Report of Coursework 2 Part 2

Group 5

Content

Task 1 3

Newton Approach 3

Lagrange Approach 7

Task2 10

Task3 12

For Step Input: 13

For Sine Wave Input: 23

Task4 31

Model Evaluation 34

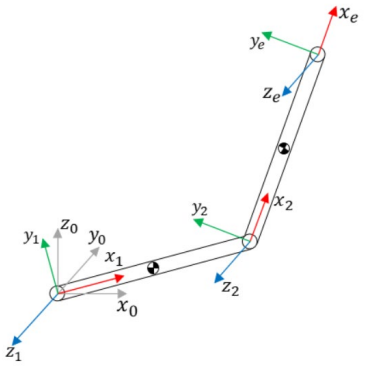
Task 1

Develop the equations of motion for the two-link robot.

Newton Approach

Developing the equations of motion for the two-link robot using Newton approach

1.Building coordinate system



We use the improved DH method to establish the linkage coordinate system of the two-link robot. The following is the symbol description:

$x_0y_0z_0$: Base coordinate system

$x_1y_1z_1$: link 1 coordinate system

$x_2y_2z_2$: link 2 coordinate system

$x_ey_ez_e$: end effector frame

l_1: the length of link 1

l_2: the length of link 2

2. Improvement of DH parameter table

frame	α	a	q	d
link 1	$\frac{\pi}{2}$	0	q_1	0
link 2	0	l_1	q_2	0
end effector	0	l_2	0	0

3. Forward and reverse recursion

3.1 Forward recursion of velocity

The forward recurrence of velocity is to calculate the center of mass velocity and angular velocity of each link successively starting from the base coordinate system of the robot.

3.1.1 Velocity recursion from base to link 1

For a two-link robot the base is fixed, so the linear velocity and angular velocity are zero.

$$\omega_0^0 = 0,\ \dot{P}_0^0 = 0,$$

Since joint 1 is a rotary joint, the transfer relation of angular velocity from the base to link 1 is

$$\omega_1^1 = R_0^1\omega_0^0 + \dot{q}_1 z_1^1$$

ω_1^1 is the angular velocity of link one in the coordinate system of link one. R_0^1 is the rotation matrix of the vector from the base coordinate system to the link 1 coordinate system (solved in terms of the DH parameter).$\dot{q1}$ is the angular velocity of joint one. (This is a scalar quantity). z_1^1 is the expression of z axis in link 1 coordinate system and it is a constant

$$z_1^1 = (001)^T$$

According to the velocity recurrence formula, the velocity transfer relation from the base to link 1 can be written:.

$$\dot{P}_1^1 = R_0^1\left(\dot{P}_0^0 + \omega_0^0 \times r_{0,1}^0\right)$$

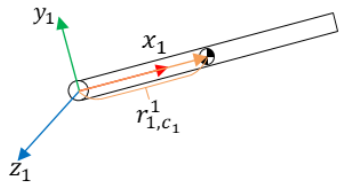
$\dot{p}_1^1$ is the linear velocity of the origin of link one in link one coordinate system. $r_{0,1}^0$ is the representation in the base coordinate system of the origin of the base coordinate system to the origin of the link 1 coordinate system in the base coordinate system. (In the two-link robot , the origin of the base coordinate system and the origin of the link 1 coordinate system coincide, so this is a zero vector.) and we're looking for the linear and angular velocities of the center of mass in terms of the center of mass coordinates. The angular velocity at any point on the rigid body is the same, so:

$$\omega_{c_1}^{c1} = \omega_1^1$$

The velocity relationship between any two points on the rigid body is:

$$\dot{P}_{c_1}^{c1} = \dot{P}_1^1 + \omega_1^1 \times r_{1,c1}^1$$

$\dot{p}_{c1}^{c1}$ is the linear velocity of the center of mass of link 1 in the center of mass coordinate system (because we specify that the center of mass coordinate system and the link coordinate system are in the same direction.) $r_{1,c1}^1$ is the vector from the origin of link one to the centroid of link one in link one coordinates. Show the graph 2:



3.1.2 The velocity recursion from link 1 to link 2

The process of velocity recursion from link 1 to link 2 is similar to the base to link 1. Therefore, the transfer relation of angular velocity from link 1 to link 2 is:

$$\omega_2^2 = R_1^2\omega_1^1 + \dot{q}_2 z_2^2$$

And the velocity transfer relation from link 1 to link 2 is:

$$\dot{P}_2^2 = R_1^2\left(\dot{P}_1^1 + \omega_1^1 \times r_{1,2}^1\right)$$

And the linear and angular velocities of the center of mass in terms of the center of mass coordinates is

The linear and angular velocities of the center of mass

$$\omega_{c2}^{c2} = \omega_2^2$$

$$\dot{P}_{c2}^{c2} = \dot{P}_2^2 + \omega_2^2 \times r_{2,c2}^2$$

3.2 The forward recursion of acceleration

3.2.1 Acceleration recursion from base to link 1

The angular acceleration of the base of the robot is 0 ($\dot{\omega}_0^0$ = 0) in order to simplify the expression of gravity, we replace gravity with an acceleration field, so the base of the robot is considered to move with an acceleration of -g ($\ddot{p}_0^0$ = -g).

The angular acceleration transfer formula from the base to link 1 is as follows:

$$\dot{\omega}_1^1 = R_0^1 \dot{\omega}_0^0 + \left(R_0^1 \omega_0^0\right) \times \dot{q}_1 z_1^1 + \ddot{q}_1 z_1^1$$

The linear acceleration transfer formula from the base to link 1 is as follows:

$$\ddot{P}_1^1 = R_0^1 \left(\ddot{P}_0^0 + \dot{\omega}_0^0 \times r_{0,1}^0 + \omega_0^0 \times \left(\omega_0^0 \times r_{0,1}^0\right)\right)$$

Similarly, what we need to solve is the expression of the angular and linear acceleration of the centroid of link 1 in the centroid coordinate system, because the angular velocity of any point on the rigid body is the same, so the angular acceleration is also the same:

$$\dot{\omega}_{c1}^{c1} = \dot{\omega}_1^1$$

The acceleration relation between the origin of the coordinate system of link 1 and the center of mass is:

$$\ddot{P}_{c1}^{c1} = \ddot{P}_1^1 + \dot{\omega}_1^1 \times r_{1,c1}^1 + \omega_1^1 \times \left(\omega_1^1 \times r_{1,c1}^1\right)$$

3.2.2 The acceleration recursion from link 1 to link 2

The recursion process here is also repeated as 3.2.1. The angular acceleration transfer formula from link 1 to link 2 is as follows:

$$\dot{\omega}_2^2 = R_1^2 \dot{\omega}_1^1 + \left(R_1^2 \omega_1^1\right) \times \dot{q}_2 z_2^2 + \ddot{q}_2 z_2^2$$

The linear acceleration transfer formula from link 1 to link 2 is as follows:

$$\ddot{P}_2^2 = R_1^2 \left(\ddot{P}_1^1 + \dot{\omega}_1^1 \times r_{1,2}^1 + \omega_1^1 \times \left(\omega_1^1 \times r_{1,2}^1\right)\right)$$

Similarly, we need to solve the expression of angular acceleration and linear acceleration of the centroid of link 2 in the centroid coordinate system, then the angular acceleration of the centroid of link 2 is:

$$\dot{\omega}_{c2}^{c2} = \dot{\omega}_2^2$$

The acceleration relation between the origin of the coordinate system of link 2 and the center of mass is:

$$\ddot{P}_{c2}^{c2} = \ddot{P}_2^2 + \dot{\omega}_2^2 \times r_{2,c2}^2 + \omega_2^2 \times \left(\omega_2^2 \times r_{2,c2}^2\right)$$

3.3 Back up recursion

3.3.1 The torque at joint 2

We believe that the two-link planar arm is not subjected to any external force, so The Newton equation of link 2 has the following form:

$$m_2 \ddot{P}_{c2}^{c2} = f_2^2$$

f_2^2 is the force exerted by joint two on link two in the centroid coordinate system of link two. And we can solve f_2^2 through above equation. The Euler equation of link 2 has the following form:

$$I_{c2}^{c2} \dot{\omega}_{c2}^{c2} + \omega_{c2}^{c2} \times I_{c2}^{c2} \omega_{c2}^{c2} = \mu_2^2 - r_{2,c2}^2 \times f_2^2$$

According to Euler equation, we can solve μ_2^2. And it represents the torque applied by joint two to link two in link two centroid coordinates (or link two coordinates).

3.3.2 The torque at joint 1

The forces and moments applied by joint 2 to link 2 are f_2^2 and μ_2^2. According to Newton's third law, the reaction force and reaction moment applied to link 1 by link 2 through joint 2 are $-f_2^2$ and $-\mu_2^2$. However, these two physical quantities are currently expressed in the coordinate system of link 2, so the calculation of Newton's equation and Euler's equation of link 1 requires the transformation of these two physical quantities to the coordinate system of centroid of link 1 through the rotation matrix. The Newton equation of link 1 has the following form:

$$m_1 \ddot{P}_{c1}^{c1} = f_1^1 - R_2^1 f_2^2$$

$R_2^1 f_2^2$ is the force exerted on link two by joint two in the centroid frame of link one. The Euler equation of link 1 has the following form:

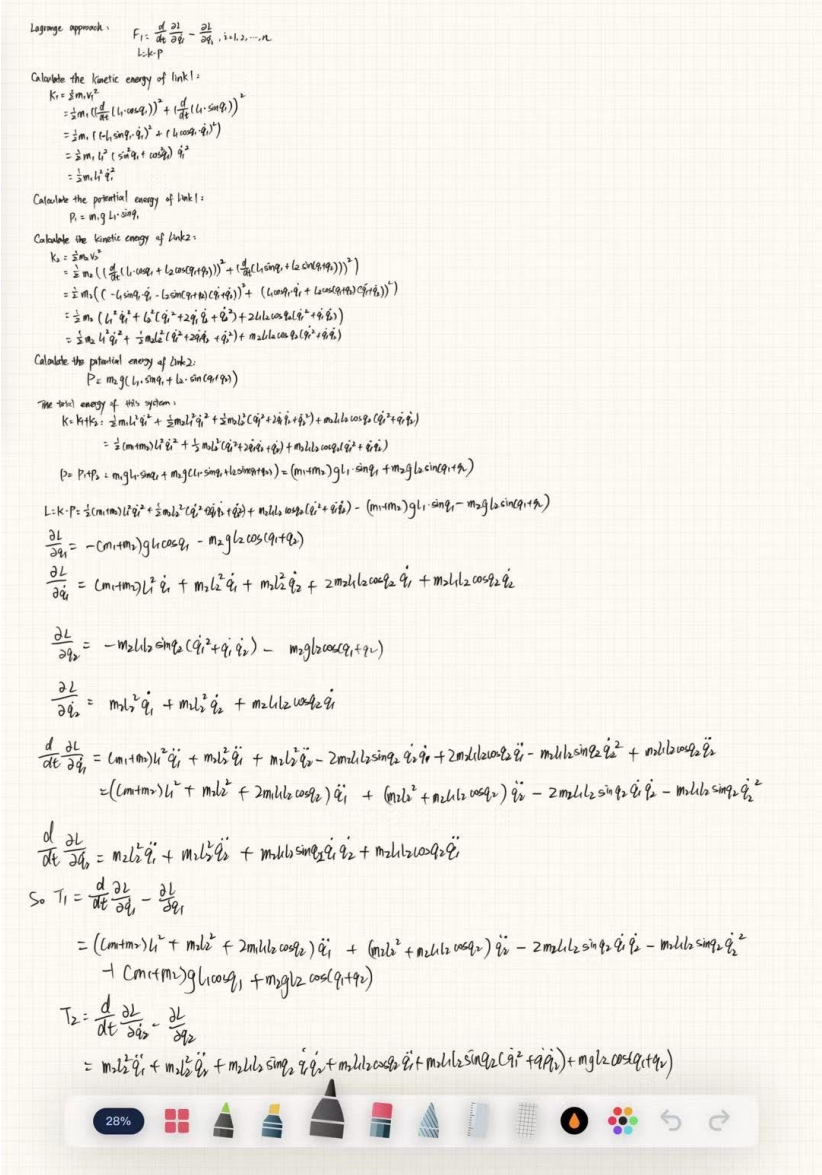
$$I_{c1}^{c1} \dot{\omega}_{c1}^{c1} + \omega_{c1}^{c1} \times I_{c1}^{c1} \omega_{c1}^{c1} = \mu_1^1 - \mu_2^1 + \left(-r_{1,c1}^1\right) \times f_1^1 + \left(-r_{2,c1}^1\right) \times \left(-f_2^1\right)$$

And μ_2^1.= $R_2^1 \mu_2^2$ is the torque applied by joint two to link two in the centroid coordinate system of link one. Therefore, the torque at joint two is:

$$\tau_1 = \left(\mu_1^1\right)^T z_1^1$$

Lagrange Approach

We use both handwriting and MATLAB to develop these equations.
Here is a sample for the **handwriting** part:



And we get the equations as above.
However, I forgot to calculate the rotational kinetic energy, but only calculate the translational energy.
In this case, I use MATLAB to develop the equations, which is also convenient for Task 2 implementation.

MATLAB Implementation:

```
clear;
%Initialize the parameter of the links
%mass, acceleration of gravity and the moment of inertia
syms m1 m2 g i1 i2
%The length of two links
syms l1 l2
%the angle of two links
syms t q1(t)
syms   q2(t)
%The center of mass of link 1 and 2
syms c1 c2

%The (x,y) position of Link 1
x1_c = c1*cos(q1(t));
y1_c = c1*sin(q1(t));
%The (x,y) position of Link 2
x2_c = l1*cos(q1(t))+c2*cos(q1(t)+q2(t));
y2_c = l1*sin(q1(t))+c2*sin(q1(t)+q2(t));

%The velocity and accelerated velocity of Link 1
%In the direction of X
x1_v = diff(x1_c,t);
x1_a = diff(x1_c,t, 2);
%In the direction of Y
y1_v = diff(y1_c,t);
y1_a = diff(y1_c,t, 2);

%The velocity and accelerated velocity of Link 2
%In the direction of X
x2_v = diff(x2_c,t);
x2_a = diff(x2_c,t, 2);
%In the direction of Y
y2_v = diff(y2_c,t);
y2_a = diff(y2_c,t, 2);

%Then calculate the system's kinetic energy
%The translational kinetic energy
KE_trans = 0.5*m1*(x1_v^2+y1_v^2) + 0.5*m2*(x2_v^2+y2_v^2);
%The rotation energy
KE_rot = 0.5*i1*((diff(q1,t))^2) + 0.5*i2*((diff(q1,t)+diff(q2,t))^2);
%The total energy
KE = KE_trans + KE_rot;

%Calculate the system's position energy (Only gravity)
PE = m1*g*y1_c + m2*g*y2_c;
```

You can open the file “Lagrange.m” to see the entire code.

```
%d/dt (∂L/∂q1_d) - ∂L/∂q1 = Q1
syms Q1
EoM_T1_LeftPart =  der_of_dLdq1d - dLdq1;
EoM_T1_LeftPart = simplify(EoM_T1_LeftPart);
EoM_T1_RightPart = Q1;
EoM_T1 = EoM_T1_LeftPart==EoM_T1_RightPart;
simplify(EoM_T1);

%d/dt (∂L/∂q2_d) - ∂L/∂q2 = Q2
syms Q2
EoM_T2_LeftPart =  der_of_dLdq2d - dLdq2;
EoM_T2_LeftPart = simplify(EoM_T2_LeftPart);
EoM_T2_RightPart = Q2;
EoM_T2 = EoM_T2_LeftPart==EoM_T2_RightPart;
simplify(EoM_T2);
```

Here, we get the equations of motion for the two-link robot.

