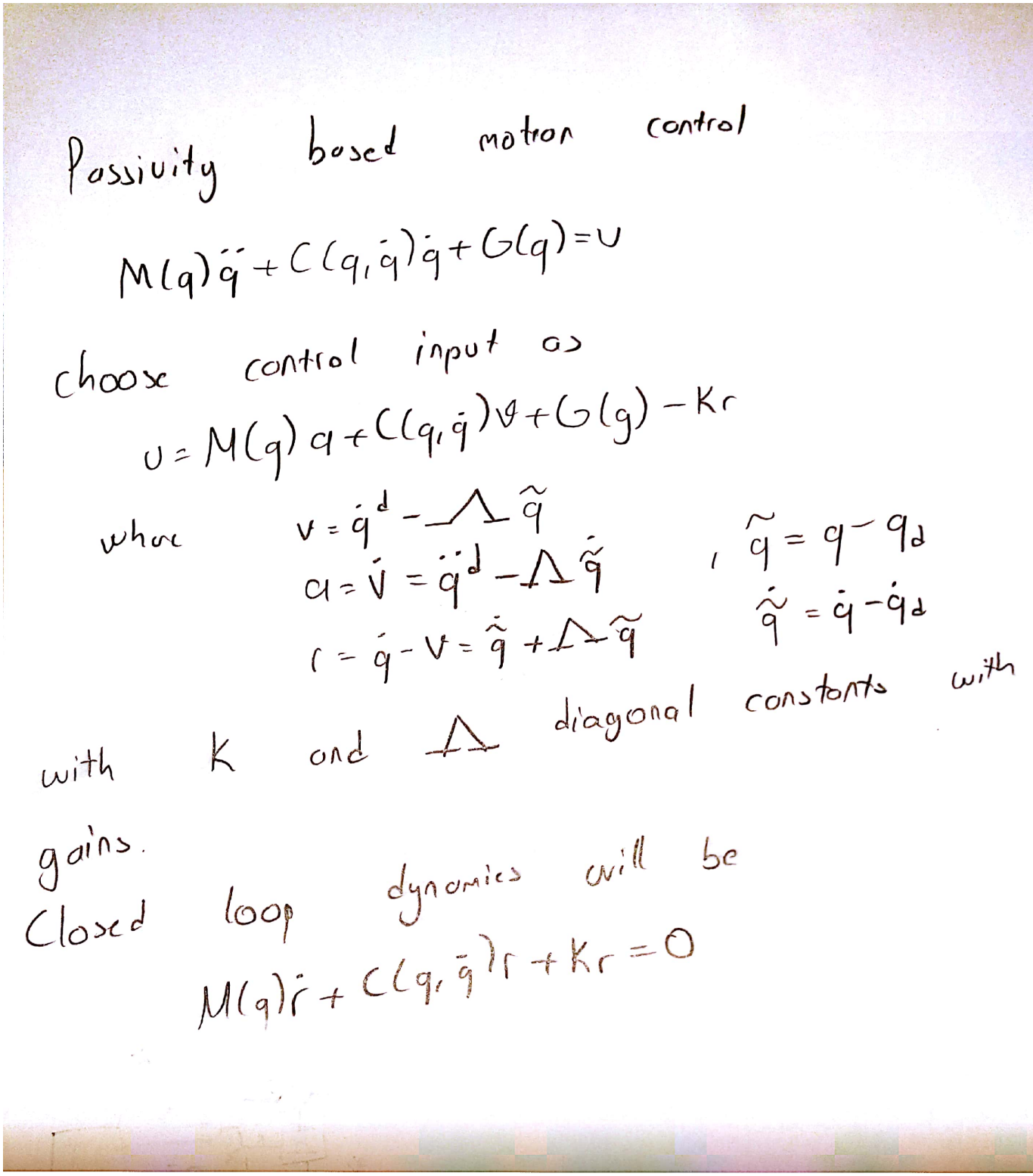
7.2 PD + PD gravity

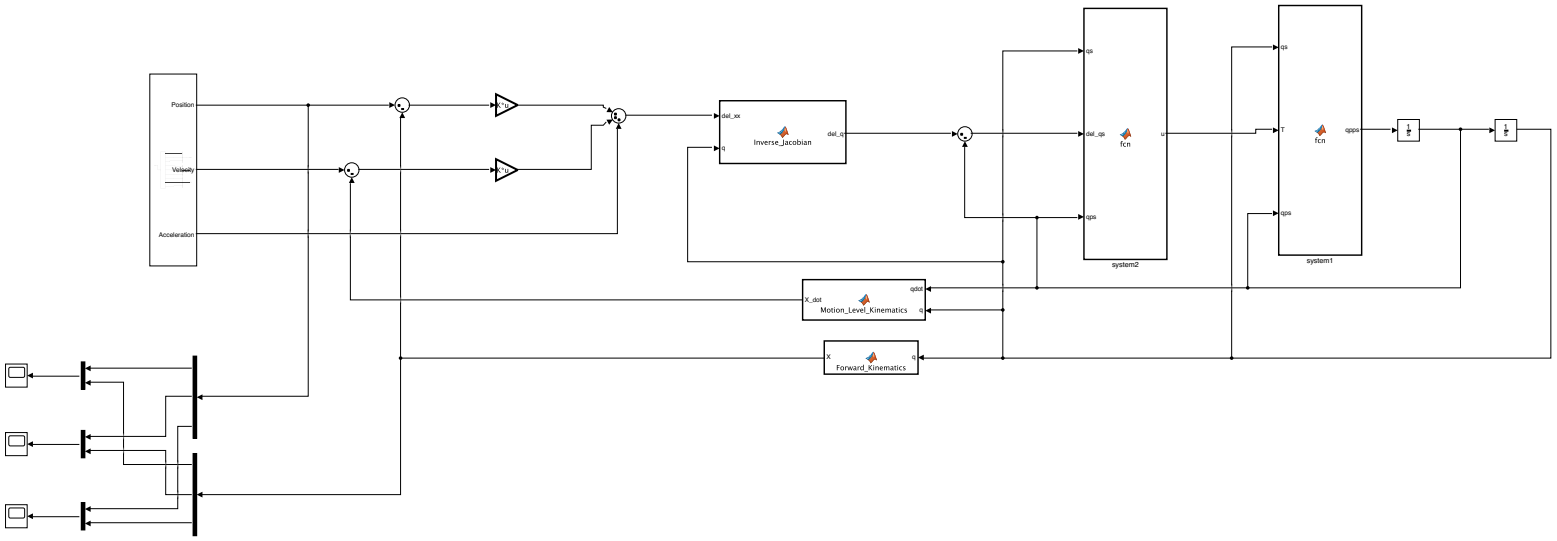



T
Position
Velocity
Pos
Vel
Torque
PD
q1
q2
q3
qpps
fcn
t
qps
system
x
y
z

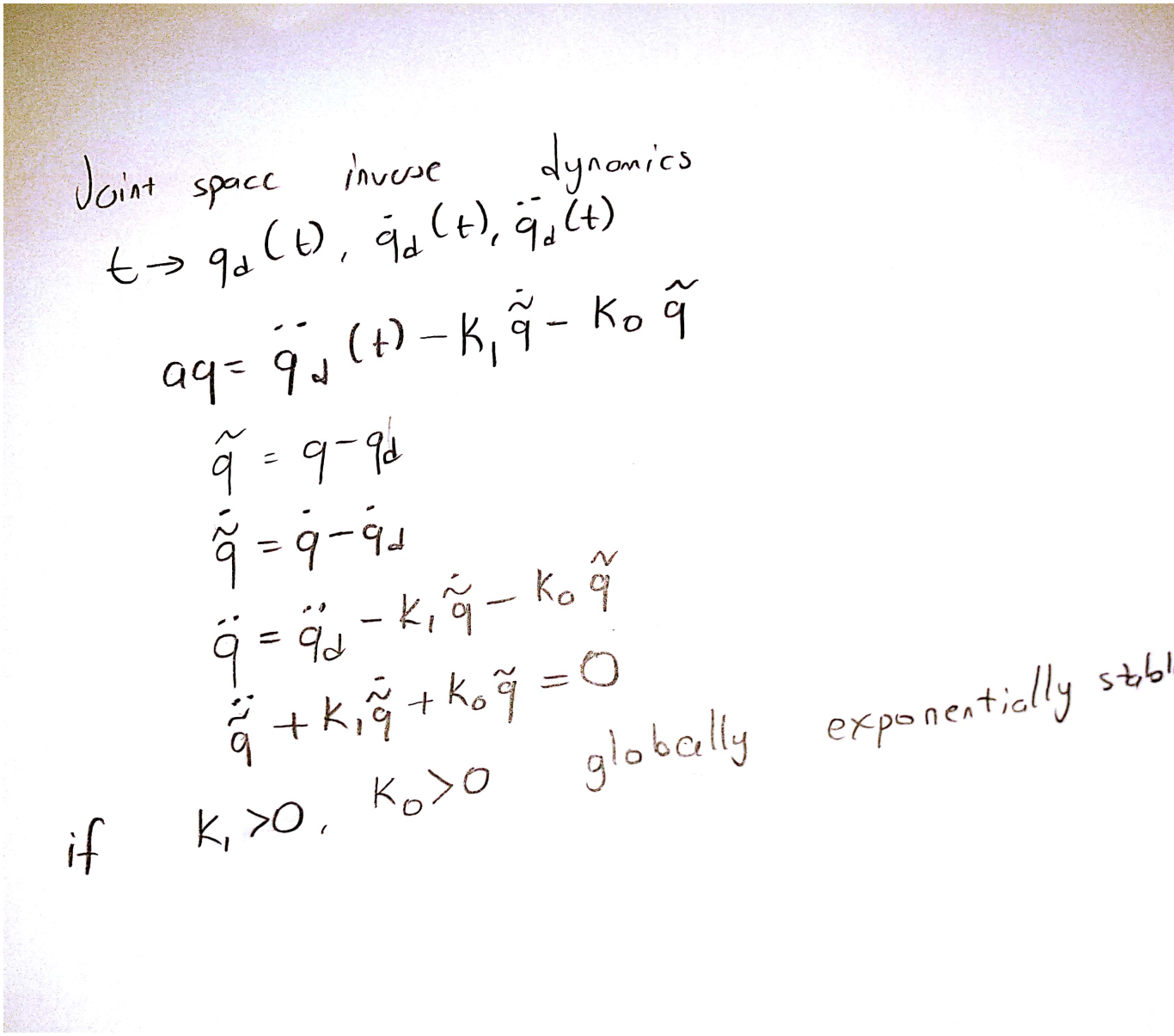

T
Position
Velocity
Pos
Vel
Torque
PD
q1
q2
q3
qpps
qps
fcn
T
G
system
1/s
1/s

7.3 Passivity Based Configuration Space




Position
Velocity
Acceleration
Inverse_Jacobian
fcn
system2
fcn
system1
Motion_Level_Kinematics
Forward_Kinematics

