ENGG 5402 Course Project: Simulation of PUMA 560 Manipulator

ZHENG Fan, 1155051778

mrzhengfan@gmail.com

April 5, 2015

0. Preface

This project is to derive programs for simulation of inverse dynamics and control based on the model of PUMA 560 manipulator, with the last 3 joints neglected.

PUMA 560 is widely used in academic and industrial application, and has been modeled with several D-H schemes with slightly different measured parameters [1]. Here we adopt the scheme and parameters of B. Armstrong [2], with the configuration and parameters shown below.

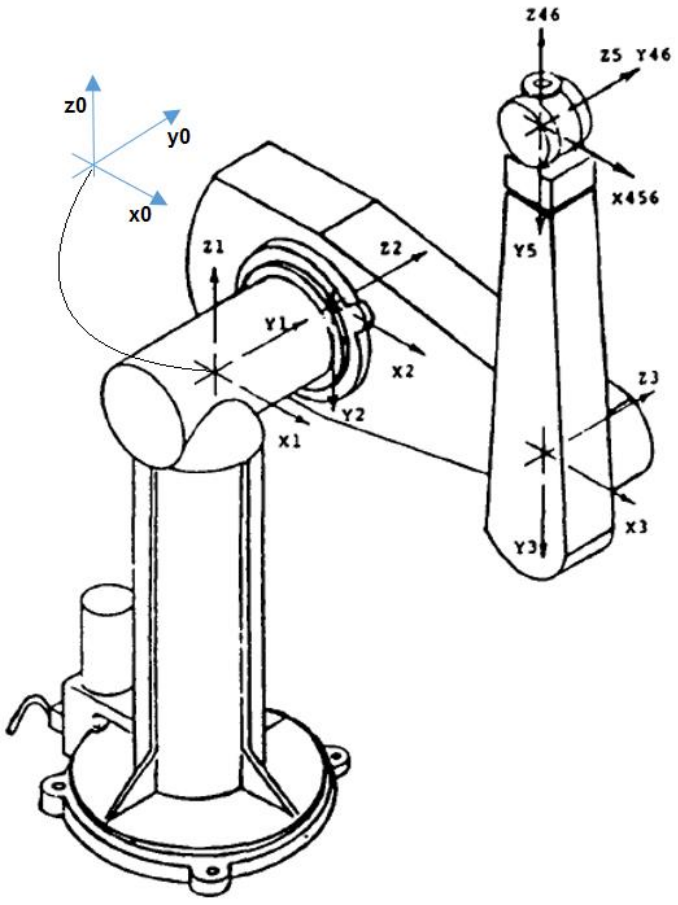

Figure 1 D-H model

Armstrong		
α_{i-1}	A_{i-1}	D_i
0	0	0
-90	0	243.5
0	431.8	-93.4
90	-20.3	433.1
-90	0	0
90	0	0

Table 1 D-H parameters [mm]

Param	*Armstrong*
m_1	-
m_2	17.40
m_3	4.80
m_4	0.82
m_5	0.35
m_6	0.09

Table 2 Link mass [kg]

Param	*Armstrong*
s_{x1}	-
s_{y1}	-
s_{z1}	-
s_{x2}	68
s_{y2}	6
s_{z2}	-16
s_{x3}	0
s_{y3}	-70
s_{z3}	14
s_{x4}	0
s_{y4}	0
s_{z4}	-19
s_{x5}	0
s_{y5}	0
s_{z5}	0
s_{x6}	0
s_{y6}	0
s_{z6}	32

Table 3 Link center of gravity [mm]

Param	*Armstrong*
I_{xx_1}	-
I_{yy_1}	-
I_{zz_1}	0.350
I_{xx_2}	0.130
I_{yy_2}	0.524
I_{zz_2}	0.539
I_{xx_3}	66.0 e-3
I_{yy_3}	12.5 e-3
I_{zz_3}	86.0 e-3
I_{xx_4}	1.80 e-3
I_{yy_4}	1.80 e-3
I_{zz_4}	1.30 e-3
I_{xx_5}	300 e-6
I_{yy_5}	300 e-6
I_{zz_5}	400 e-6
I_{xx_6}	150 e-6
I_{yy_6}	150 e-6
I_{zz_6}	40 e-6

Table 4 Moment of inertia about COG [kg m^2]

1. Inverse Dynamics with Newton-Euler Formulation

In this part we make a program to calculate its inverse dynamics using the Newton-Euler formulation and plot the input torques at joints when the first three joints move in trajectories $q_i = h_i \sin \omega_i t$. Here we select $h_1 = h_2 = h_3 = 1, \omega_1 = \omega_2 = \omega_3 = 1\ rad/s$.

Derive kinematic equations (all expressed in the base frame {x0 y0 z0}):

$$q_1 = q_2 = q_3 = \sin t = q$$

$$\dot{q}_1 = \dot{q}_2 = \dot{q}_3 = \cos t = \dot{q}$$

$$\ddot{q}_1 = \ddot{q}_2 = \ddot{q}_3 = -\sin t = \ddot{q}$$

Write $s_i = \sin q_i$, $c_i = \cos q_i$.