Then, display them.

EoM_T1(t) =

i1*diff(q1(t), t, t) + i2*diff(q1(t), t, t) + i2*diff(q2(t), t, t) + c1^2*m1*diff(q1(t), t, t) + c2^2*m2*diff(q1(t), t, t) + c2^2*m2*diff(q2(t), t, t) + l1^2*m2*diff(q1(t), t, t) + c2*g*m2*cos(q1(t) + q2(t)) + c1*g*m1*cos(q1(t)) + g*l1*m2*cos(q1(t)) - c2*l1*m2*sin(q2(t))*diff(q2(t), t)^2 + 2*c2*l1*m2*cos(q2(t))*diff(q1(t), t, t) + c2*l1*m2*cos(q2(t))*diff(q2(t), t, t) - 2*c2*l1*m2*sin(q2(t))*diff(q1(t), t)*diff(q2(t), t) == Q1

(Q1=u-b1*diff(q1(t), t))

EoM_T2(t) =

i2*diff(q1(t), t, t) + i2*diff(q2(t), t, t) + c2^2*m2*diff(q1(t), t, t) + c2^2*m2*diff(q2(t), t, t) + c2*g*m2*cos(q1(t) + q2(t)) + c2*l1*m2*sin(q2(t))*diff(q1(t), t)^2 + c2*l1*m2*cos(q2(t))*diff(q1(t), t, t) == Q2

(Q2=u-b2*diff(q2(t), t))

Finally, Task 1 is finished.

Task2

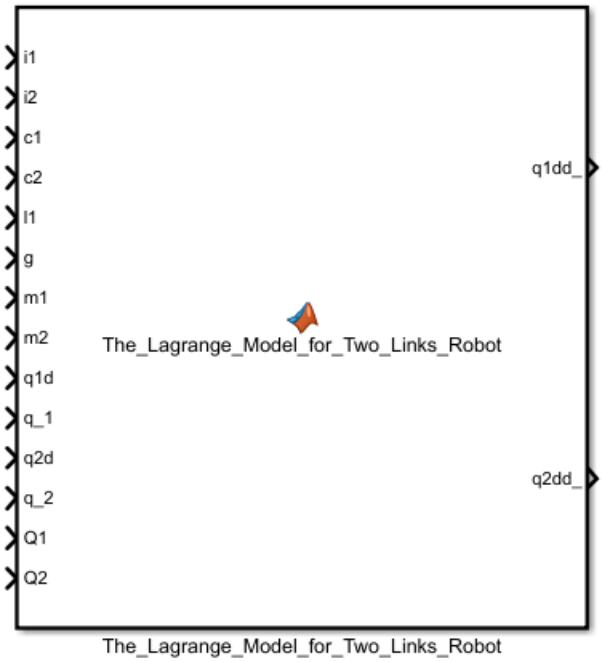
We use MATLAB to separate the equations and develop the Simulink sub system for our two-link robot.

Here is the coding screenshot:

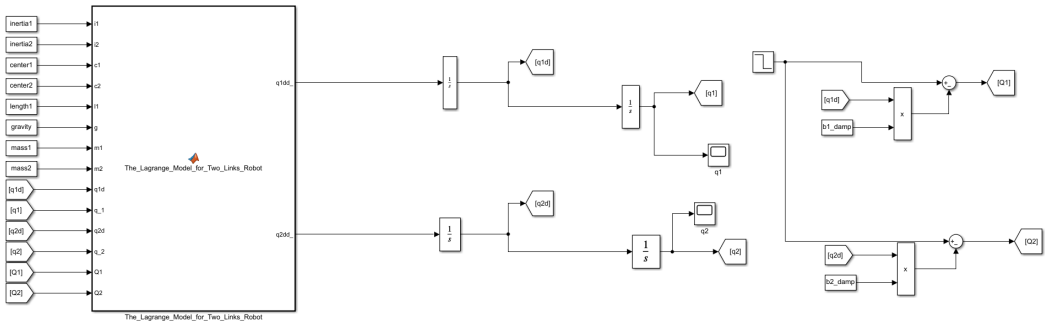
```
%Separate the equation
%Map the parameter
syms q_1 q_2 q1d q1dd q2d q2dd
old_list1 = [q1, diff(q1,1), diff(q1,2)];
new_list1 = [q_1, q1d, q1dd];
old_list2 = [q2, diff(q2,1), diff(q2,2)];
new_list2 = [q_2, q2d, q2dd];
EoM_T1 = subs(EoM_T1, [old_list1, old_list2], [new_list1, new_list2]);
EoM_T2 = subs(EoM_T2, [old_list1, old_list2], [new_list1, new_list2]);
S = solve([EoM_T1, EoM_T2], [q1dd, q2dd]);
Result1 = S.q1dd;
Result2 = S.q2dd;



tf_i_should_create_SL_block = true;
if(true==tf_i_should_create_SL_block)
    MODEL_NAME = 'Lagr_Model';
    if(4==exist(MODEL_NAME))
        close_system(MODEL_NAME, 0);
        delete(MODEL_NAME);
    end
    new_system(MODEL_NAME)
    open_system(MODEL_NAME)

    INPUT_VAR_ORDER = { 'i1', 'i2', ...
                        'c1', 'c2', 'l1', 'g', 'm1', 'm2', ...
                        'q1d', 'q_1', ...
                        'q2d', 'q_2', ...
                        'Q1', 'Q2'};

    % Put BOTH equations into one block
    matlabFunctionBlock( [MODEL_NAME,'/The_Lagrange_Model_for_Two_Links_Robot'], Result1, Result2, ...
                         'Optimize', false, ...
                         'Vars',     INPUT_VAR_ORDER, ...
                         'Outputs', {'q1dd_', 'q2dd_'}   );
end
```

You can see the entire code in “Lagrange.m”

Run this code, and it will develop the sub system automatically.



And then, we link this sub system to all the parameters to develop a simulation model. (You can run the model in “Lagr_Model_.slx”)



Then, we initialize the constant variables and run this model.

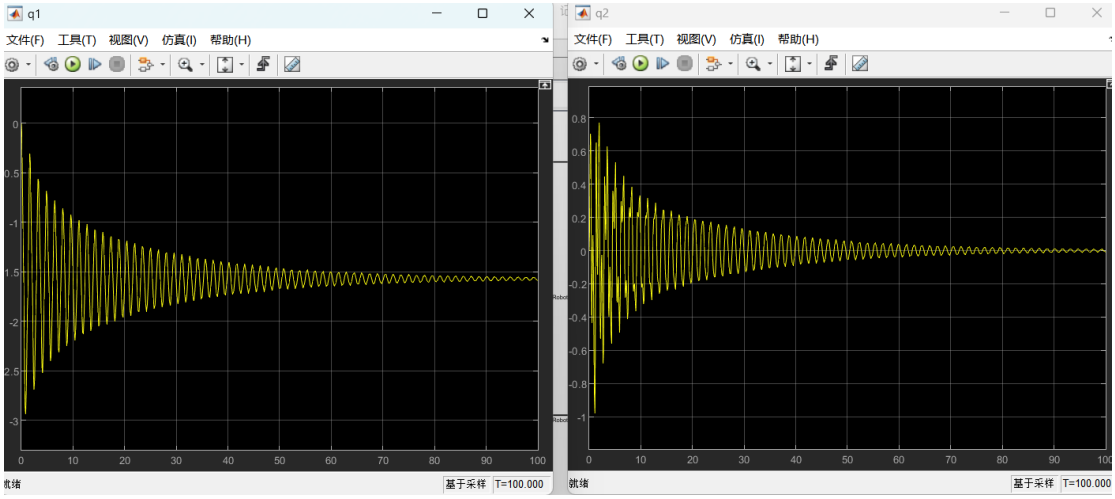
```
clear;
%without payload
inertia1 = 0.247;
inertia2 = 0.177;

center1 = 0.216;
center2 = 0.216;

length1 = 0.432;

gravity = 9.8;

mass1 = 15.91;
mass2 = 11.36;

b1_damp = 0.5; %0.1 (N.m/(rad/sec));
b2_damp = 0.5; %0.1 (N.m/(rad/sec));
```

You can see the code in “variable_for_simulink.m”.

We suppose the two link friction(damp) are both 0.5, and the stop time is 100. Use step

as input.

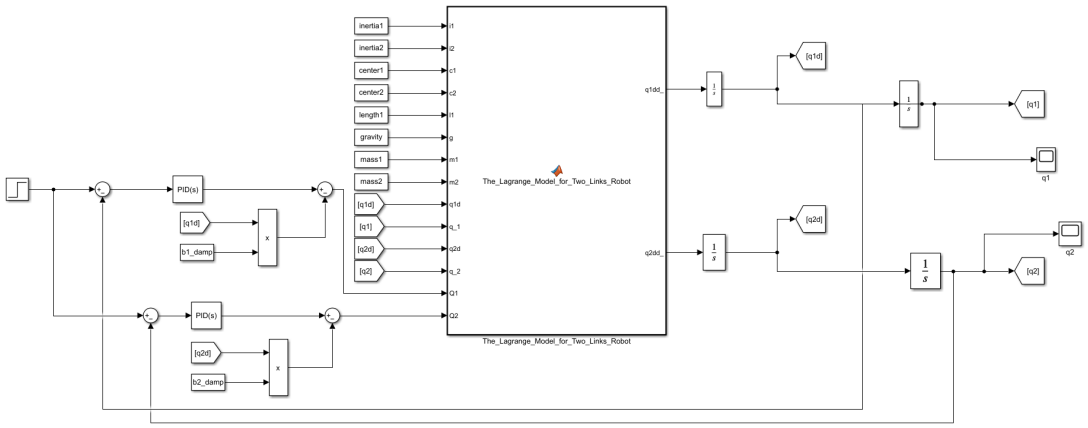
Then we got the variation tendency of the angles of two joints.



Then we have finished Task2 so far.

Task3

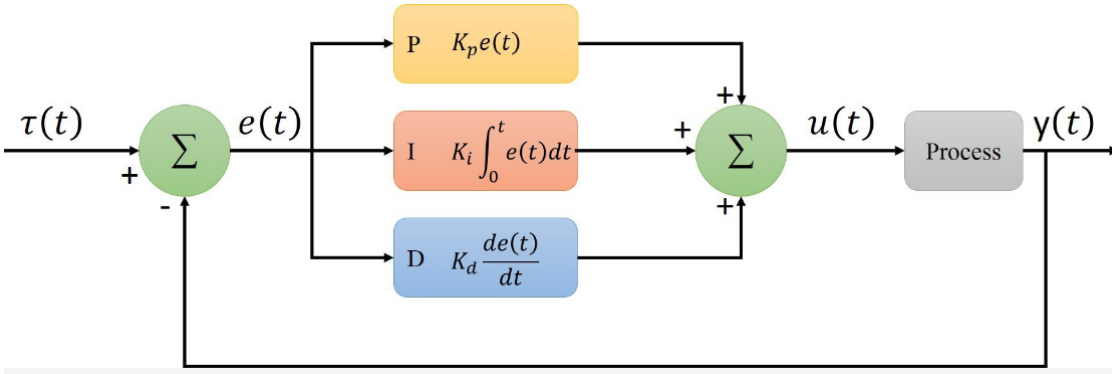
We design a PID control to implement the computed torque control law to stabilize the model.



For our two-link model, we need to control the two joints, that means, we need two PID controllers.

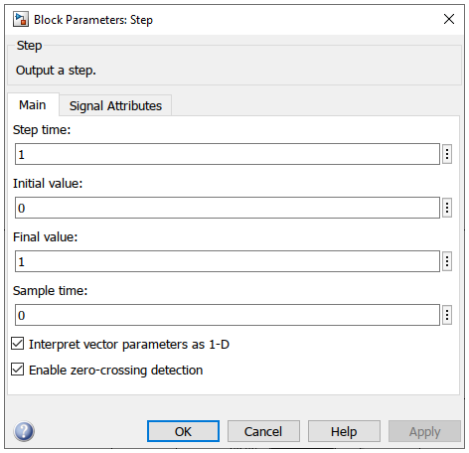
Hence, we use two PID modules connect two inputs and outputs (there are two Lagrange equations for this system).

The principle is as follows:



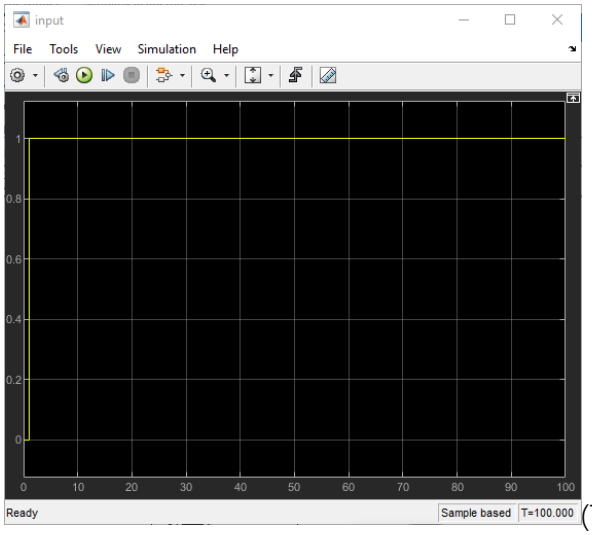
For Step Input:

(You can see the model in the file: PID_control_Model_step_input.slx)



We need the output be close to the ideal graph.