7.4 Computed Torque



Task space inverse dynamics

$$\dot{x} = J_a \dot{q}$$

$$\ddot{x} = \dot{J}_a \dot{q} + J_a \ddot{q}$$

$$\ddot{q} = J_a^{-1} (\ddot{x} - \dot{J}_a \dot{q})$$

apply control law

$$a_q = J_a^{-1} (a_x - \dot{J}_a \dot{q})$$

CL dynamics $\ddot{x} = a_x$

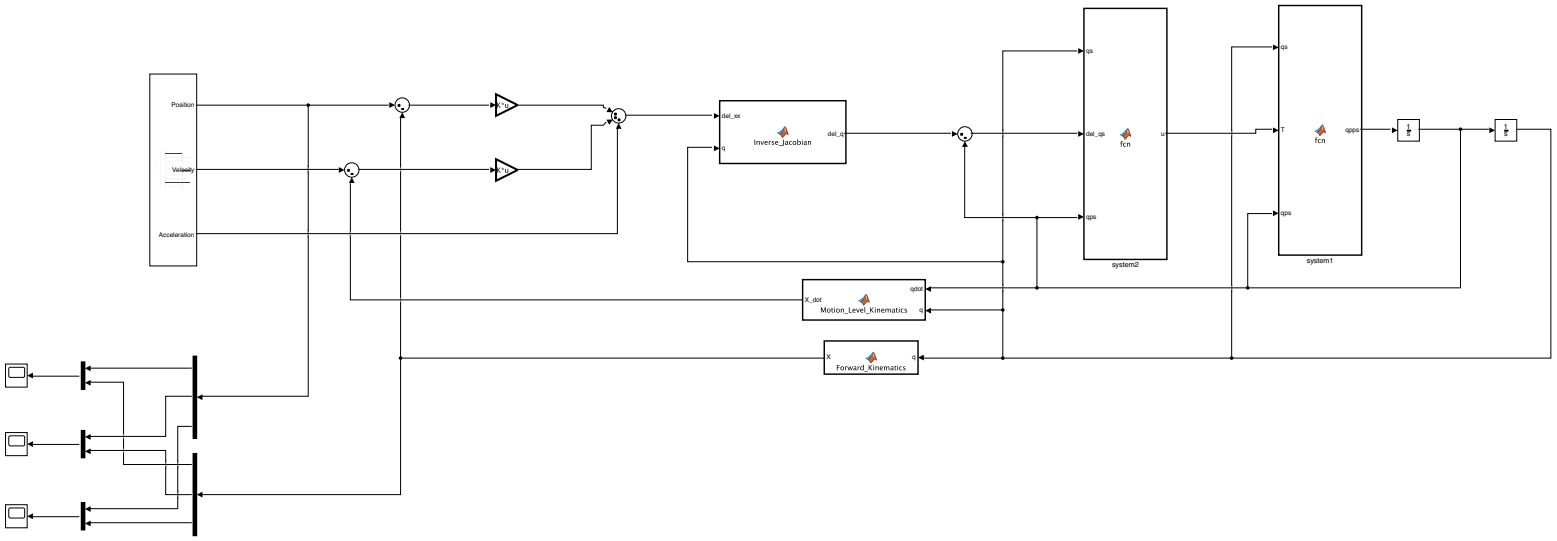
desired traj $t = x^d, \dot{x}^d, \ddot{x}^d$