Rotation matrix:

$$ {}^0_1R = \begin{bmatrix} c_1 & -s_1 & 0 \\ s & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, {}^0_2R = \begin{bmatrix} c_1c_2 & -c_1s_2 & -s_1 \\ s_1c_2 & -s_1s_2 & c_1 \\ -s_2 & -c_2 & 0 \end{bmatrix}, {}^0_3R = \begin{bmatrix} c_1c_{2+3} & -c_1s_{2+3} & -s_1 \\ s_1c_{2+3} & -s_1s_{2+3} & c_1 \\ -s_{2+3} & -c_{2+3} & 0 \end{bmatrix} $$

Corresponding z axis:

$$ \boldsymbol{z_1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \boldsymbol{z_2} = \boldsymbol{z_3} = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} $$

Relative position of frame origin and center of mass:

$$ \boldsymbol{s_1} = 0.243 \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} $$

$$ \boldsymbol{s_2} = {}^0_2R \begin{bmatrix} 0.432 \\ 0 \\ -0.093 \end{bmatrix}, \boldsymbol{r_2} = {}^0_2R \begin{bmatrix} 0.068 \\ 0.006 \\ -0.016 \end{bmatrix} $$

$$ \boldsymbol{r_3} = {}^0_3R \begin{bmatrix} 0 \\ -0.07 \\ 0.014 \end{bmatrix} $$

Angular velocity and differentiation:

$$ \boldsymbol{\omega_1} = \begin{bmatrix} 0 \\ 0 \\ \dot{q} \end{bmatrix}, \dot{\boldsymbol{\omega}}_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{q} \end{bmatrix} $$

$$ \boldsymbol{\omega_2} = \boldsymbol{\omega_1} + \boldsymbol{z_2}\dot{q} = \dot{q} \begin{bmatrix} -s_1 \\ c_1 \\ 1 \end{bmatrix}, \dot{\boldsymbol{\omega}}_2 = \ddot{q} \begin{bmatrix} -s_1 \\ c_1 \\ 1 \end{bmatrix} + \dot{q}^2 \begin{bmatrix} -c_1 \\ -s_1 \\ 0 \end{bmatrix} $$

$$ \boldsymbol{\omega_3} = \boldsymbol{\omega_2} + \boldsymbol{z_3}\dot{q} = \dot{q} \begin{bmatrix} -2s_1 \\ 2c_1 \\ 1 \end{bmatrix}, \dot{\boldsymbol{\omega}}_3 = \ddot{q} \begin{bmatrix} -2s_1 \\ 2c_1 \\ 1 \end{bmatrix} + \dot{q}^2 \begin{bmatrix} -2c_1 \\ -2s_1 \\ 0 \end{bmatrix} $$

Velocity of frame origin:

$$ \boldsymbol{v_1} = \boldsymbol{0} $$

$$ \boldsymbol{v_2} = \boldsymbol{v_1} + \boldsymbol{\omega_1} \times \boldsymbol{s_1} $$

$$ \boldsymbol{v_3} = \boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{s_2} $$

Acceleration of frame origin:

$$ \boldsymbol{a_1} = \boldsymbol{0} $$

$$ \boldsymbol{a_2} = \boldsymbol{a_1} + \dot{\boldsymbol{\omega}}_1 \times \boldsymbol{s_1} + \boldsymbol{\omega_1} \times (\boldsymbol{\omega_1} \times \boldsymbol{s_1}) $$

$$ \boldsymbol{a_3} = \boldsymbol{a_2} + \dot{\boldsymbol{\omega}}_2 \times \boldsymbol{s_2} + \boldsymbol{\omega_2} \times (\boldsymbol{\omega_2} \times \boldsymbol{s_2}) $$

Acceleration of center of mass:

$$ \boldsymbol{a_{c2}} = \boldsymbol{a_2} + \dot{\boldsymbol{\omega}}_2 \times \boldsymbol{r_2} + \boldsymbol{\omega_2} \times (\boldsymbol{\omega_2} \times \boldsymbol{r_2}) $$

$$ \boldsymbol{a_{c3}} = \boldsymbol{a_3} + \dot{\boldsymbol{\omega}}_3 \times \boldsymbol{r_3} + \boldsymbol{\omega_3} \times (\boldsymbol{\omega_3} \times \boldsymbol{r_3}) $$

Then derive the backward dynamic equations:

Inertia tensor matrix:

$$\boldsymbol{I_1} = \begin{bmatrix} * & 0 & 0 \\ 0 & * & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix}, \boldsymbol{I_2} = {}^0_2R \begin{bmatrix} I_{xx2} & 0 & 0 \\ 0 & I_{yy2} & 0 \\ 0 & 0 & I_{zz2} \end{bmatrix} {}^0_2R^T, \boldsymbol{I_3} = {}^0_3R \begin{bmatrix} I_{xx3} & 0 & 0 \\ 0 & I_{yy3} & 0 \\ 0 & 0 & I_{zz3} \end{bmatrix} {}^0_3R^T$$