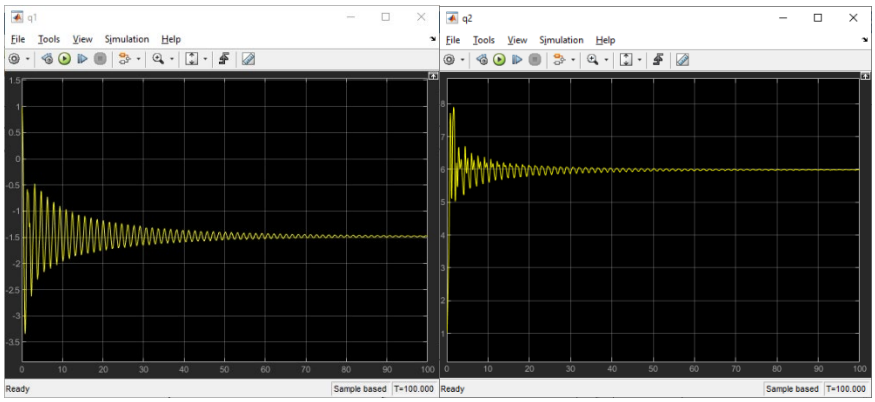
The graph of ideal graph is shown below:



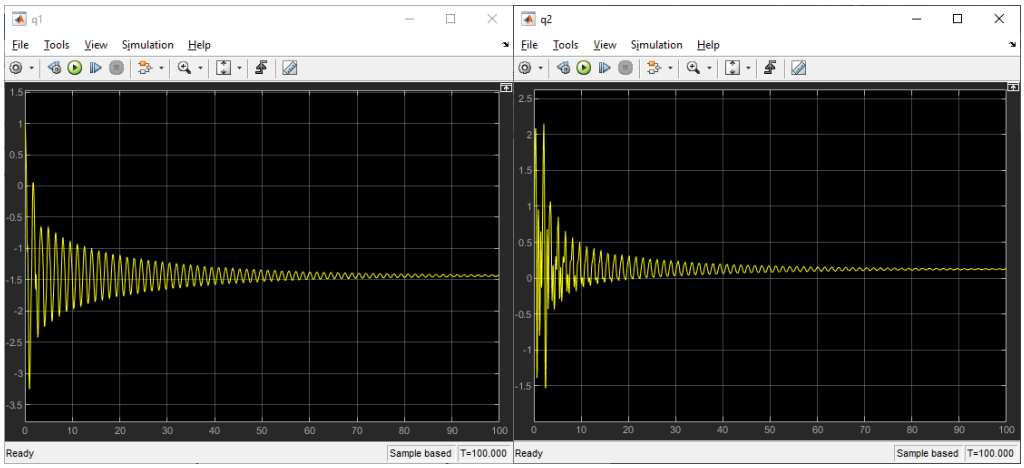
(This is the expectation value (the input))

Step1: set Kp be 1 and Ki and Kd be 0.

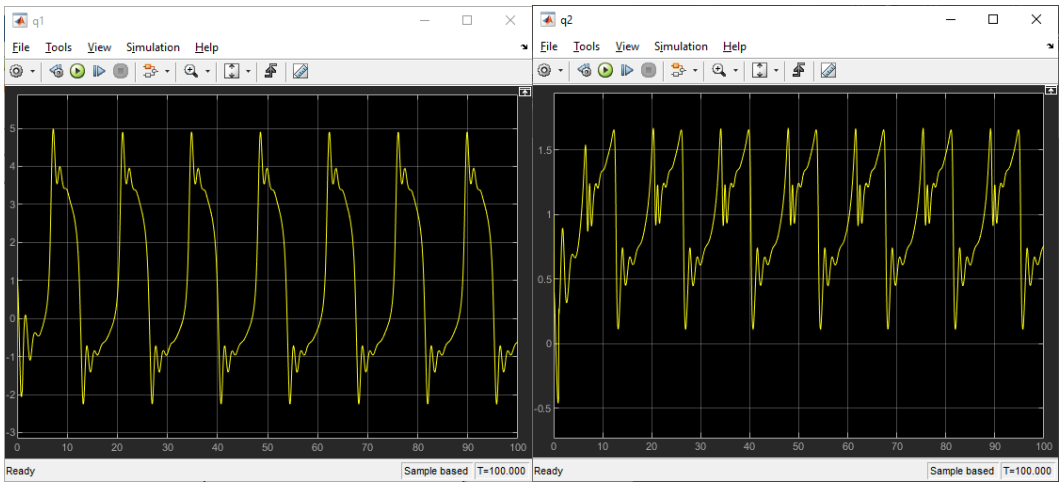
The initial wave is like this



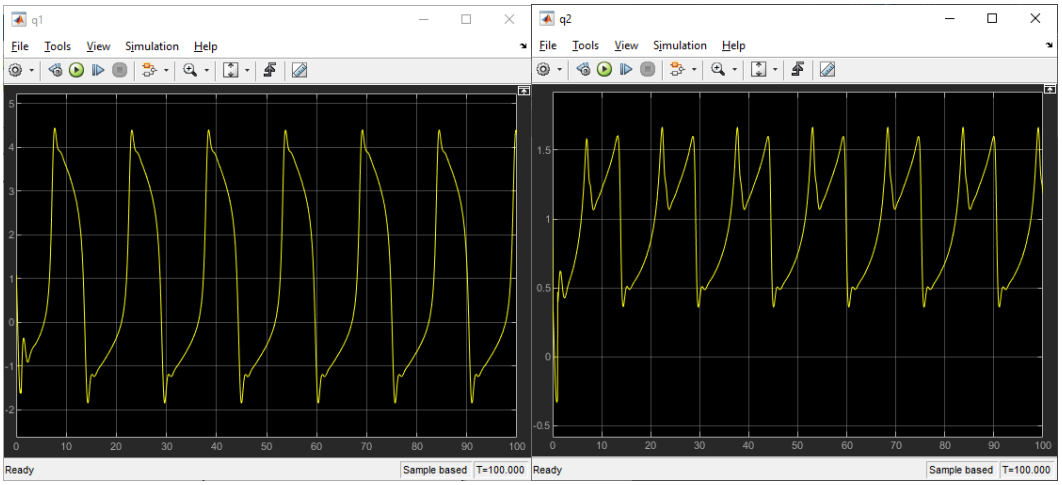
Step2: increase Kp, the wave become disorderd.



Step3: Increase Ki and Kd as well.
This is the wave when Kp=30,Ki=10,kd=10

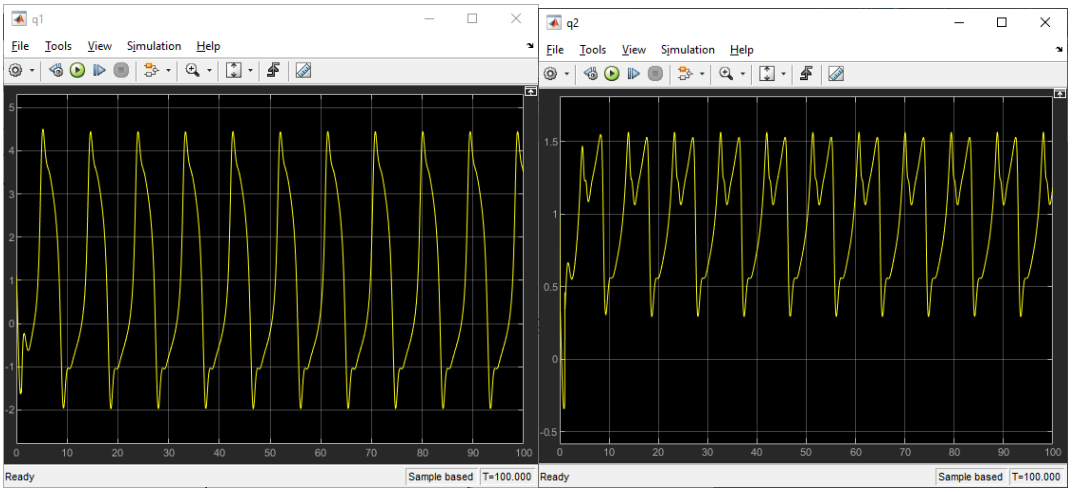


Step4: continue to increase Kd, found that if Kd become larger, the range of the wave will increase.
This is the wave when Kp=30,Ki=10,kd=20
Compare to step3 where Kd=10. This wave has a smaller height from -2 to 4.5.



Step5: we try to increase Ki and found that the range of the wave will increase. The difference is not bigger than increase Kd.

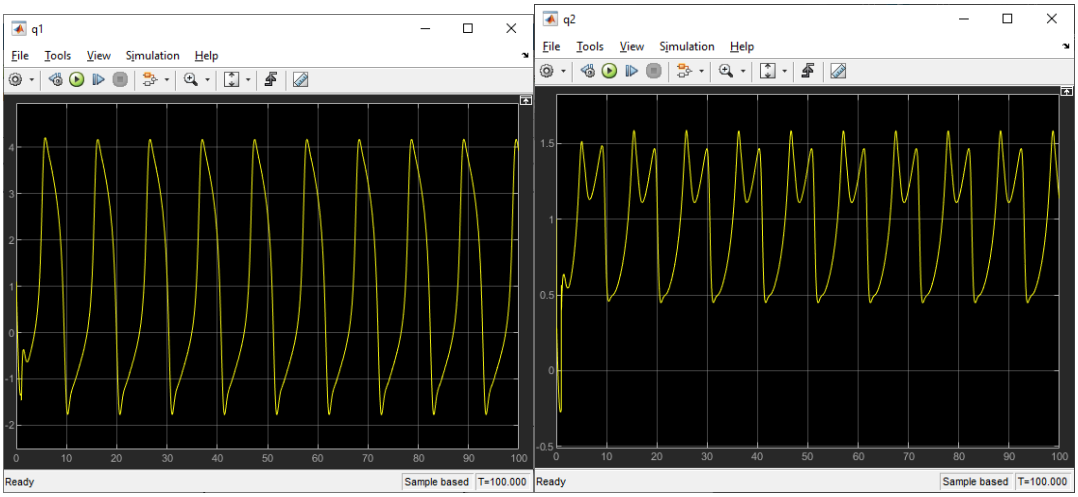
See the wave when Kp=30,Ki=20,kd=20



So we try to increase Kd more.

See the wave when Kp=30,Ki=20,kd=30

The intersection part of the wave looks more separately.

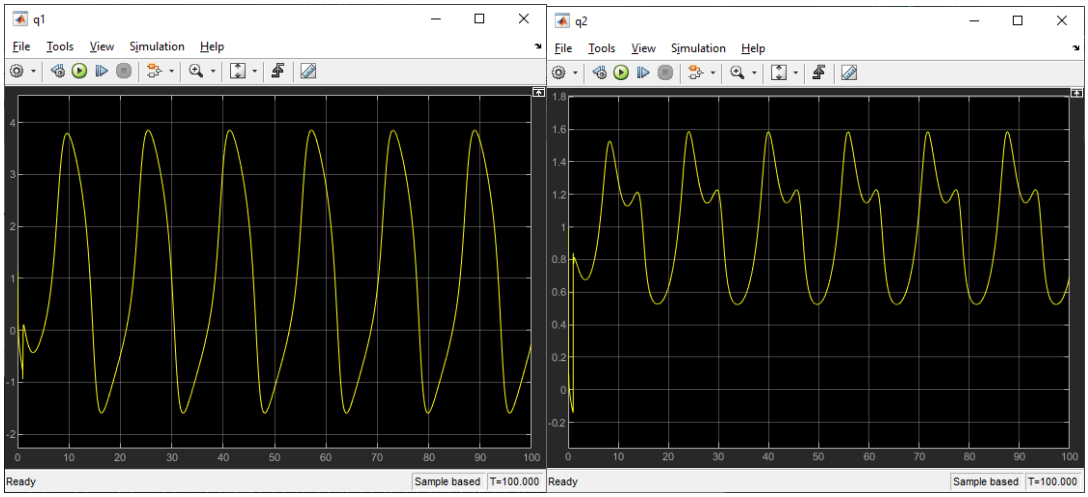


Step6: When we increase Kd more, up to more than 100

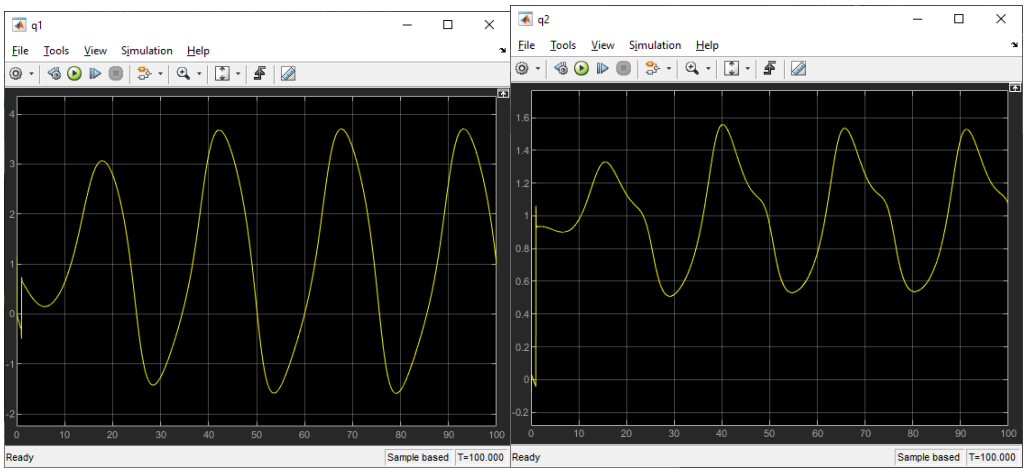
We found that the wave of q2 keep in around -1 to 1, but the wave of q1 is still

disordered

See the wave when Kp=30,Ki=20,kd=100

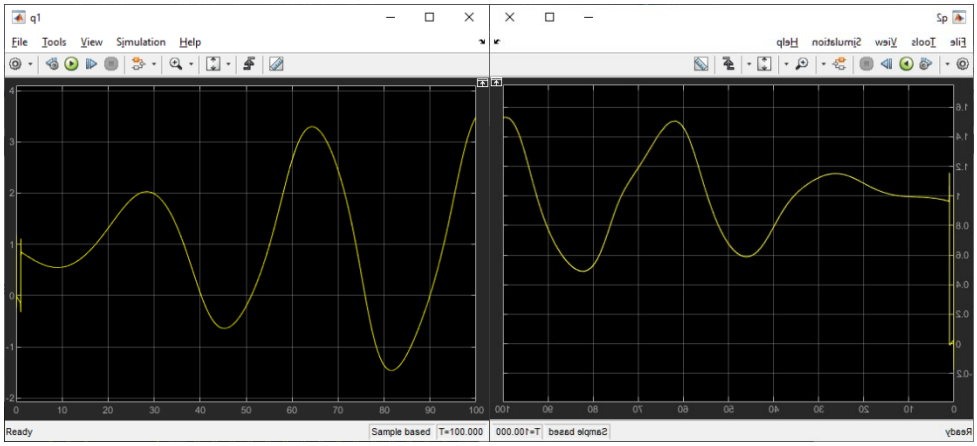


See the wave when Kp=30,Ki=20,kd=300

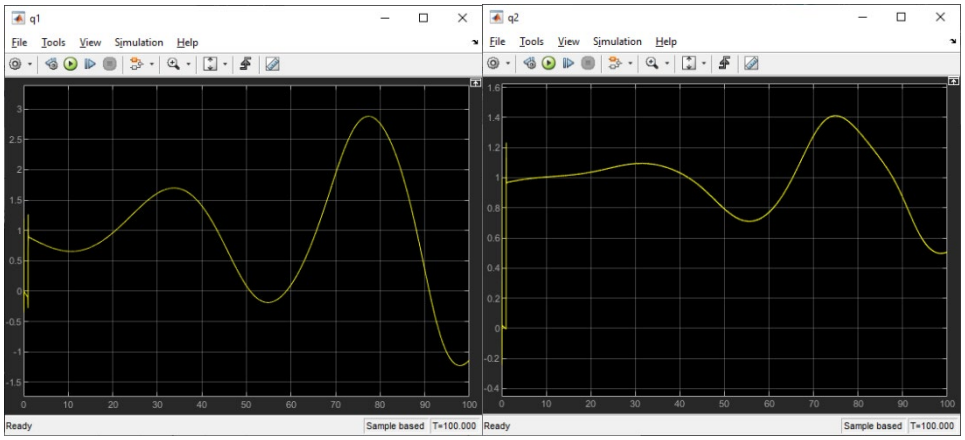


Step7: keep increasing the Kd, when Kd up to 1000, the wave of q2 show a trend of sin wave