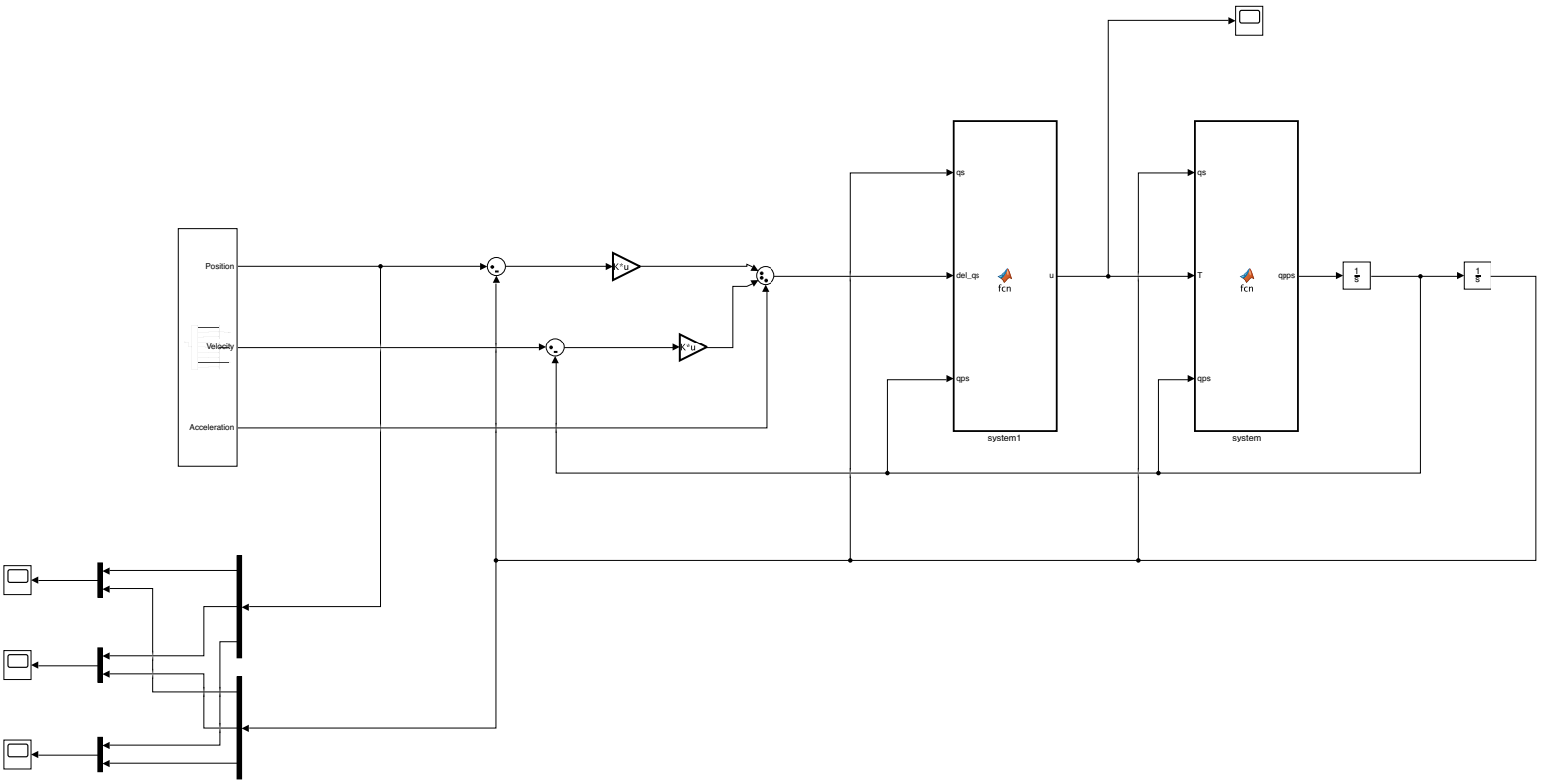
$$\tilde{x} = x - x^d, \quad \dot{\tilde{x}} = \dot{x} - \dot{x}^d$$

$$a_x = \ddot{x}^d - K_A \dot{\tilde{x}} - K_B \tilde{x}$$

$$\ddot{\tilde{x}} + K_A \dot{\tilde{x}} + K_B \tilde{x} = 0$$

$K_A > 0$, $K_B > 0 \Rightarrow$ global exponential stability


Position
Velocity
Acceleration
Inverse_Jacobian
fcn
system2
fcn
system1
Motion_Level_Kinematics
Forward_Kinematics


Position
Velocity
Acceleration
K*u
K*u
qs
del_qs
qps
fcn
u
system1
qs
T
qps
fcn
qpps
system
1/s
1/s

8 Derivation of Equation of Motion

The matlab code for derivation of equation of motion is provided in the project folder. Derivated EoM is represented in the appendix of the report.