Force and torque at joint:

$$\boldsymbol{f_3} = m_3\boldsymbol{a_{c3}} + \begin{bmatrix} 0 \\ 0 \\ m_3 g \end{bmatrix}$$

$$\boldsymbol{n_3} = \boldsymbol{I_3}\dot{\boldsymbol{\omega}}_3 + \boldsymbol{\omega_3} \times \boldsymbol{I_3}\boldsymbol{\omega_3} + \boldsymbol{r_3} \times \boldsymbol{f_3}$$

$$\tau_3 = \boldsymbol{z_3^T}\boldsymbol{n_3}$$

$$\boldsymbol{f_2} = m_2\boldsymbol{a_{c2}} + \boldsymbol{f_3} + \begin{bmatrix} 0 \\ 0 \\ m_2 g \end{bmatrix}$$

$$\boldsymbol{n_2} = \boldsymbol{I_2}\dot{\boldsymbol{\omega}}_2 + \boldsymbol{\omega_2} \times \boldsymbol{I_2}\boldsymbol{\omega_2} + \boldsymbol{n_3} + \boldsymbol{r_2} \times \boldsymbol{f_2} + (\boldsymbol{s_2} - \boldsymbol{r_2}) \times \boldsymbol{f_3}$$

$$\tau_2 = \boldsymbol{z_2^T}\boldsymbol{n_2}$$

$$\boldsymbol{n_1} = \boldsymbol{I_1}\dot{\boldsymbol{\omega}}_1 + \boldsymbol{n_2} + \boldsymbol{s_1} \times \boldsymbol{f_2}$$

$$\tau_1 = \boldsymbol{z_1^T}\boldsymbol{n_1}$$

Finally τ_1, τ_2, τ_3 are the input torques at joints. Write program in MATLAB (see Appendix-1) and plot them as below:

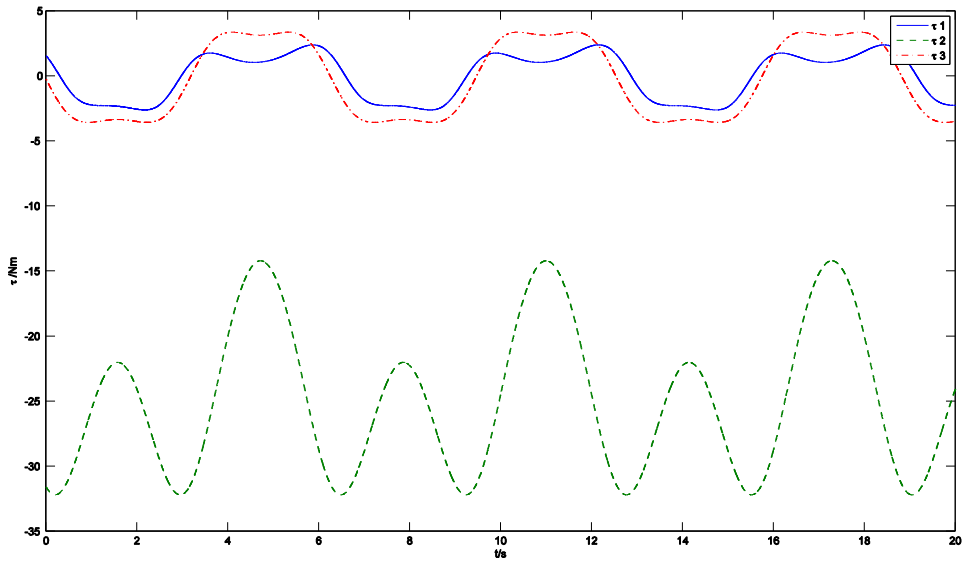

Figure 2 Inverse dynamic simulation result

2. Controller simulation

In this part we make programs to simulate the performance of the robot under the control of the PD controller, the PID controller, and the PD plus gravity compensator controller, respectively.

First we derive the Lagrange formulation of robot dynamics. Take {x0 y0 z0} as base frame and plane x0-y0 as zero potential energy plane.

Generalized coordinates $\boldsymbol{q} = [q_1\ q_2\ q_3]^T$.

Calculating velocities:

$$\boldsymbol{\omega_1} = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 \end{bmatrix}, \dot{\boldsymbol{\omega}}_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{q}_1 \end{bmatrix}$$

$$\boldsymbol{\omega_2} = \boldsymbol{\omega_1} + \boldsymbol{z_2}\dot{q}_2 = \begin{bmatrix} -\dot{q}_2 s_1 \\ \dot{q}_2 c_1 \\ \dot{q}_1 \end{bmatrix}, \dot{\boldsymbol{\omega}}_2 = \ddot{q} \begin{bmatrix} -s_1 \\ c_1 \\ 1 \end{bmatrix} + \dot{q}^2 \begin{bmatrix} -c_1 \\ -s_1 \\ 0 \end{bmatrix}$$

$$\boldsymbol{\omega_3} = \boldsymbol{\omega_2} + \boldsymbol{z_3}\dot{q} = \begin{bmatrix} -(\dot{q}_2 + \dot{q}_3) s_1 \\ (\dot{q}_2 + \dot{q}_3) c_1 \\ \dot{q}_1 \end{bmatrix}, \dot{\boldsymbol{\omega}}_3 = \ddot{q} \begin{bmatrix} -2s_1 \\ 2c_1 \\ 1 \end{bmatrix} + \dot{q}^2 \begin{bmatrix} -2c_1 \\ -2s_1 \\ 0 \end{bmatrix}$$

$$\boldsymbol{v_{c2}} = \boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} = \dot{q}_1 \begin{bmatrix} -0.984c_1 - 0.068s_1c_2 + 0.006s_1s_2 \\ -0.984s_1 + 0.068c_1c_2 - 0.006c_1s_2 \\ 0 \end{bmatrix} + \dot{q}_2 \begin{bmatrix} -0.068c_1s_2 - 0.006c_1c_2 \\ -0.068s_1s_2 - 0.006s_1c_2 \\ -0.068c_2 + 0.006s_2 \end{bmatrix}$$

$$\boldsymbol{v_{c3}} = \boldsymbol{v_3} + \boldsymbol{\omega_3} \times \boldsymbol{r_3} = \dot{q}_1 \begin{bmatrix} -0.921c_1 - 0.432s_1c_2 - 0.07s_1s_{2+3} \\ -0.921s_1 + 0.432c_1c_2 + 0.07c_1s_{2+3} \\ 0 \end{bmatrix}$$

$$+\dot{q}_2 \begin{bmatrix} -0.432c_1s_2 + 0.07c_1c_{2+3} \\ -0.432s_1s_2 + 0.07s_1c_{2+3} \\ -0.432c_2 - 0.07s_{2+3} \end{bmatrix} + \dot{q}_3 \begin{bmatrix} 0.07c_1c_{2+3} \\ 0.07s_1c_{2+3} \\ -0.07s_{2+3} \end{bmatrix}$$