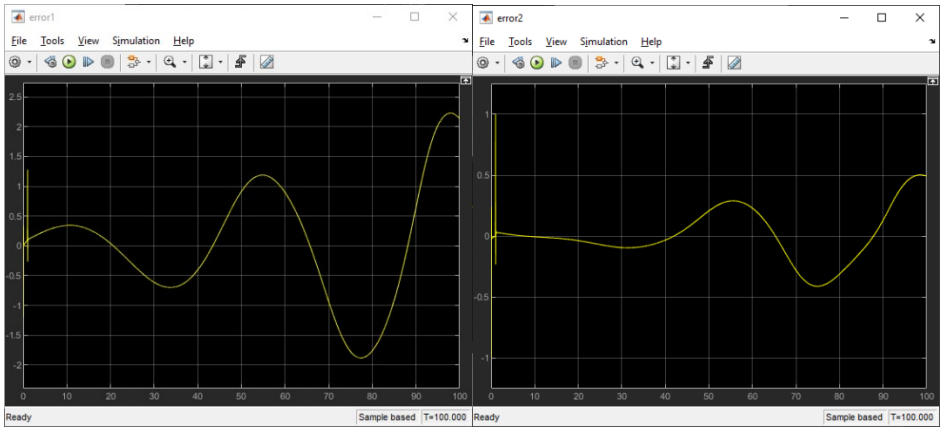
See the wave when Kp=30,Ki=20,kd=700



See the wave when Kp=30,Ki=20,kd=1000

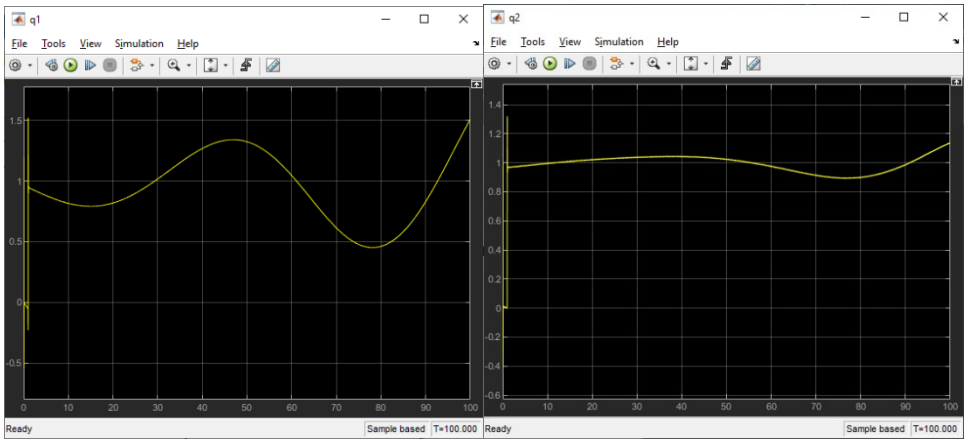


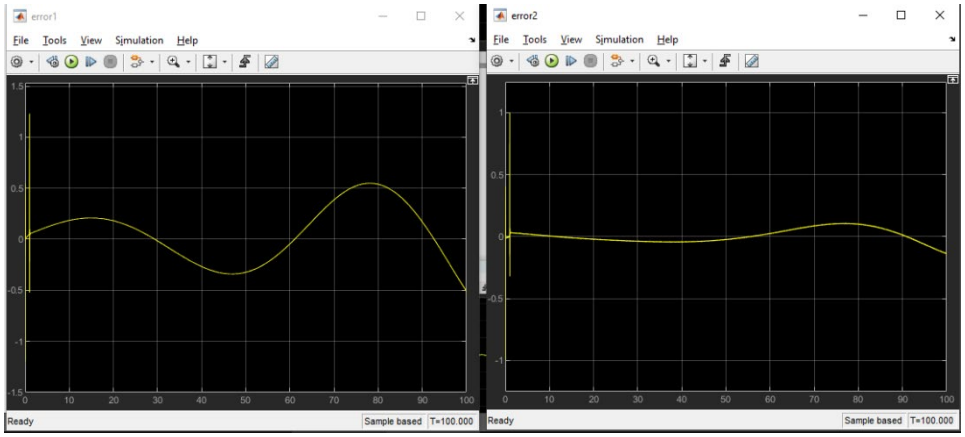
And the error is:



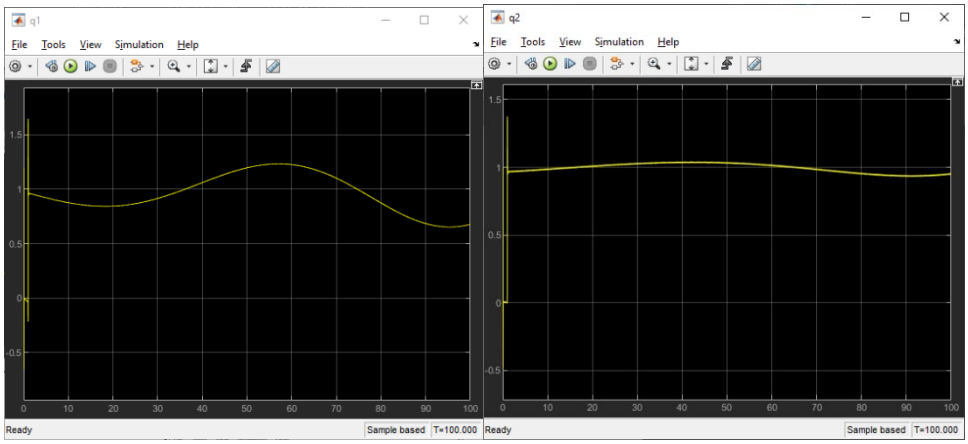
Step8: keep increasing the Kd, the wave of q1 and q2 are already stable and similar. But the error wave being closer to 0, when the Kd from 1000 to 3000

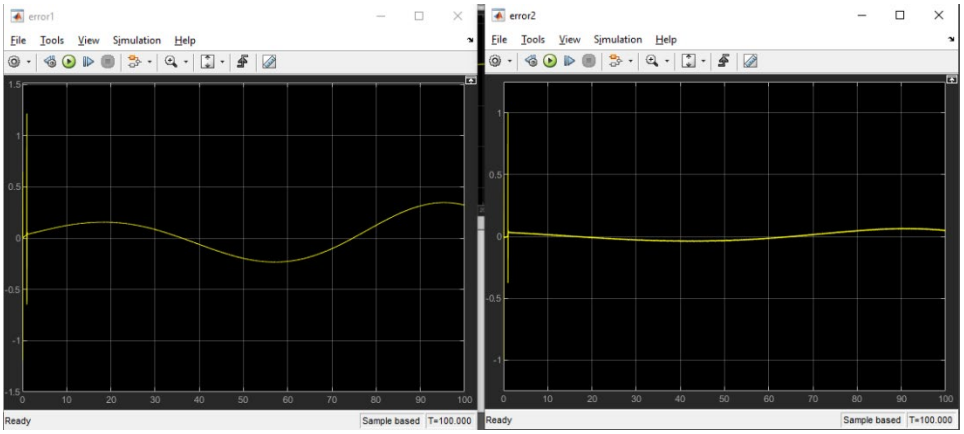
See the wave when Kp=30, Ki=20, Kd=2000



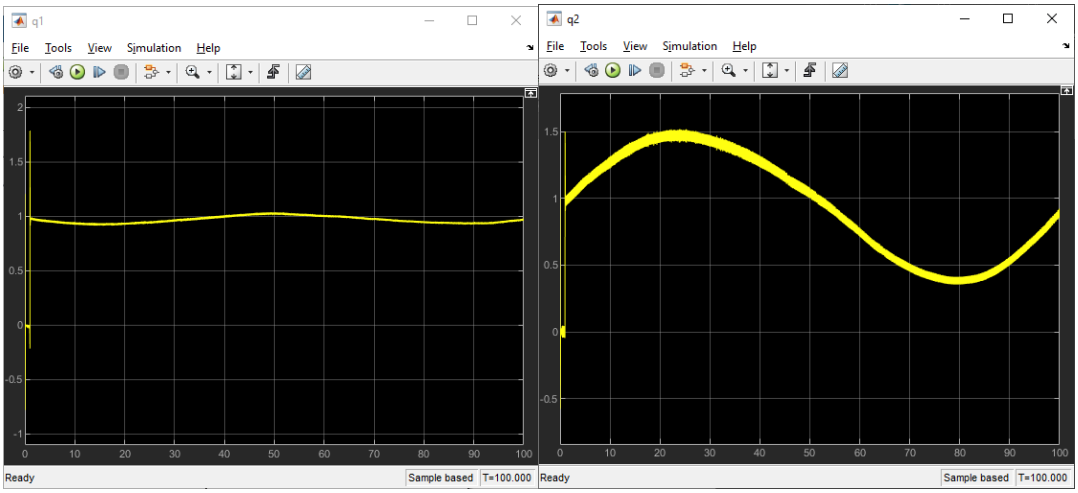


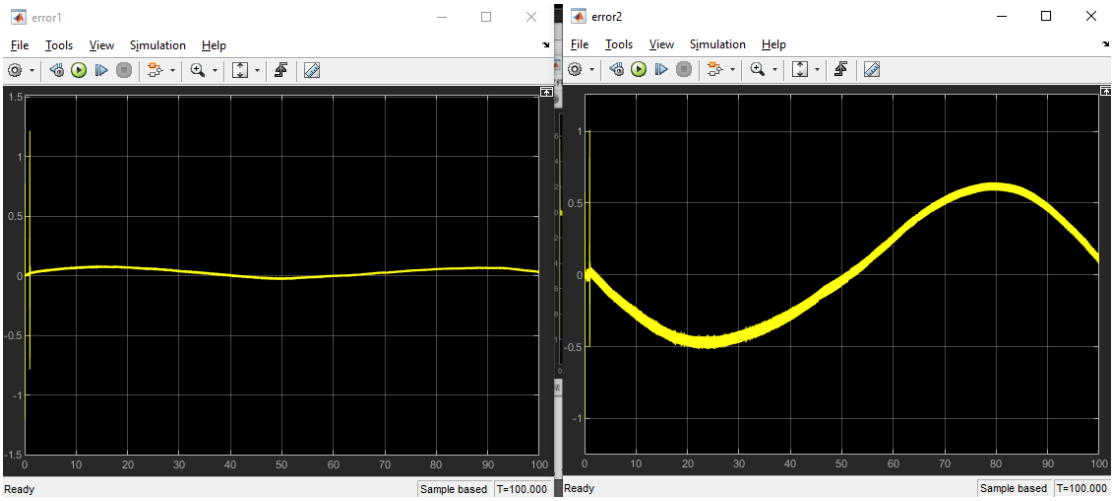
See the wave when Kp=30, Ki=20, Kd=3000





But if the Kd too larger, it will be wrong
See when Kd = 5000

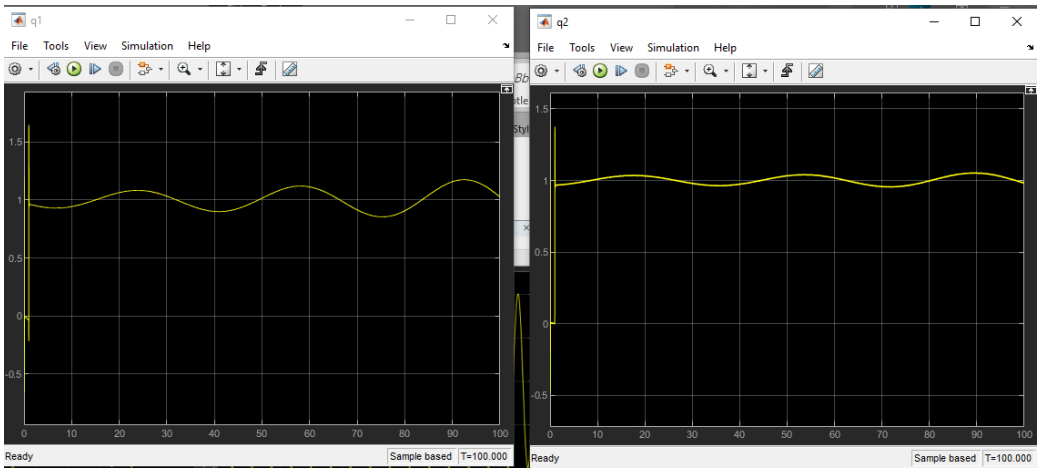


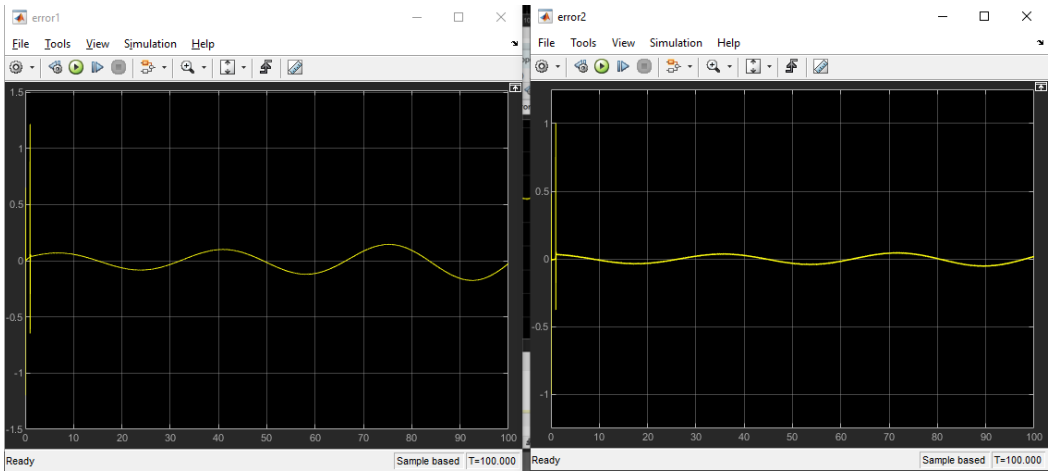


Step9:

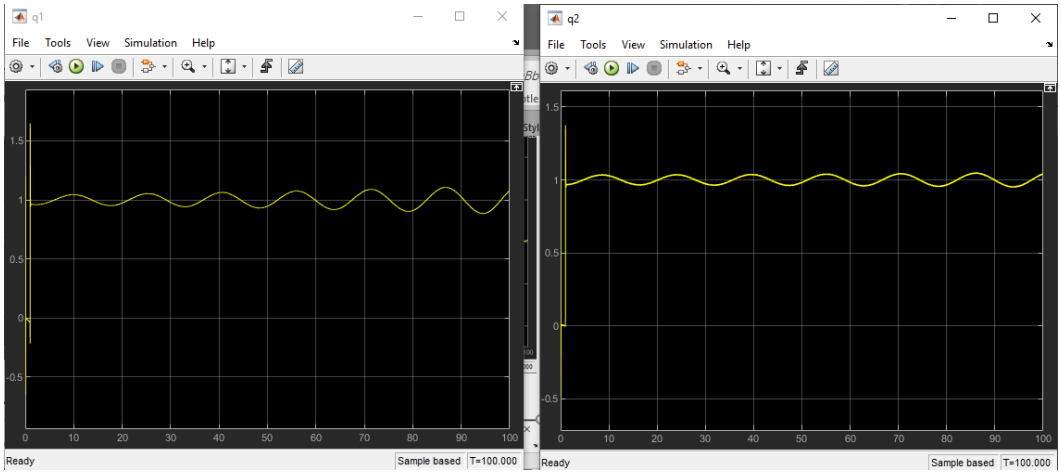
So we keep Kd=3000, and continue to change the Ki

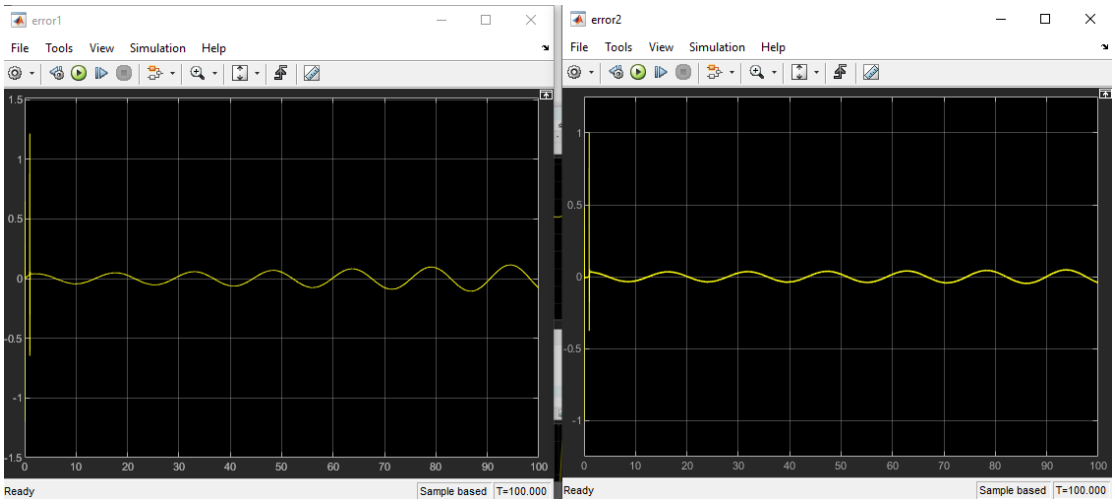
See the wave when Kp=30,Ki=100,Kd=3000



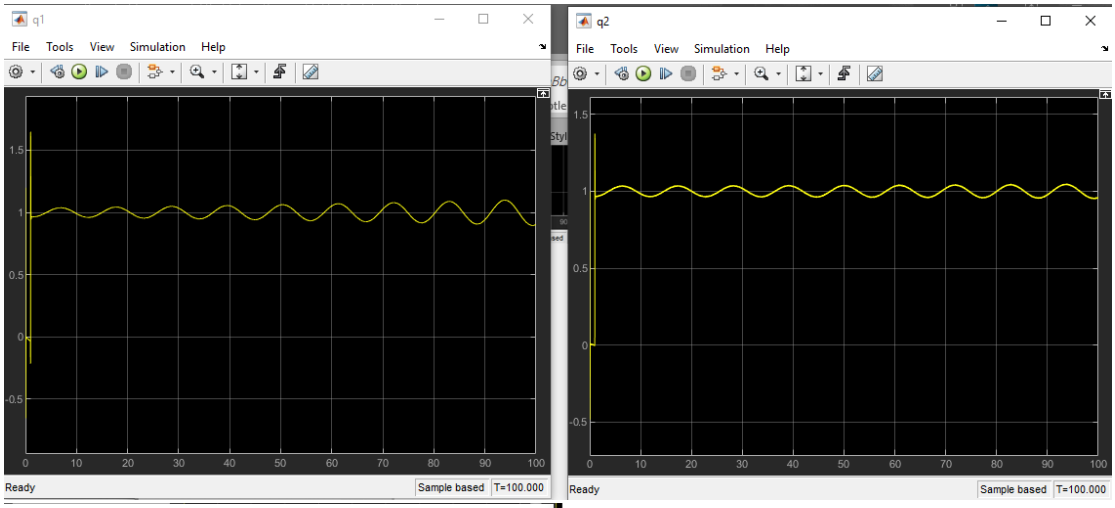


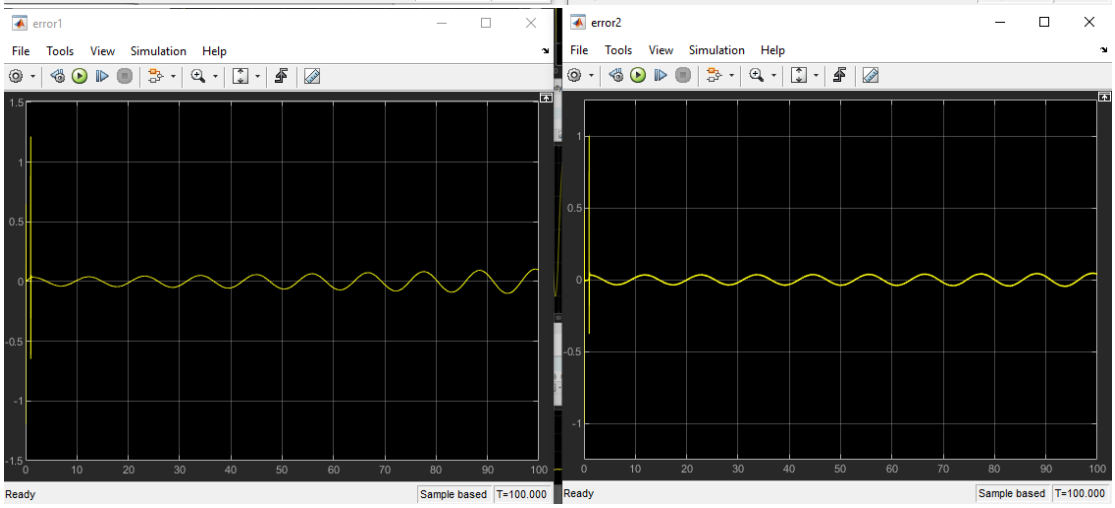
See the wave when Kp=30,Ki=500,kd=3000





See the wave when Kp=30,Ki=1000,kd=3000



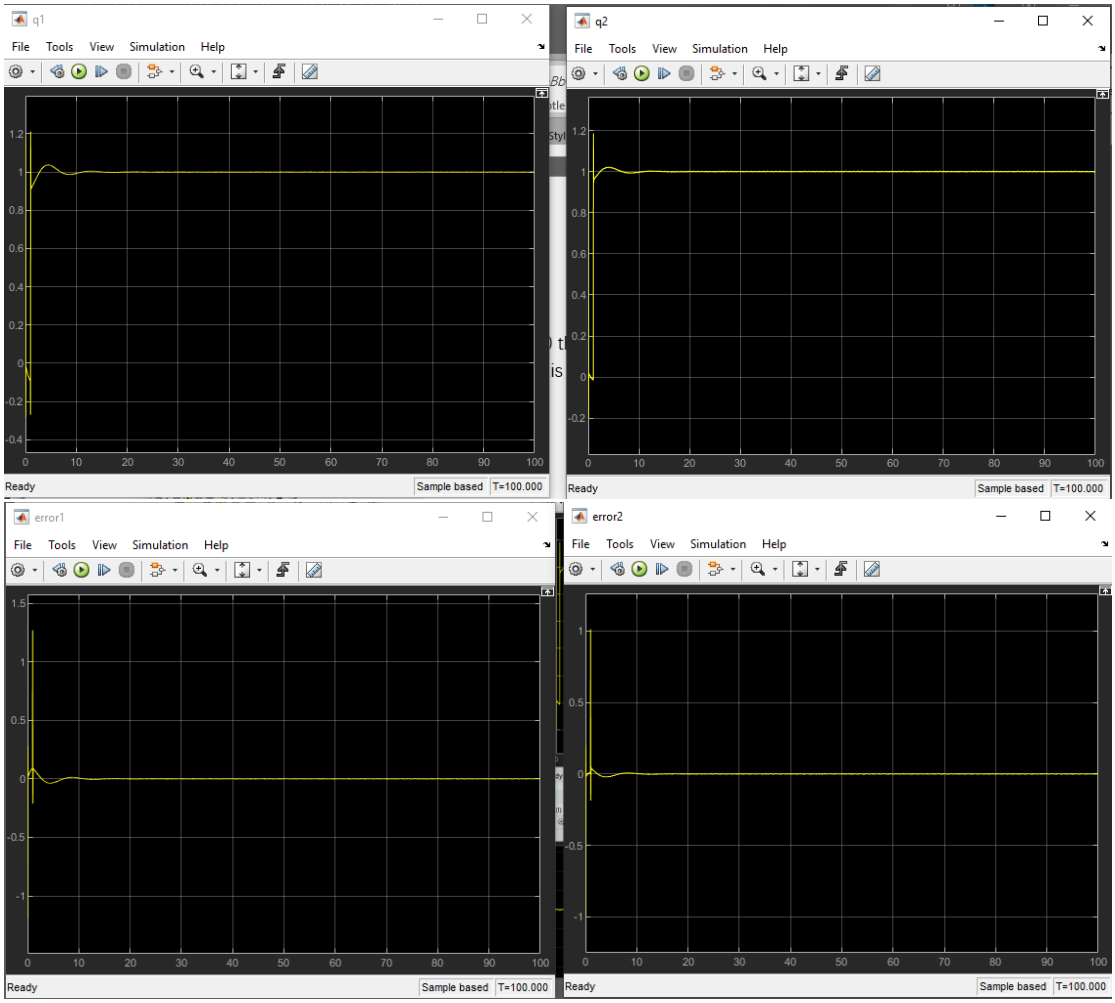


We can see that when Kp and Kd not changed, if Ki become larger up to 1000, the shock in q1 and q2 will be more severe.

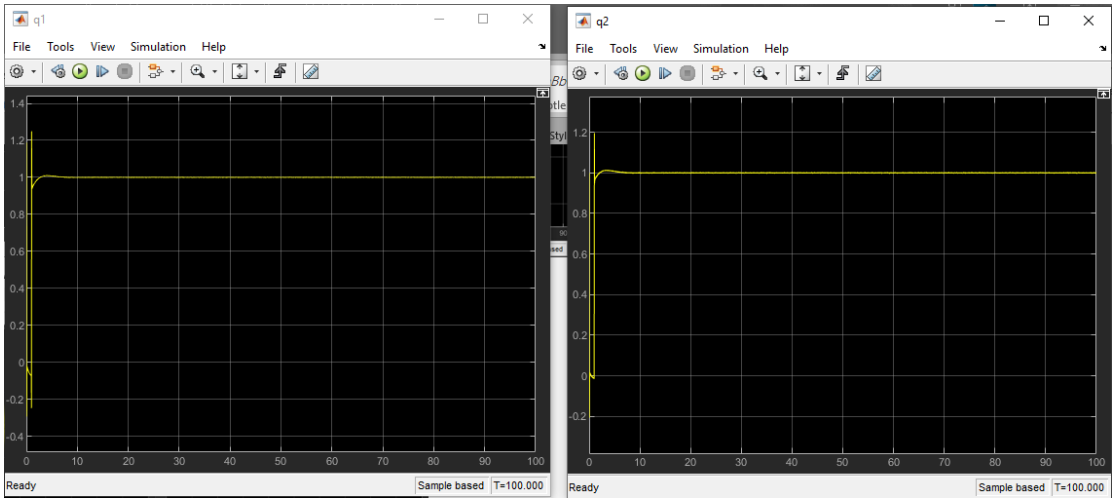
Step10: so we keep Ki=500 and increase kp, we found that when kp more than 500 the error is tend to be a line which is close to 0. And when kp is up to 1000, the error is better.