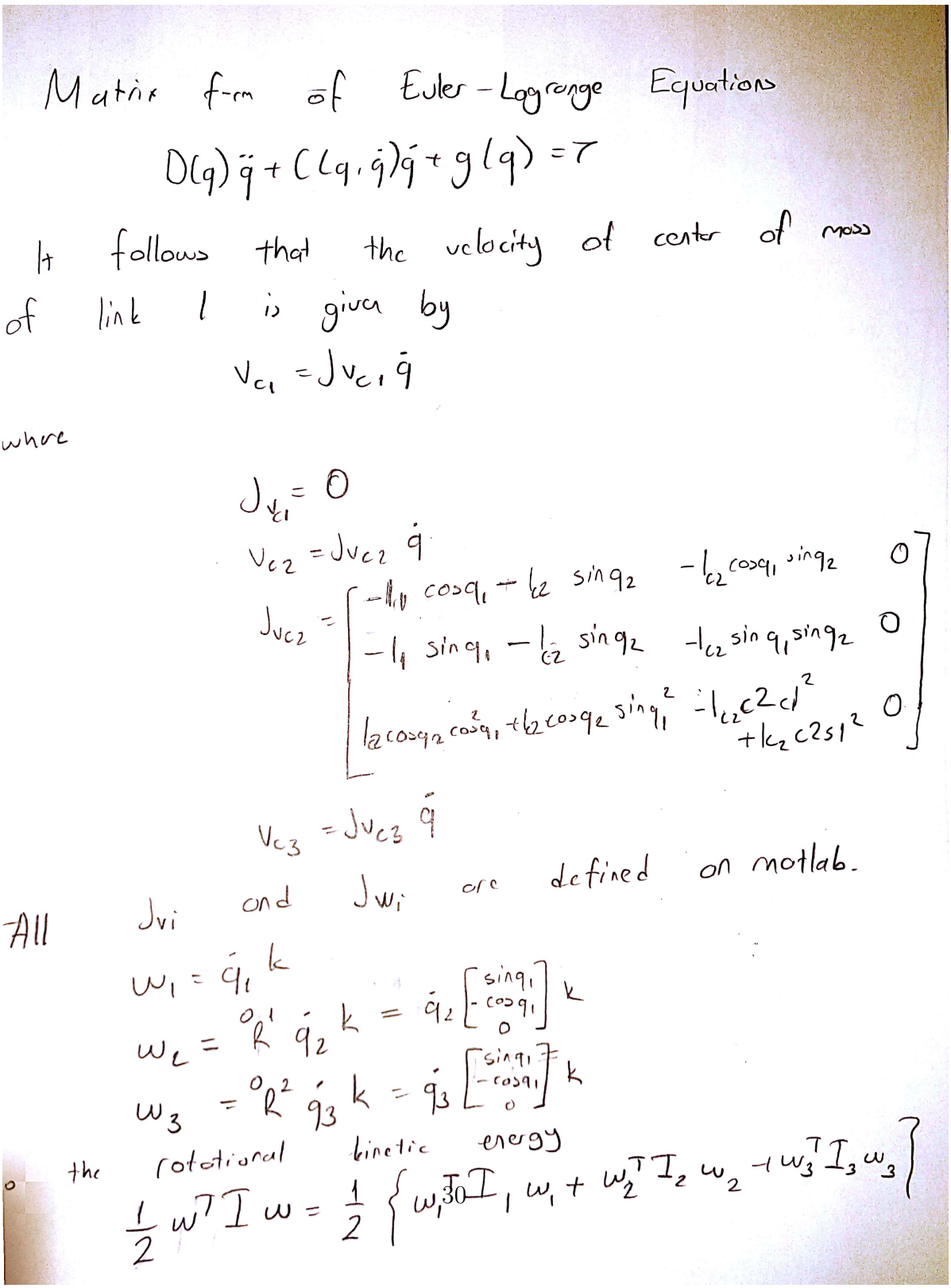
References

[1] Irb-460. https://new.abb.com/products/robotics/industrial-robots/irb-460. Accessed: 2019-01-09.

[2] Motoman epl300. https://www.robots.com/robots/motoman-epl300. Accessed: 2019-01-09.

9 Appendix

9.1 Handwritter derivation for EoM



Matrix f-m of Euler-Lagrange Equations

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

It follows that the velocity of center of mass of link 1 is given by

$$V_{c_1} = J_{v_{c_1}} \dot{q}$$

where

$$J_{v_1} = 0$$

$$V_{c_2} = J_{v_{c_2}} \dot{q}$$

$$J_{v_{c_2}} = \begin{bmatrix} -l_{11}\cos q_1 - l_{c_2}\sin q_2 & -l_{c_2}\cos q_1 \sin q_2 & 0 \\ -l_1 \sin q_1 - l_{c_2}\sin q_2 & -l_{c_2}\sin q_1 \sin q_2 & 0 \\ l_{c_2}\cos q_2 \cos q_1 + l_{c_2}\cos q_2 \sin q_1 & -l_{c_2}c_2c_1^2 + l_{c_2}c_2s_1^2 & 0 \end{bmatrix}$$

$$V_{c_3} = J_{v_{c_3}} \dot{q}$$

All J_{vi} and J_{w_i} are defined on matlab.

$$\omega_1 = \dot{q}_1 k$$

$$\omega_2 = {}^0R^1 \dot{q}_2 k = \dot{q}_2 \begin{bmatrix} \sin q_1 \\ -\cos q_1 \\ 0 \end{bmatrix} k$$

$$\omega_3 = {}^0R^2 \dot{q}_3 k = \dot{q}_3 \begin{bmatrix} \sin q_1 \\ -\cos q_1 \\ 0 \end{bmatrix} k$$

o the rotational kinetic energy

$$\frac{1}{2} w^T I w = \frac{1}{2} \left\{ \omega_1^T {}^0I_1 \omega_1 + \omega_2^T I_2 \omega_2 + \omega_3^T I_3 \omega_3 \right\}$$

since w_i is alligned with k, it becomes

$$\frac{1}{2}\dot{q}^T\left\{I_1\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix}+(I_2+I_3)\begin{bmatrix}\sin^2 q_1 & -\sin q_1\cos q_1 & 0\\ -\sin q_1\cos q_1 & \cos^2 q_1 & 0\\ 0&0&0\end{bmatrix}\right.$$

so that the inertia matrix becomes

$$D(q)=m_i J_{v_{Ci}}^T J_{v_{Ci}}+\left[I_1\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix}+I_2+I_3\begin{bmatrix}\sin^2 q_1 & -\sin q_1\cos q_1 & 0\\ -\sin q_1\cos q_1 & \cos^2 q_1 & 0\\ 0&0&0\end{bmatrix}\right.$$

$D(q)$ is computed analytically. Now we can compute skew symmetric coriolis matrix by using $D(1,1)$ $D(1,2)=D(2,1)$, $D(2,2)$, $D(3,2)=D(2,3)$, $D(3,3)$

$$C_{ijk}=\frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i}+\frac{\partial d_{ki}}{\partial q_j}-\frac{\partial d_{ij}}{\partial q_k}\right\}$$

eliminate constant terms in the matrix will give us. the coriolis matrix. The coriolis matrix is also computed analytically by MATLAB. The last term is gravity matrix.