Kinetic energy:

$$K = \frac{1}{2}(I_{zz1}\omega_1^2 + I_{zz2}\omega_2^2 + I_{zz3}\omega_3^2 + m_2v_{c2}^2 + m_3v_{c3}^2)$$

$$\approx (11 + 0.45c_2^2)\dot{q}_1^2 + 0.83\dot{q}_2^2 + 0.055\dot{q}_3^2 + (1.17 + 1.92s_2)\dot{q}_1\dot{q}_2 - 0.3c_{2+3}\dot{q}_1\dot{q}_3 - 0.29\dot{q}_3\dot{q}_2$$

Potential energy:

$$U = -0.068m_2gc_2 + m_3g[-0.432c_2 + 0.07c(q_2 + q_3)]$$

$$= -31.9c_2 + 3.3c_3c_2 - 3.3s_2s_3$$

Lagrangian $L = K - U$.

$$\frac{\partial L}{\partial \boldsymbol{q}} = \begin{bmatrix} 0 \\ -91.9s_2 + 3.3s_2c_3 + 3.3c_2s_3 - 0.9c_2s_2\dot{q}_1^2 + 1.92c_2\dot{q}_1\dot{q}_2 + 0.3s_{2+3}\dot{q}_1\dot{q}_3 \\ 3.3s_3c_2 + 3.3s_2c_3 + 0.3s_{2+3}\dot{q}_1\dot{q}_3 \end{bmatrix}$$

$$\frac{\partial L}{\partial \dot{\boldsymbol{q}}} = \begin{bmatrix} (22 + 0.9c_2^2)\dot{q}_1 + (1.17 + 1.92s_2)\dot{q}_2 - 0.3c_{2+3}\dot{q}_3 \\ 1.66\dot{q}_2 + (1.17 + 1.92s_2)\dot{q}_1 - 0.29\dot{q}_3 \\ 0.11\dot{q}_3 - 0.3c_{2+3}\dot{q}_1 - 0.29\dot{q}_2 \end{bmatrix}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}}$$

$$= \begin{bmatrix} (22 + 0.9c_2^2)\ddot{q}_1 + (1.17 + 1.92s_2)\ddot{q}_2 - 0.3c_{2+3}\ddot{q}_3 - 1.8c_2 s_2 \dot{q}_1\dot{q}_2 + 1.92c_2\dot{q}_2^2 + 0.3s_{2+3}\dot{q}_2\dot{q}_3 + 0.3s_{2+3}\dot{q}_3^2 \\ 1.66\ddot{q}_2 + (1.17 + 1.92s_2)\ddot{q}_1 - 0.29\ddot{q}_3 + 1.92c_2\dot{q}_1\dot{q}_2 \\ 0.11\ddot{q}_3 - 0.3c_{2+3}\ddot{q}_1 - 0.29\ddot{q}_2 + 0.3s_{2+3}\dot{q}_1\dot{q}_2 + 0.3s_{2+3}\dot{q}_1\dot{q}_3 \end{bmatrix}$$

Dynamic equation of the manipulator:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L}{\partial \boldsymbol{q}} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

Based on this dynamic equation, we develop several controller to simulate the robot's performance. Here we consider only position control problem, and select the desired position as

$$\boldsymbol{q_d} = \begin{bmatrix} \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} \end{bmatrix}^{\boldsymbol{T}}$$

1) PD controller:

PD controller input is $\boldsymbol{\tau} = -\boldsymbol{A}(\boldsymbol{q} - \boldsymbol{q_d}) - \boldsymbol{B}\dot{\boldsymbol{q}}$. Choose proper matrix **A** and **B**, construct state space expression and use MATLAB program with ode method (see Appendix-2) to simulate the performance. The result is shown below.