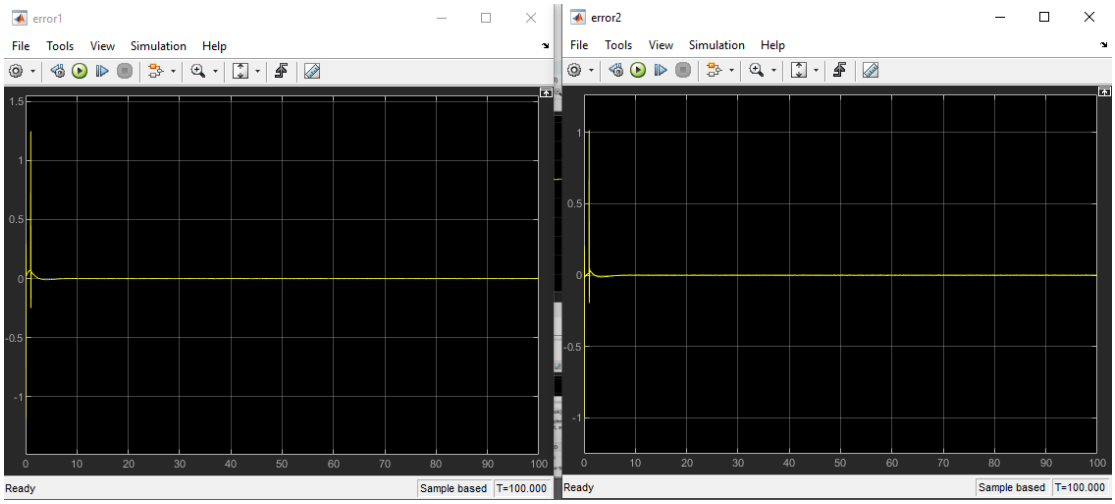
Thus, we stop here and our PID is Kp=1000, Ki=500, Kd=800

See the wave when Kp=500, Ki=500, Kd=800



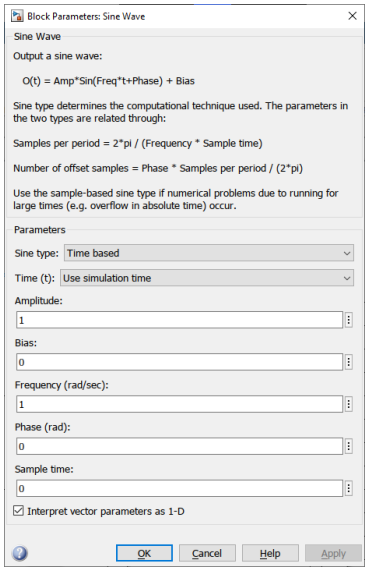
See the wave when Kp=1000, Ki=500, Kd=800





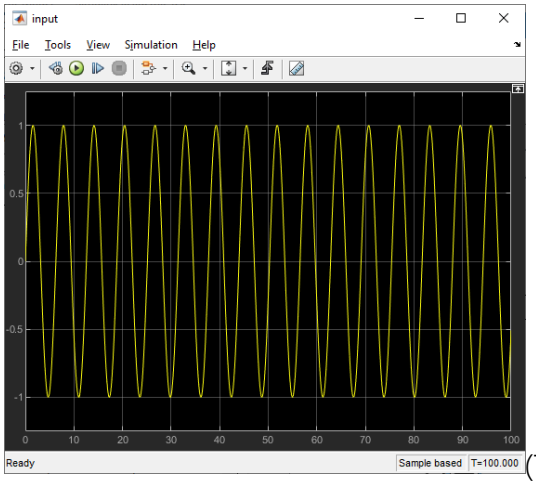
For Sine Wave Input:

(You can see the model in the file: PID_control_Model_sine_wave.slx)



We need the output be close to the ideal graph.

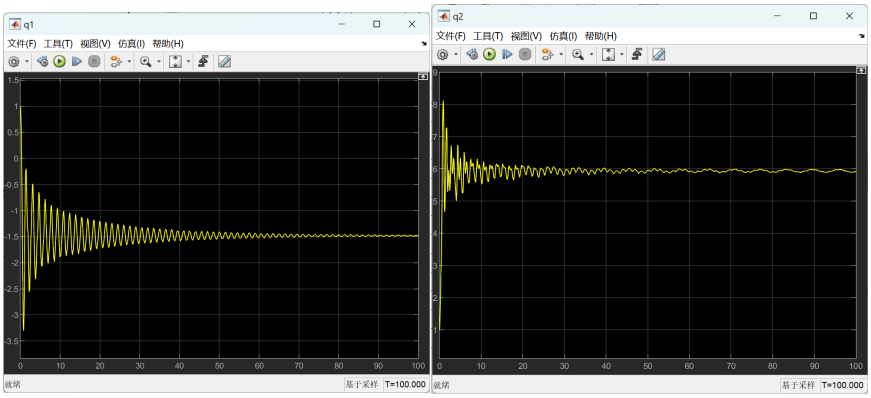
The graph of ideal graph is shown below:



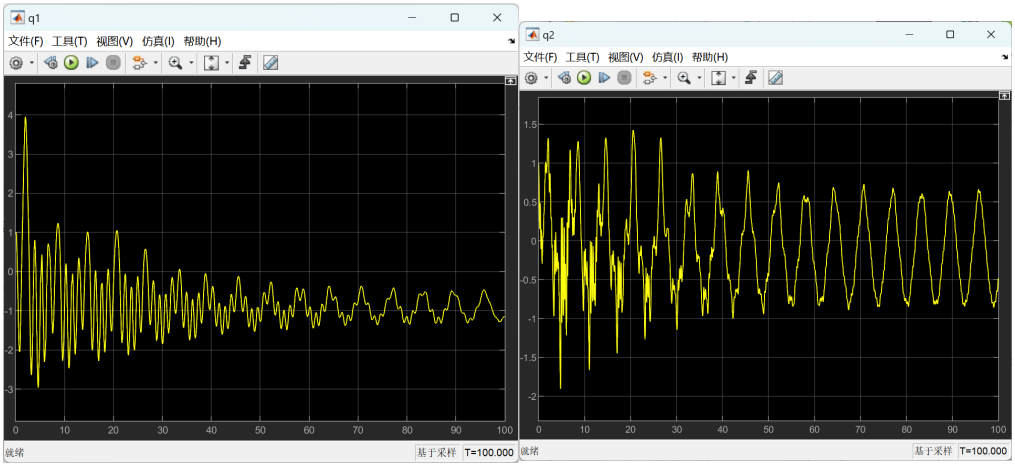
(This is the expectation value (the input))

Step1: set Kp be 1 and Ki and Kd be 0.

The initial wave is like this.

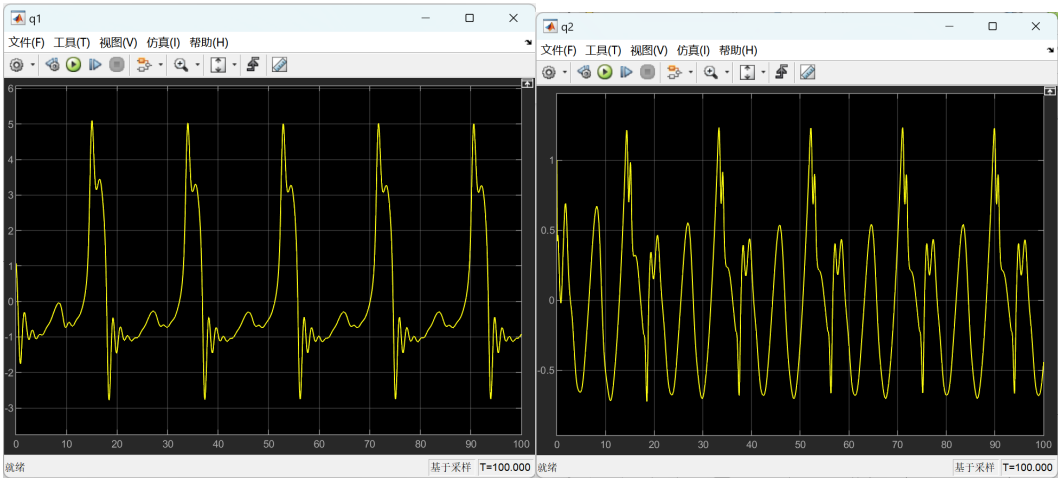


Step2: increase Kp， the wave become disorderd.



Step3: Increase Ki and Kd as well.

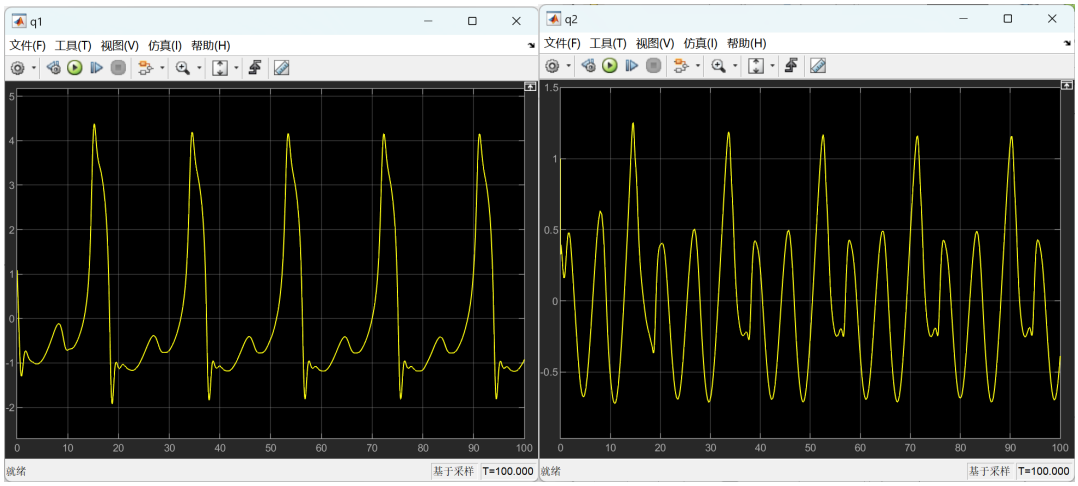
This is the wave when Kp=30,Ki=10,kd=10



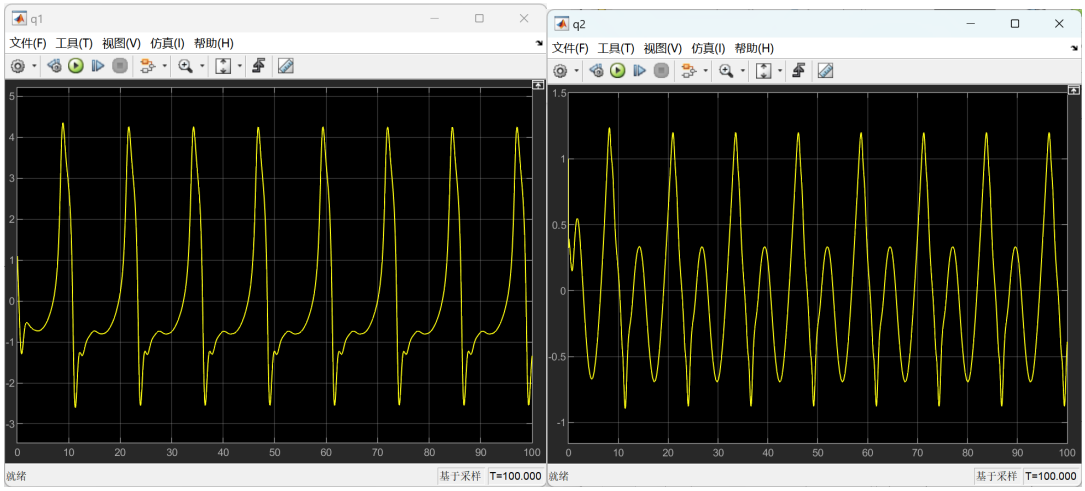
Step4: continue to increase Kd, found that if Kd become larger, the range of the wave will increase.

This is the wave when Kp=30,Ki=10,kd=20

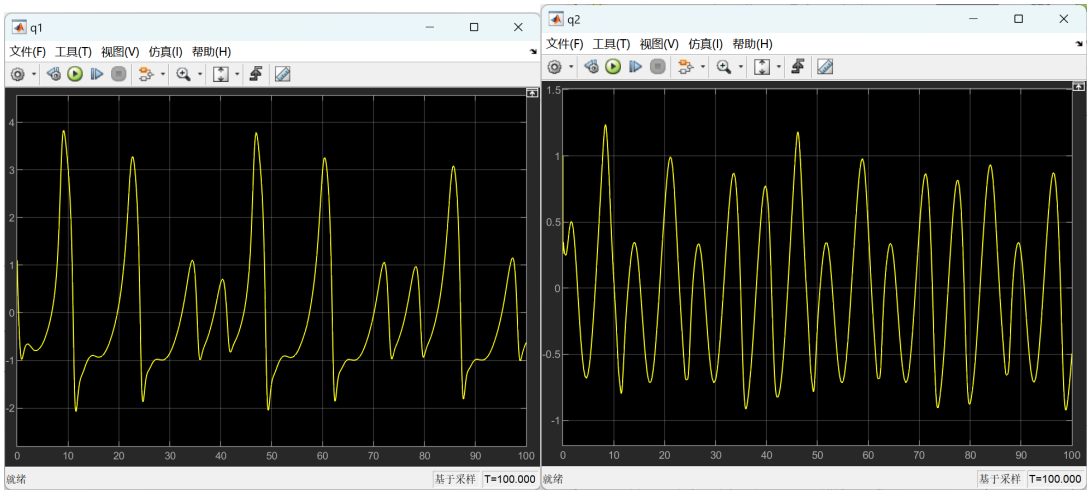
Compare to steep3 where Kd=10. This wave has a smaller range from -2 to 4.