$$P_1=m_1 g l_{c1}$$
$$P_2=m_2 g(l_{c2}\sin q_2+l_1)$$
$$P_3=m_3 g(l_1+l_2\sin q_2+l_{c3}\sin q_3)$$
$$P=P_1+P_2+P_3$$

$$g(q) = \begin{bmatrix} \frac{\partial P}{\partial q_1} \\ \frac{\partial P}{\partial q_2} \\ \frac{\partial P}{\partial q_3} \end{bmatrix}$$

$$P = m_1 g l_{c1} + m_2 l_{c2} g \sin q_2 + m_2 g l_1 + m_3 g l_1 + m_3 g l_2 \sin q_2 + m_3 g l_{c3} \sin q_3$$

$$g(q) = \begin{bmatrix} 0 \\ m_2 l_{c2} g \cos q_2 + m_3 g l_2 \cos q_2 \\ m_3 g l_{c3} \cos q_3 \end{bmatrix}$$

so that

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

all elements are defined analytically.

$$\ddot{q} = D^{-1}(q)\left(\tau - C(q,\dot{q})\dot{q} + g(q)\right)$$

9.2 Matlab derivation for EoM

```
syms q1 q2 q3 l1 l2 l3 qdot1 qdot2 qdot3 w1 w2 w3 lc1 lc2 lc3 g
assume(q1, 'real')
assume(q2, 'real')
assume(q3, 'real')
assume(lc1, 'real')
assume(lc2, 'real')
assume(lc3, 'real')
assume(l1, 'real')
assume(l2, 'real')
assume(l3, 'real')
l1 = [106];
l2 = [98.7];
l3 = [140.25];


p0=[0;0;0]
```

```
p0 = 3×1
     0
     0
     0
```

```
p1=str2sym('[0 ; 0 ;l1]')
```

```
p1 =
```

$$\begin{pmatrix} 0 \\ 0 \\ l_1 \end{pmatrix}$$

```
p2=str2sym('[l2*cos(q1)*cos(q2); l2*sin(q1)*cos(q2);l2*sin(q2)+l1]')
```

```
p2 =
```

$$\begin{pmatrix} l_2 \cos(q_1) \cos(q_2) \\ l_2 \cos(q_2) \sin(q_1) \\ l_1 + l_2 \sin(q_2) \end{pmatrix}$$

```
p3=str2sym('[l3*cos(q2+q3)*cos(q1)+l2*cos(q1)*cos(q2); l3*cos(q2+q3)*sin(q1)+l2*sin(q1)*cos(q2
```

```
p3 =
```

$$\begin{pmatrix} l_3 \cos(q_2 + q_3) \cos(q_1) + l_2 \cos(q_1) \cos(q_2) \\ l_3 \cos(q_2 + q_3) \sin(q_1) + l_2 \cos(q_2) \sin(q_1) \\ l_1 + l_3 \sin(q_2 + q_3) + l_2 \sin(q_2) \end{pmatrix}$$

```
lc1=38.29 ;
lc2=24.45 ;
lc3=37.9 ;
```

```
l1 = 106;
l2 = 98.7;
l3 = 140.25;
w12=[1 0 0; 0 0 0; 0 0 0];%square of angular velocity of link1
w22=[sin(q1)^2 -sin(q1)*cos(q1) 0; %square of angular velocity of link2
    -sin(q1)*cos(q1) cos(q1)^2 0;
    0 0 0];
m1=258.22;
I1=[290212.21 0 0;
    0 275331.77 0;
    0 0 509323.35];
m2=57.56;
I2=[7409.27 0 0;
    0 52876.31 0;
   0 0 50680.51];
m3=47.17;
I3=[3801.21 0 0;
    0 87085.96 0 ;
    0 0 86685.11];
Jv1=str2sym('[ -lc1*cos(q1), 0, 0; -lc1*sin(q1), 0, 0;          0, 0, 0]')
```

Jv1 =

$$\begin{pmatrix} -\mathrm{lc}_1\cos(q_1) & 0 & 0 \\ -\mathrm{lc}_1\sin(q_1) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

```
Jv2=str2sym('[                -cos(q1)*(l1 + lc2*sin(q2)),                        -lc2*cos(q
Jv3=str2sym('[-cos(q1)*(l1 + lc3*sin(q2 + q3) + l2*sin(q2)), -cos(q1)*(lc3*sin(q2 + q3) + l2*s
D=m1*Jv1'*Jv1+m2*Jv2'*Jv2+m3*Jv3'*Jv3+I1*w12+(I1+I2)*w22;

d=[(simplify(D(1,1))) (simplify(D(1,2))) (simplify(D(1,3))); (simplify(D(2,1))) (simplify(D(2,
q=str2sym('[q1 q2 q3]')
```

q = $(q_1 \quad q_2 \quad q_3)$

```
c=str2sym('[0 0 0;0 0 0;0 0 0]');
    for j=1:3
        for k=1:3
            for i=1:3
            c(k,j)=c(k,j)+(diff(d(k,j),q(i))+diff(d(k,i),q(j))+diff(d(i,j),q(k)))/2;
        end
        end
    end
gr=9810;
g=str2sym('[0; m2*lc2*gr*cos(q2)+m3*gr*l2*cos(q2);m3*g*lc3*cos(q3)]')
```