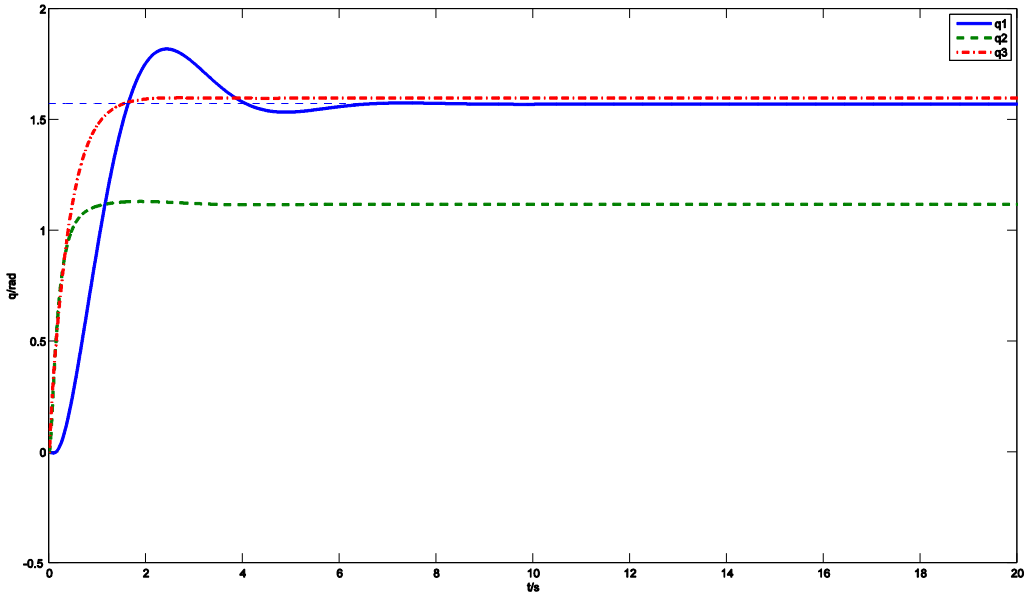

Figure 3 PD control simulation result

It can be seen from the figure that all the 3 joints is stable. However, only joint 1 converge to the desired position, while there exit offset error between desired and actual position in both joint 2 and 3.

The offset error is caused by the existence of gravity. We can kill this error by PID control or PD feedback plus gravity compensation.

2) PID controller

PID controller input is $\boldsymbol{\tau} = -\boldsymbol{A}(\boldsymbol{q} - \boldsymbol{q_d}) - \boldsymbol{B}\dot{\boldsymbol{q}} - \boldsymbol{C}\int_0^t (\boldsymbol{q} - \boldsymbol{q_d})dt$. Choose proper matrix **A, C** and **B**, construct state space expression and use MATLAB program with ode method (see Appendix-3) to simulate the performance. The result is shown below.

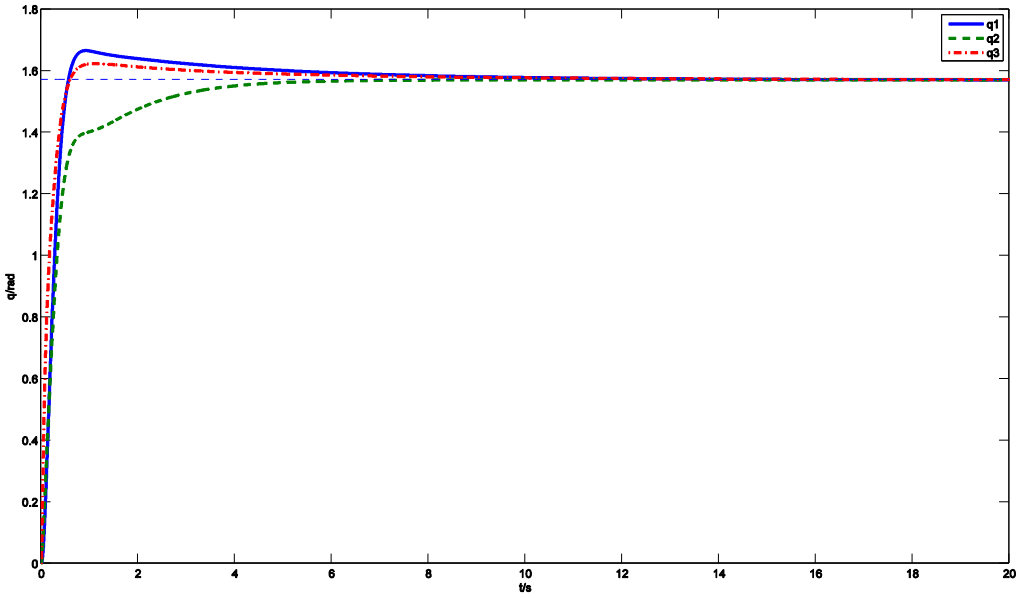

Figure 4 PID control simulation result

As the figure shows, all 3 joints converge to desired values and no offset error exists.

3) PD plus gravity compensator controller

PD plus gravity compensator controller input is $\boldsymbol{\tau} = -\boldsymbol{A}(\boldsymbol{q} - \boldsymbol{q_d}) - \boldsymbol{B}\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q})$. Choose proper matrix **A** and **B**, construct state space expression and use MATLAB program with ode method (see Appendix-4) to simulate the performance. The result is shown below.