Step5: we try to increase Ki and found that the difference is not bigger than increase Kd.
See the wave when Kp=30,Ki=20,kd=20

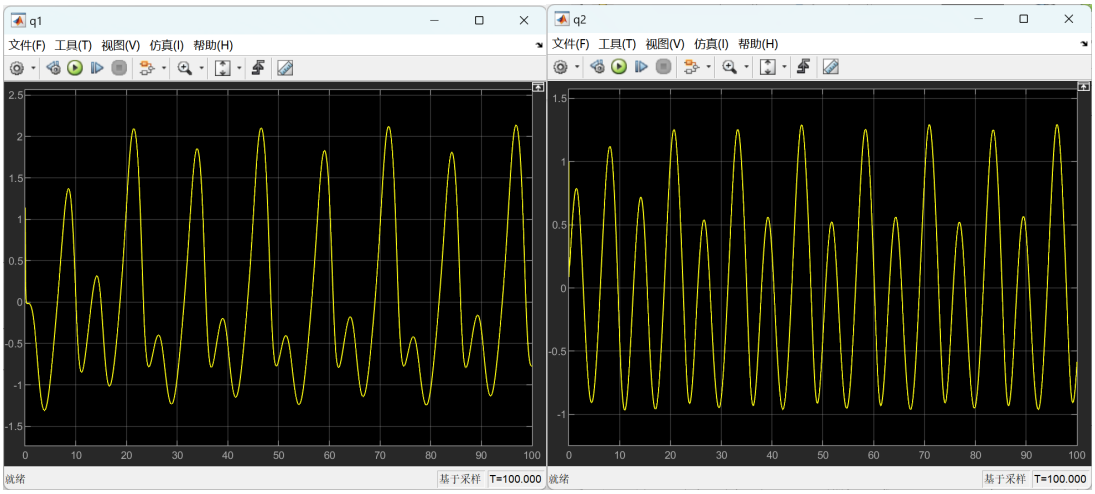


So we try to increase Kd more.
See the wave when Kp=30,Ki=20,kd=30

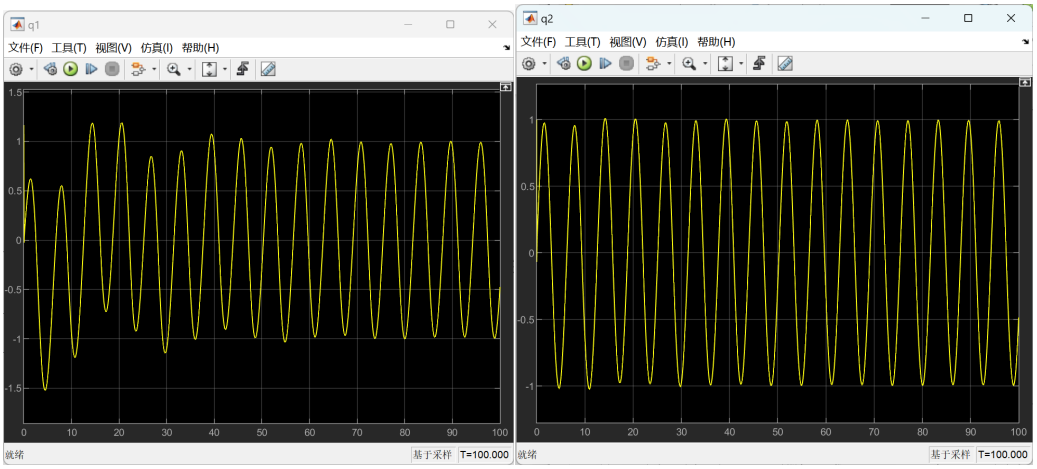


Step6: When we increase Kd more, up to more than 100
We found that the wave of q2 keep in around -1 to 1, but the wave of q1 is still disordered

See the wave when Kp=30,Ki=20,kd=100

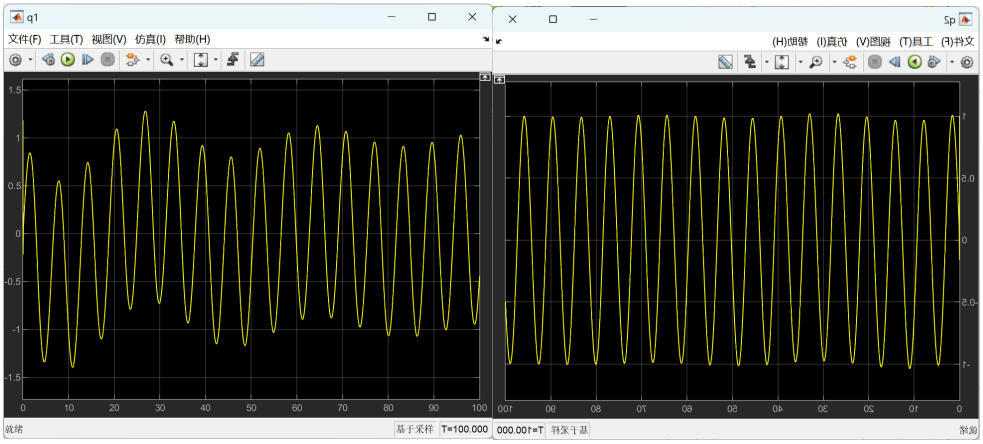


See the wave when Kp=30,Ki=20,kd=300

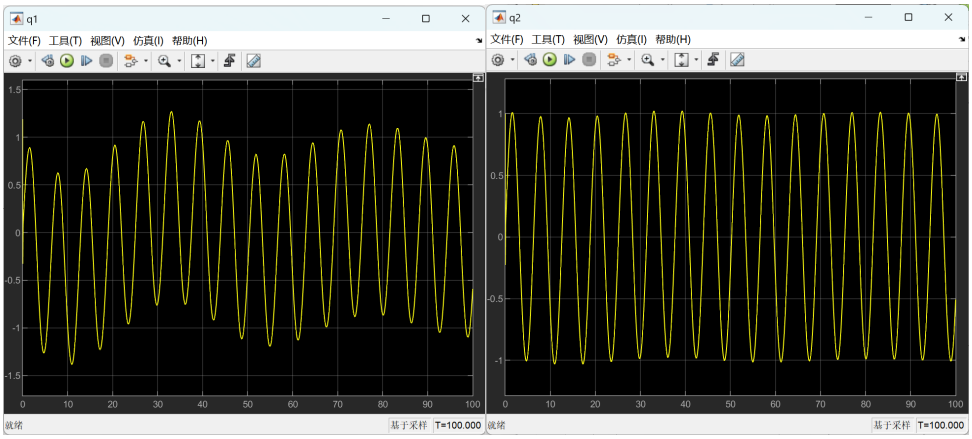


Step7: keep increasing the Kd, when Kd up to 1000, the wave of q2 show a trend of sin wave

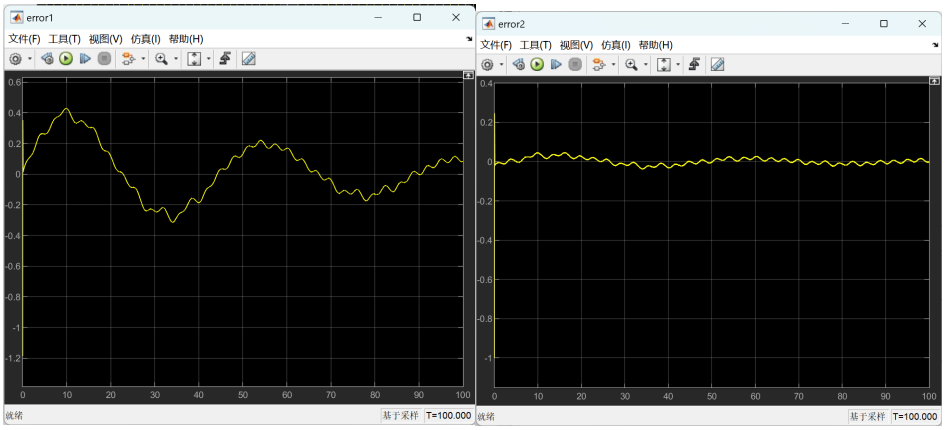
See the wave when Kp=30,Ki=20,kd=700



See the wave when Kp=30,Ki=20,kd=1000

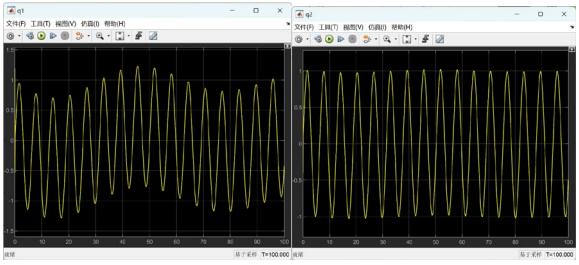


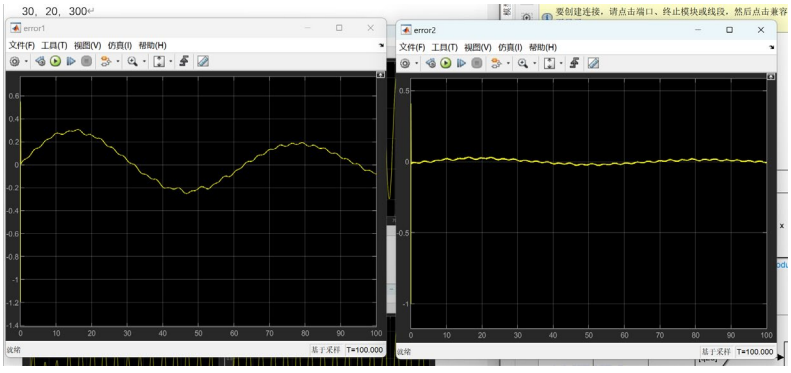
And the error is:



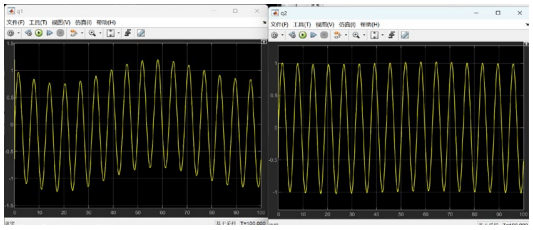
Step8: keep increasing the Kd, the wave of q1 and q2 are already stable and similar. But the error wave being more close to 0, when the Kd from 1000 to 3000

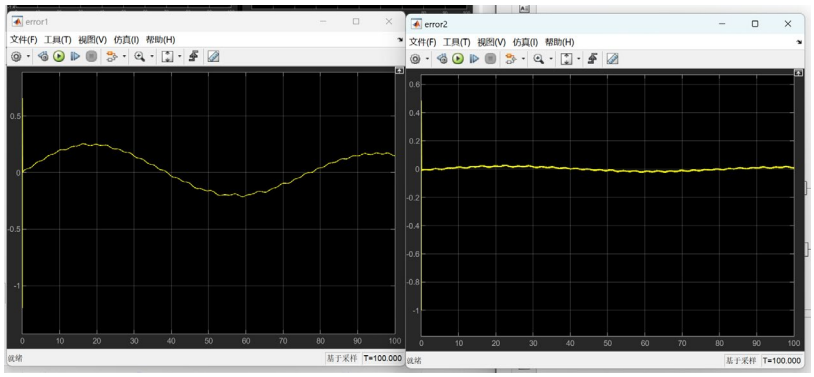
See the wave when Kp=30,Ki=20,kd=2000





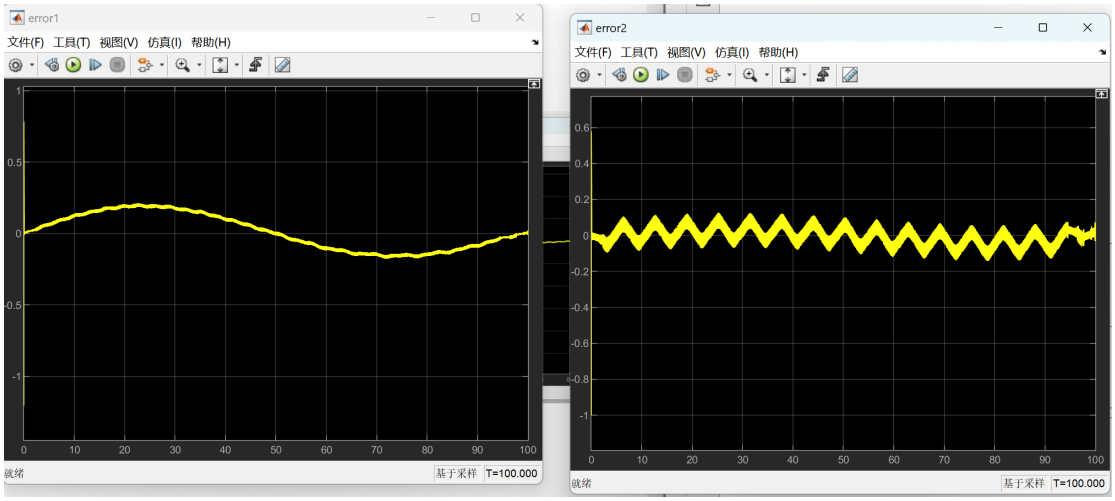
See the wave when Kp=30,Ki=20,kd=2000





But if the Kd too larger, it will be wrong

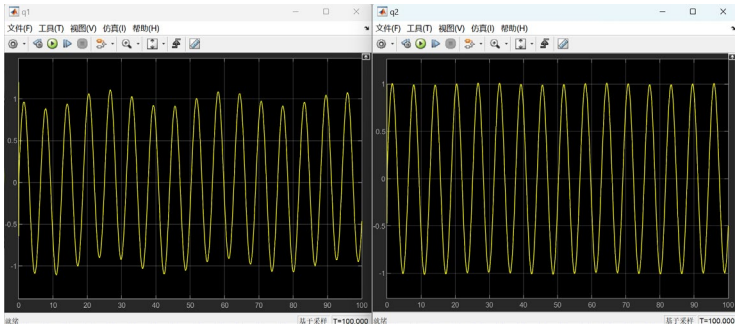
See when Kd = 5000

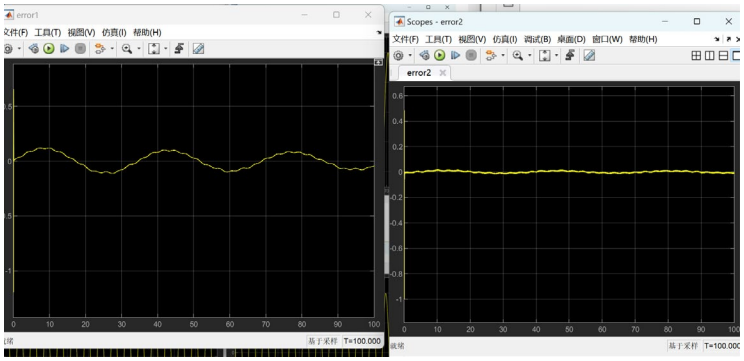


Step9:

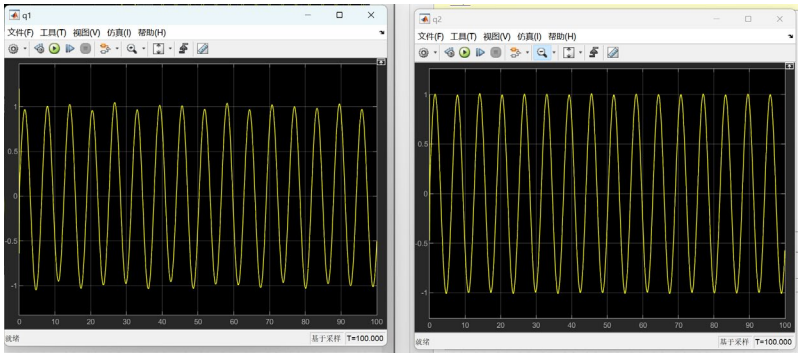
So we keep Kd=3000, and continue to change the Ki