g =

$$\begin{pmatrix} 0 \\ \mathrm{gr}\, l_2\, m_3 \cos(q_2) + \mathrm{gr}\, \mathrm{lc}_2\, m_2 \cos(q_2) \\ g\, \mathrm{lc}_3\, m_3 \cos(q_3) \end{pmatrix}$$

```
for i=1:3
    for j=1:3
        D(i,j)
    end
end
```

ans =

$$\frac{5113098092748473\sin(q_1)^2}{17179869184} + \frac{12911\,\mathrm{lc}_1^2\cos(q_1)^2}{50} + \frac{1439\cos(q_1)^2\,\sigma_4}{25} + \frac{12911\,\mathrm{lc}_1^2\sin(q_1)^2}{50} + \frac{1439\sin(q_1)^2\,\sigma_4}{25} + (\mathrm{lc}_2\cos$$

where

$$\sigma_1 = \mathrm{lc}_3\cos(q_2+q_3)\sin(q_1) + l_2\cos(q_2)\sin(q_1)$$

$$\sigma_2 = \mathrm{lc}_3\cos(q_2+q_3)\cos(q_1) + l_2\cos(q_1)\cos(q_2)$$

$$\sigma_3 = (l_1 + \mathrm{lc}_3\sin(q_2+q_3) + l_2\sin(q_2))^2$$

$$\sigma_4 = (l_1 + \mathrm{lc}_2\sin(q_2))^2$$

ans =

$$(\mathrm{lc}_2\cos(q_2)\cos(q_1)^2 + \mathrm{lc}_2\cos(q_2)\sin(q_1)^2)\left(\frac{1439\,\mathrm{lc}_2\cos(q_2)\cos(q_1)^2}{25} + \frac{1439\,\mathrm{lc}_2\cos(q_2)\sin(q_1)^2}{25}\right) - \frac{5113098092748}{17179}$$

where

$$\sigma_1 = \mathrm{lc}_3\cos(q_2+q_3)\sin(q_1) + l_2\cos(q_2)\sin(q_1)$$

$$\sigma_2 = \mathrm{lc}_3\cos(q_2+q_3)\cos(q_1) + l_2\cos(q_1)\cos(q_2)$$

$$\sigma_3 = l_1 + \mathrm{lc}_2\sin(q_2)$$

$$\sigma_4 = l_1 + \sigma_6 + l_2\sin(q_2)$$

$$\sigma_5 = \sigma_6 + l_2\sin(q_2)$$

$$\sigma_6 = \mathrm{lc}_3\sin(q_2+q_3)$$

ans =

$$\left(\frac{4717\cos(q_1)\ (\mathrm{lc}_3\cos(q_2+q_3)\cos(q_1)+l_2\cos(q_1)\cos(q_2))}{100}+\frac{4717\sin(q_1)\ (\mathrm{lc}_3\cos(q_2+q_3)\sin(q_1)+l_2\cos(q_2)\sin(q_1)}{100}\right.$$

where

$$\sigma_1 = l_1+\mathrm{lc}_3\sin(q_2+q_3)+l_2\sin(q_2)$$

ans =

$$\left(\mathrm{lc}_2\cos(q_2)\cos(q_1)^2+\mathrm{lc}_2\cos(q_2)\sin(q_1)^2\right)\left(\frac{1439\,\mathrm{lc}_2\cos(q_2)\cos(q_1)^2}{25}+\frac{1439\,\mathrm{lc}_2\cos(q_2)\sin(q_1)^2}{25}\right)-\frac{5638571879531}{17179}$$

where

$$\sigma_1 = \mathrm{lc}_3\cos(q_2+q_3)\sin(q_1)+l_2\cos(q_2)\sin(q_1)$$

$$\sigma_2 = \mathrm{lc}_3\cos(q_2+q_3)\cos(q_1)+l_2\cos(q_1)\cos(q_2)$$

$$\sigma_3 = l_1+\mathrm{lc}_2\sin(q_2)$$

$$\sigma_4 = l_1+\sigma_6+l_2\sin(q_2)$$

$$\sigma_5 = \sigma_6+l_2\sin(q_2)$$

$$\sigma_6 = \mathrm{lc}_3\sin(q_2+q_3)$$

ans =

$$\frac{141009278180786183\cos(q_1)^2}{429496729600}+\frac{10473\sin(q_1)^2}{100}+\frac{4717\cos(q_1)^2\sigma_3}{100}+\left(\mathrm{lc}_2\cos(q_2)\cos(q_1)^2+\mathrm{lc}_2\cos(q_2)\sin(q_1)^2\right)\left(\frac{143\right.$$

where

$$\sigma_1 = \mathrm{lc}_3\cos(q_2+q_3)\sin(q_1)+l_2\cos(q_2)\sin(q_1)$$

$$\sigma_2 = \mathrm{lc}_3\cos(q_2+q_3)\cos(q_1)+l_2\cos(q_1)\cos(q_2)$$

$$\sigma_3 = (\mathrm{lc}_3\sin(q_2+q_3)+l_2\sin(q_2))^2$$

ans =

$$\frac{4717\cos(q_1)^2}{100}+\left(\frac{4717\cos(q_1)\ (\mathrm{lc}_3\cos(q_2+q_3)\cos(q_1)+l_2\cos(q_1)\cos(q_2))}{100}+\frac{4717\sin(q_1)\ (\mathrm{lc}_3\cos(q_2+q_3)\sin(q_1)+\right.$$