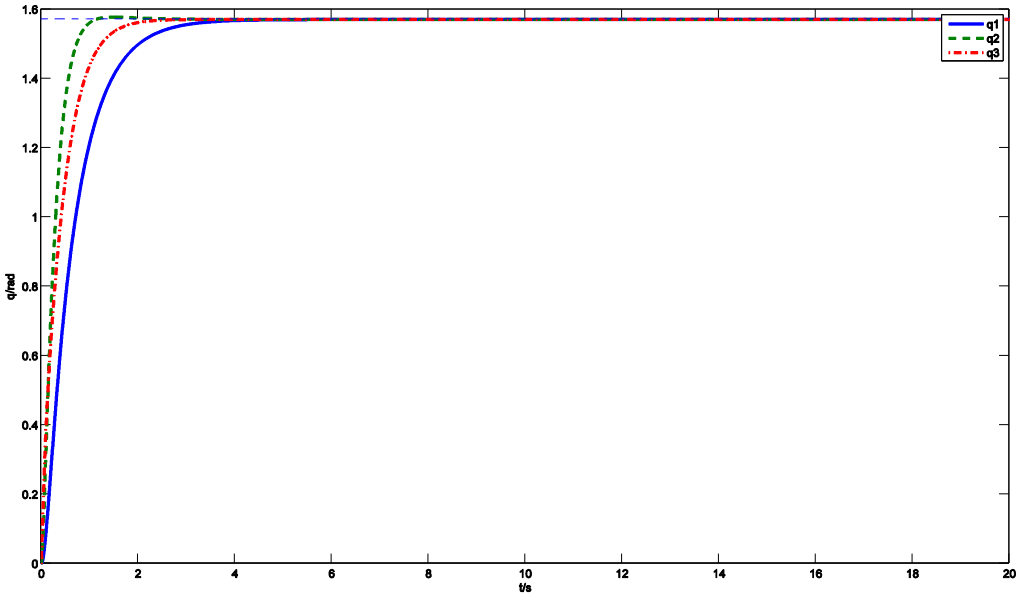

Figure 5 PD control plus gravity compensation simulation result

The 3 joints also converge to desired values with no offset error, and with satisfactory transition process.

3. Computed torque control

In this part we make a program to simulate the performance of the robot under the control of the computed torque control method.

We select desired trajectory as $\boldsymbol{q_d} = \begin{bmatrix} \frac{\pi}{2} + \sin t \\ \frac{\pi}{2} + \sin t \\ \frac{\pi}{2} + \sin t \end{bmatrix}$. The control input is

$$\boldsymbol{\tau = C(q,\dot{q}) + G(q) + H(q)(\ddot{q}_d - A(\dot{q} - \dot{q}_d) - B(q - q_d))}$$

The closed loop system is then

$$\boldsymbol{\ddot{q} = \ddot{q}_d - A(\dot{q} - \dot{q}_d) - B(q - q_d)}$$

Choose proper matrix **A** and **B**, construct state space expression and use MATLAB program with ode method (see Appendix-5) to simulate the performance. The result is shown below.

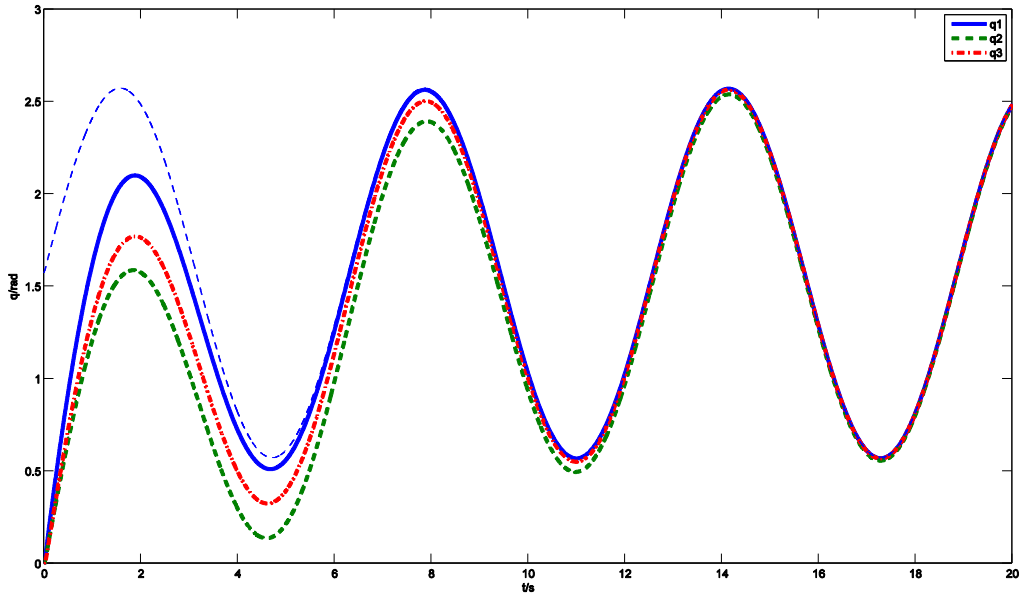

Figure 6 Computed torque control simulation result: joint position

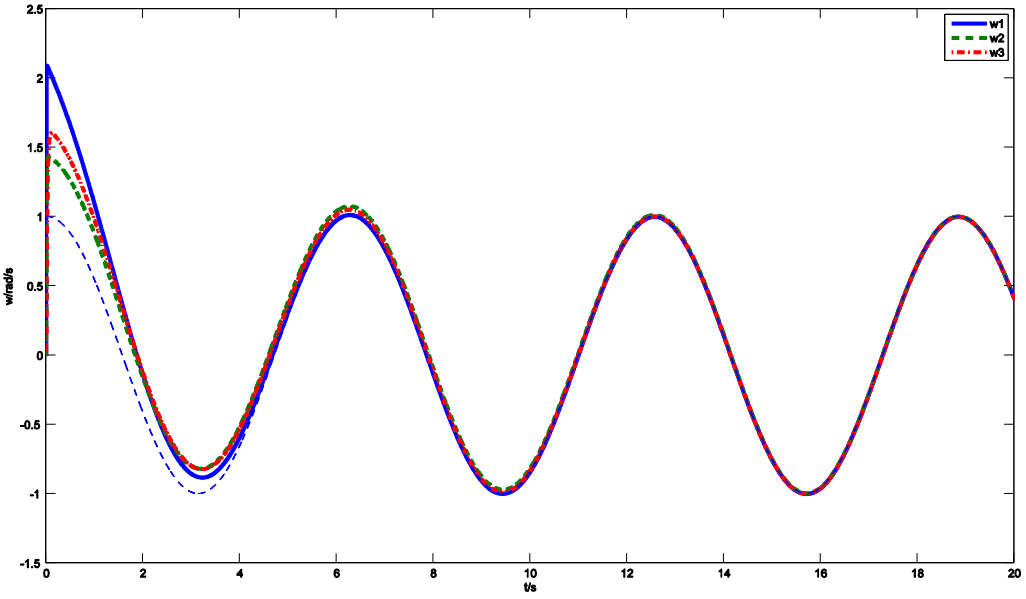

Figure 7 Computed torque control simulation result: joint velocity

Preferences

[1] Corke P I, Armstrong-Helouvry B. A search for consensus among model parameters reported for the PUMA 560 robot[C]//Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on. IEEE, 1994: 1608-1613.

[2] Armstrong B, Khatib O, Burdick J. The explicit dynamic model and inertial parameters of the PUMA 560 arm[C]//Robotics and Automation. Proceedings. 1986 IEEE International Conference on. IEEE, 1986, 3: 510-518.

Appendix

All MATLAB code and this report are available in http://izhengfan.github.io/puma.html.

1. Inverse dynamics simulation code.

```
function main_puma01

tau1 = zeros(1,20001);
tau2 = zeros(1,20001);
tau3 = zeros(1,20001);

for i = 0:1:20000;

t = i*0.001;
q = sin(t);
qdot = cos(t);
qddot = -sin(t);

cq = cos(q);
sq = sin(q);
c2q = cos(q+q);
s2q = sin(q+q);

R01 = [cq,-sq,0;sq,cq,0;0,0,1];
R02 = [cq*cq,-cq*sq,-sq;sq*cq,-sq*sq,cq;-sq,-cq,0];
R03 = [cq*c2q,-cq*s2q,-sq;sq*c2q,-sq*s2q,cq;-s2q,-c2q,0];

z1 = [0;0;1];
z2 = [-sq;cq;0];
z3 = z2;

s1 = 0.243*z2;
s2 = R02*[0.432;0;-0.093];
r2 = R02*[0.068;0.006;-0.016];
r3 = R03*[0;-0.07;0.014];

w1 = [0;0;qdot];
w1dot = [0;0;qddot];
w2 = qdot*(z1+z2);
w2dot = qddot*(z1+z2)+qdot*qdot*[-cq;-sq;0];
w3 = qdot*(z1+2*z2);
w3dot = qddot*(z1+2*z2)+qdot*qdot*[-2*cq;-2*sq;0];

v2 = cross(w1,s1);
v3 = v2+cross(w2,s2);

a2 = cross(w1dot,s1)+cross(w1,cross(w1,s1));
a3 = a2+cross(w2dot,s2)+cross(w2,cross(w2,s2));

ac2 = a2+cross(w2dot,r2)+cross(w2,cross(w2,r2));
ac3 = a3+cross(w3dot,r3)+cross(w3,cross(w3,r3));

m2 = 17.4;
m3 = 4.8;
I1 = 0.35;
I2 = R02*diag([0.13,0.542,0.539])*R02';
I3 = R03*diag([0.066,0.0125,0.086])*R03';
g = 9.8;
```

```
f3 = m3*ac3+[0;0;m3*g];
n3 = I3*w3dot+cross(w3,I3*w3)+cross(r3,f3);
t3 = z3'*n3;

f2 = m2*ac2+f3+[0;0;m2*g];
n2 = I2*w2dot+cross(w2,I2*w2)+cross(r2,f2)+n3+cross((s2-r2),f3);
t2 = z2'*n2;

n1 = I1*w1+n2+cross(s1,f2);
t1 = z1'*n1;

tau1(i+1) = t1;
tau2(i+1) = t2;
tau3(i+1) = t3;

end

t = 0:0.001:20;
plot(t,tau1,'-',t,tau2,'--',t,tau3,'-.');
xlabel('t/s'),ylabel('\tau /Nm');
legend('\tau 1','\tau 2','\tau 3');
```

2. PD control simulation code

```
function main_puma_pd
[t,x] = ode45(@puma_pd,0:0.001:20,[0 0 0 0 0 0]);

plot(t,x(:,1),'-',t,x(:,2),'--',t,x(:,3),'-.','linewidth',2);
legend('q1','q2','q3');
hold on
plot(t,pi*ones(1,length(t))/2,'--');
xlabel('t/s'),ylabel('q/rad');
hold off
end

function xdot = puma_pd(~,q)
xdot = zeros(6,1);
qd = [pi/2; pi/2; pi/2];

A = [50 0 0; 0 180 0; 0 0 50];
B = [35 0 0; 0 50 0; 0 0 20];
tau = -A*([q(1);q(2);q(3)]-[qd(1);qd(2);qd(3)])-B*[q(4);q(5);q(6)];

xdot(1) = q(4);
xdot(2) = q(5);
xdot(3) = q(6);

H = [22+0.9*cos(q(2))*cos(q(2)), 1.17+1.92*sin(q(2)), -0.3*cos(q(2)+q(3));
    1.17+1.92*sin(q(2)), 1.66, -0.29;
    -0.3*cos(q(2)+q(3)), -0.29, 0.11];
xdot(4:6) = H\(-...
    [-
1.8*cos(q(2))*sin(q(2))*q(4)*q(5)+1.92*cos(q(2))*q(5)*q(5)+0.3*sin(q(2)+q(3))*(q(5)+q(6))*
q(6);
    91.9*sin(q(2))-3.3*sin(q(2)+q(3))+0.9*sin(q(2))*cos(q(2))*q(4)*q(4)-
0.3*sin(q(2)+q(3))*q(4)*q(6);
    0.3*sin(q(2)+q(3))*q(4)*q(5)-3.3*sin(q(2)+q(3))]+tau);
```

```
end
```