where

$$\sigma_1 = \mathrm{lc}_3\sin(q_2+q_3)+l_2\sin(q_2)$$

ans =

$$(\cos(q_1)\ (\mathrm{lc}_3\cos(q_2+q_3)\cos(q_1)+l_2\cos(q_1)\cos(q_2))+\sin(q_1)\ (\mathrm{lc}_3\cos(q_2+q_3)\sin(q_1)+l_2\cos(q_2)\sin(q_1)))\ \left(\frac{4717\,l}{}\right.$$

where

$$\sigma_1 = l_1+\mathrm{lc}_3\sin(q_2+q_3)+l_2\sin(q_2)$$

ans =

$$\frac{4717\cos(q_1)^2}{100}+(\cos(q_1)\ (\mathrm{lc}_3\cos(q_2+q_3)\cos(q_1)+l_2\cos(q_1)\cos(q_2))+\sin(q_1)\ (\mathrm{lc}_3\cos(q_2+q_3)\sin(q_1)+l_2\cos(q_2)\ s$$

where

$$\sigma_1 = \mathrm{lc}_3\sin(q_2+q_3)+l_2\sin(q_2)$$

ans =

$$\frac{4717\cos(q_1)^2}{100}+\frac{4717\sin(q_1)^2}{100}+\left(\mathrm{lc}_3\cos(q_2+q_3)\cos(q_1)^2+\mathrm{lc}_3\cos(q_2+q_3)\sin(q_1)^2\right)\ \left(\frac{4717\,\mathrm{lc}_3\cos(q_2+q_3)\cos(q_1)^2}{100}\right.$$

where

$$\sigma_1 = \sin(q_2+q_3)^2$$

```
for i=1:3
    for j=1:3
        c(i,j)
    end
end
```

ans =

$$\frac{15339294278245419\sin(2\,q_1)}{34359738368}-\frac{1343958746535035\cos(2\,q_1)}{4294967296}+\frac{4717\,l_1\,l_2\cos(q_2)}{100}+\frac{1439\,l_1\,\mathrm{lc}_2\cos(q_2)}{25}-\frac{4717\,l_2\,\mathrm{lc}_3\sin(q_3}{100}$$

ans =

$$\frac{4717\,l_1\,l_2\cos(q_2)}{50}-\frac{5638571879531807\cos(q_1)\sin(q_1)}{17179869184}-\frac{5113098092748473\cos(2\,q_1)}{17179869184}+\frac{2878\,l_1\,\mathrm{lc}_2\cos(q_2)}{25}-\frac{4717\,l_2\,\mathrm{lc}_3\ s}{100}$$

ans =

$$\frac{4717\,\mathrm{lc}_3\ (l_1\cos(q_2+q_3)-l_2\sin(q_3))}{100}-\frac{4717\,l_2\,\mathrm{lc}_3\sin(q_3)}{50}+\frac{4717\,l_1\,\mathrm{lc}_3\cos(q_2+q_3)}{50}$$

ans =

$$\frac{4717\,l_1\,l_2\cos(q_2)}{50}-\frac{5638571879531807\cos(q_1)\sin(q_1)}{17179869184}-\frac{5638571879531807\cos(2\,q_1)}{17179869184}+\frac{2878\,l_1\,\mathrm{lc}_2\cos(q_2)}{25}-\frac{4717\,l_2\,\mathrm{lc}_3\ s}{100}$$

ans =

$$\frac{4717\,l_1\,l_2\cos(q_2)}{100}-\frac{5638571879531807\cos(q_1)\sin(q_1)}{17179869184}+\frac{1439\,l_1\,\mathrm{lc}_2\cos(q_2)}{25}-\frac{4717\,l_2\,\mathrm{lc}_3\sin(q_3)}{100}+\frac{4717\,l_1\,\mathrm{lc}_3\cos(q_2+q_3)}{100}$$

ans =

$$\frac{4717\,l_1\,\mathrm{lc}_3\cos(q_2+q_3)}{100}-\frac{14151\,l_2\,\mathrm{lc}_3\sin(q_3)}{100}$$

ans =

$$\frac{4717\,\mathrm{lc}_3\,(l_1\cos(q_2+q_3)-l_2\sin(q_3))}{100}-\frac{4717\,l_2\,\mathrm{lc}_3\sin(q_3)}{50}+\frac{4717\,l_1\,\mathrm{lc}_3\cos(q_2+q_3)}{50}$$

ans =

$$\frac{4717\,l_1\,\mathrm{lc}_3\cos(q_2+q_3)}{100}-\frac{14151\,l_2\,\mathrm{lc}_3\sin(q_3)}{100}$$

ans =

$$\frac{4717\,\mathrm{lc}_3\,(l_1\cos(q_2+q_3)-l_2\sin(q_3))}{100}-\frac{4717\,l_2\,\mathrm{lc}_3\sin(q_3)}{100}$$

```
% ddot=str2sym('[0 0 0;0 0 0;0 0 0]');
%     for j=1:3
%         for k=1:3
%             for i=1:3
%             ddot(k,j)=ddot(k,j)+(diff(d(k,j),q(i)));
%         end
%     end
%     end
% N=simplify(ddot)-2*simplify(c)
%
% tf = issymmetric(N,'skew')
```