3. PID control simulation code

```
function main_puma_pid
[t,x] = ode45(@puma_pid,0:0.001:20,[0 0 0 0 0 0 0 0 0]);

plot(t,x(:,1),'-',t,x(:,2),'--',t,x(:,3),'-.','linewidth',2);
legend('q1','q2','q3');
hold on
plot(t,pi*ones(1,length(t))/2,'--');
xlabel('t/s'),ylabel('q/rad');
hold off
end

function xdot = puma_pid(~,q)
xdot = zeros(9,1);
qd = [pi/2; pi/2; pi/2];

A = [1200 0 0; 0 180 0; 0 0 50];
B = [300 0 0; 0 50 0; 0 0 10];
C = [300 0 0; 0 120 0; 0 0 10];
tau = -A*([q(1);q(2);q(3)]-[qd(1);qd(2);qd(3)])-B*[q(4);q(5);q(6)]-C*[q(7);q(8);q(9)];

xdot(1) = q(4);
xdot(2) = q(5);
xdot(3) = q(6);

H = [22+0.9*cos(q(2))*cos(q(2)), 1.17+1.92*sin(q(2)), -0.3*cos(q(2)+q(3));
    1.17+1.92*sin(q(2)), 1.66, -0.29;
    -0.3*cos(q(2)+q(3)), -0.29, 0.11];
xdot(4:6) = H\(-...
    [-
1.8*cos(q(2))*sin(q(2))*q(4)*q(5)+1.92*cos(q(2))*q(5)*q(5)+0.3*sin(q(2)+q(3))*(q(5)+q(6))*
q(6);
    91.9*sin(q(2))-3.3*sin(q(2)+q(3))+0.9*sin(q(2))*cos(q(2))*q(4)*q(4)-
0.3*sin(q(2)+q(3))*q(4)*q(6);
    0.3*sin(q(2)+q(3))*q(4)*q(5)-3.3*sin(q(2)+q(3))]+tau);
xdot(7:9) = q(1:3)-qd;
end
```

4. PD plus gravity compensation control simulation code

```
function main_puma_pdg
[t,x] = ode45(@puma_pdg,0:0.001:20,[0 0 0 0 0 0]);

plot(t,x(:,1),'-',t,x(:,2),'--',t,x(:,3),'-.','linewidth',2);
legend('q1','q2','q3');
hold on
plot(t,pi*ones(1,length(t))/2,'--');
xlabel('t/s'),ylabel('q/rad');
hold off
end

function xdot = puma_pdg(~,q)
xdot = zeros(6,1);
qd = [pi/2; pi/2; pi/2];

A = [500 0 0; 0 180 0; 0 0 50];
```

```
B = [350 0 0; 0 50 0; 0 0 20];
tau = -A*([q(1);q(2);q(3)]-[qd(1);qd(2);qd(3)])-B*[q(4);q(5);q(6)];

xdot(1) = q(4);
xdot(2) = q(5);
xdot(3) = q(6);

H = [22+0.9*cos(q(2))*cos(q(2)), 1.17+1.92*sin(q(2)), -0.3*cos(q(2)+q(3));
    1.17+1.92*sin(q(2)), 1.66, -0.29;
    -0.3*cos(q(2)+q(3)), -0.29, 0.11];
xdot(4:6) = H\(-...
    [-
1.8*cos(q(2))*sin(q(2))*q(4)*q(5)+1.92*cos(q(2))*q(5)*q(5)+0.3*sin(q(2)+q(3))*(q(5)+q(6))*
q(6);
    0.9*sin(q(2))*cos(q(2))*q(4)*q(4)-0.3*sin(q(2)+q(3))*q(4)*q(6);
    0.3*sin(q(2)+q(3))*q(4)*q(5)]+tau);

end
```

5. Computed torque control simulation code

```
function main_puma_ct
[t,x] = ode45(@puma_ct,0:0.001:20,[0 0 0 0 0 0]);

figure(1),
plot(t,x(:,1),'-',t,x(:,2),'--',t,x(:,3),'-.','linewidth',2.5);
legend('q1','q2','q3');
hold on
plot(t,pi/2+sin(t),'--');
xlabel('t/s'),ylabel('q/rad');
hold off
figure(2),
plot(t,x(:,4),'-',t,x(:,5),'--',t,x(:,6),'-.','linewidth',2.5);
legend('w1','w2','w3');
hold on
plot(t,cos(t),'--');
xlabel('t/s'),ylabel('w/rad/s');
hold off
end

function xdot = puma_ct(t,q)
xdot = zeros(6,1);
qd = [pi/2+sin(t); pi/2+sin(t); pi/2+sin(t)];
qddot = [cos(t); cos(t); cos(t)];
qd2dot = -[sin(t); sin(t); sin(t)];

A = [500 0 0; 0 180 0; 0 0 50];
B = [350 0 0; 0 50 0; 0 0 20];

xdot(1) = q(4);
xdot(2) = q(5);
xdot(3) = q(6);

xdot(4:6) = qd2dot - A*(q(4:6)-qddot) -B*(q(1:3)-qd);

end
```