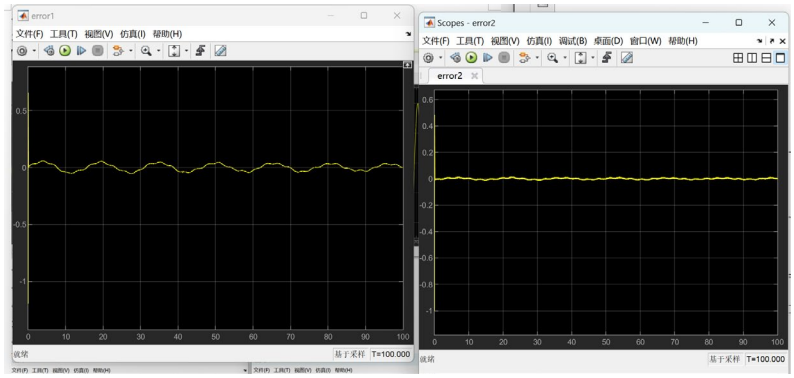
See the wave when Kp=30,Ki=100,kd=3000



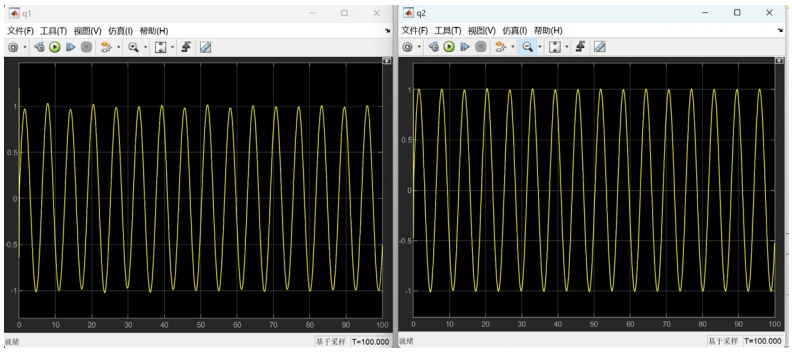


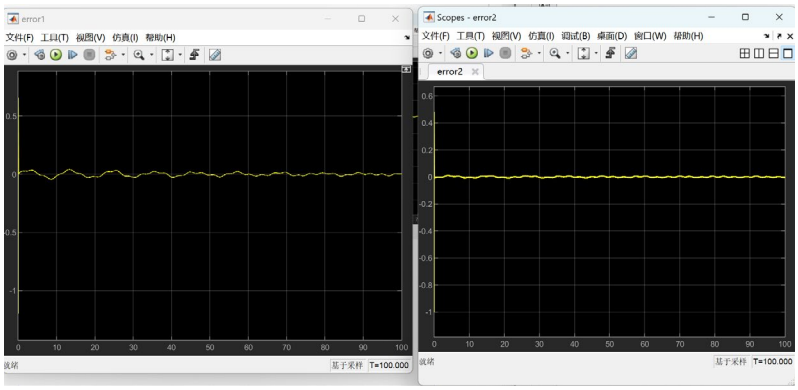
See the wave when Kp=30,Ki=500,kd=3000





See the wave when Kp=30,Ki=1000,kd=3000



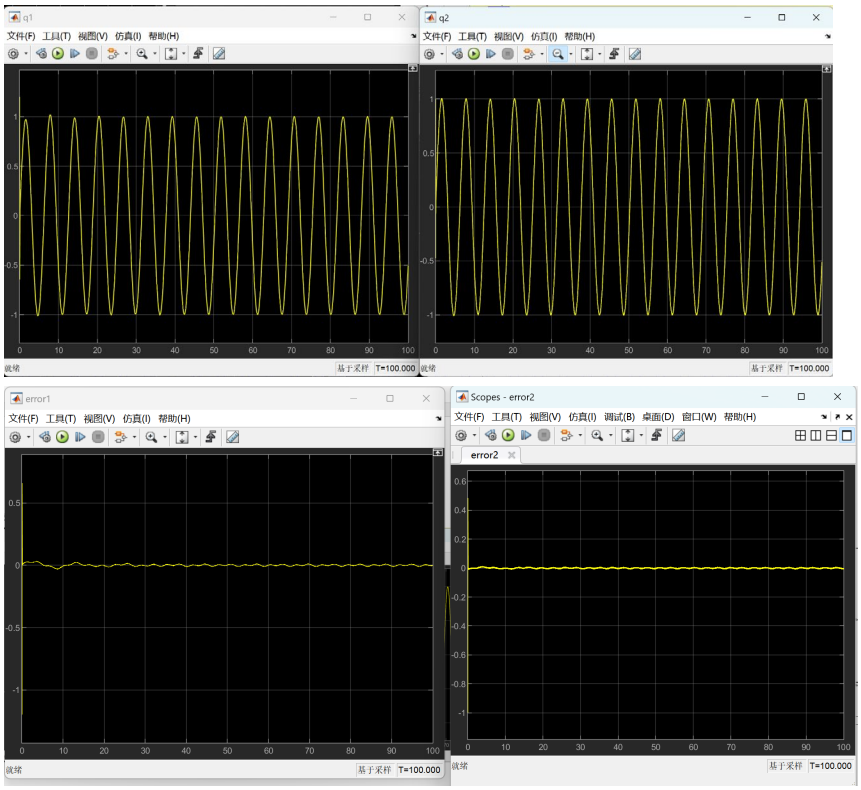


We can see that when Kp and Kd not changed, if Ki become larger up to 1000, the q1 wave become stable in the range -1 to 1 and error tend to close to 0.

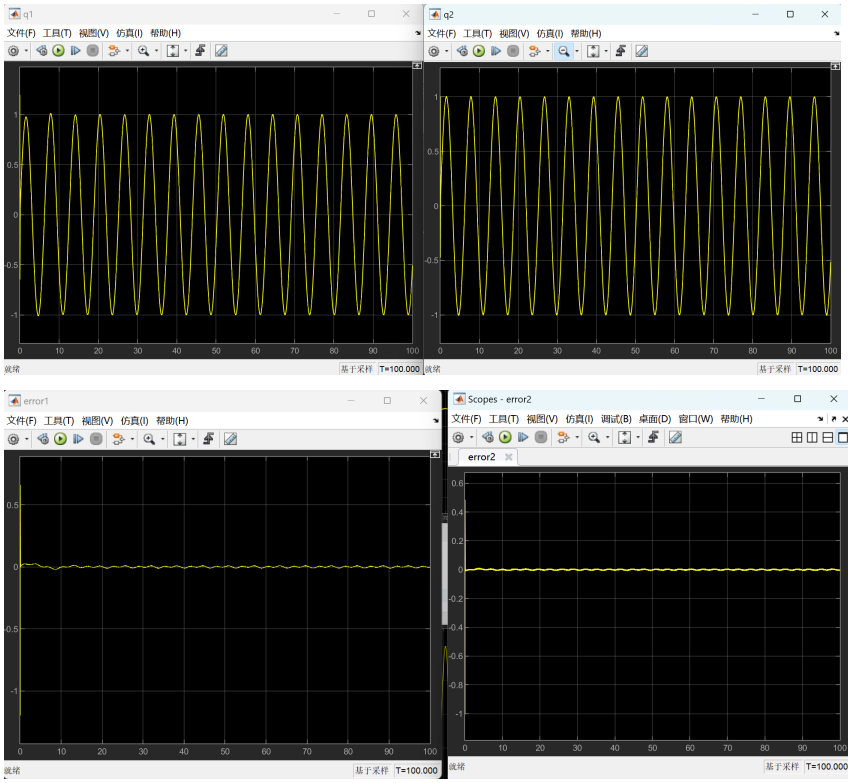
Step10: so we keep Ki=1000 and increase kp, we found that when kp more than 500 the error is tend to be a line which is close to 0. And when kp is up to 1000, the error is better.

Thus, we stop here and our PID is Kp=1000, Ki=500, kd=3000

See the wave when Kp=1000, Ki=500, kd=3000



See the wave when Kp=500,Ki=500,kd=3000

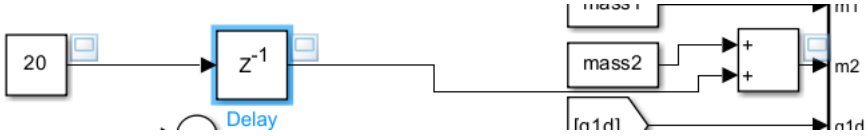


Task4

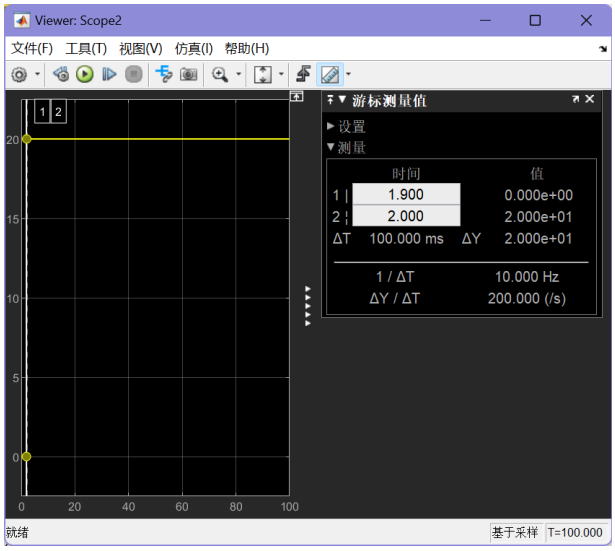
You can see the model in the file: PID_control_Model_sin_payload.slx

To add a payload at the time t=2s, we use delay model.

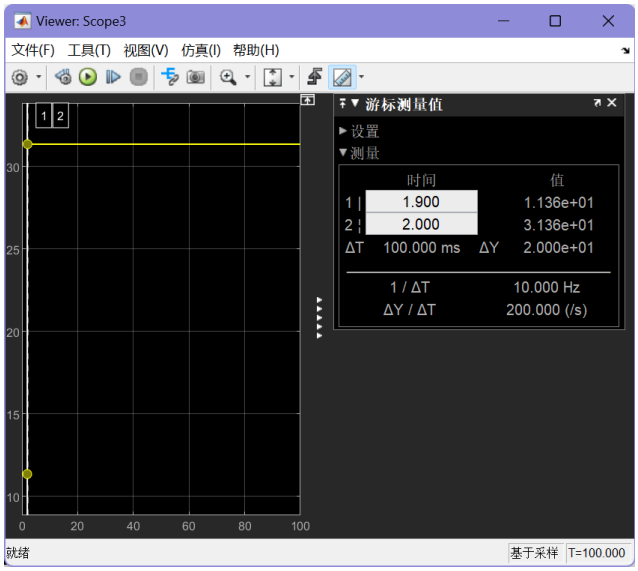
The initial value of mass2 is 11.36, we add an new constant whose value is 20 and then the total value of mass2 is 31.36. And then use delay which set a 2s delay time. In the end, we use a plus model to add the two values.



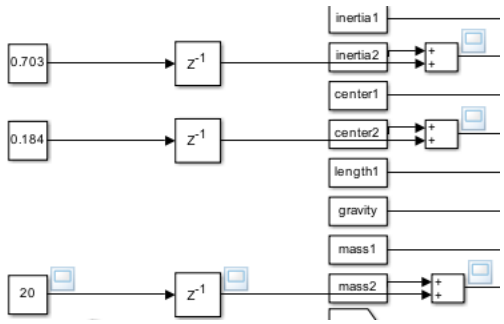
Here is the graph to see the value changing after 2s delay. You can see at the time of 2s, the value change from 0 to 20.



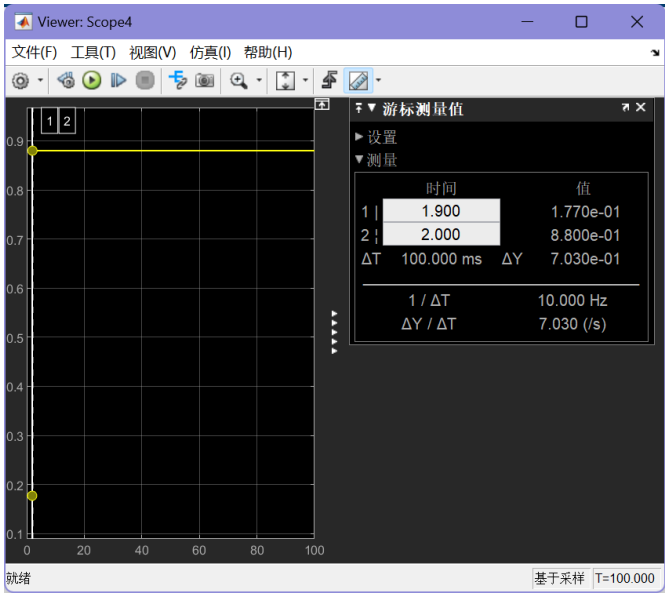
Here is the graph showing the value changing after the plus model. Before 2s, the value of mass2 is 11.36, after 2s the value become 31.36.



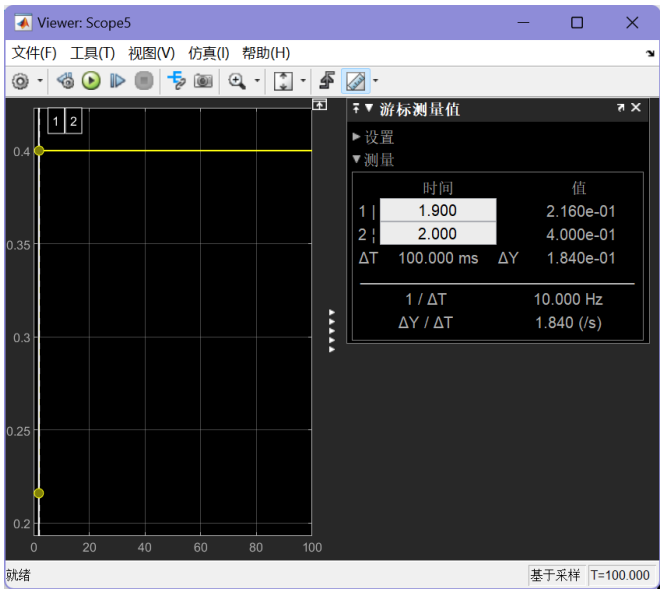
Similarly, we use the same method to change other parameters at the time t=2s.



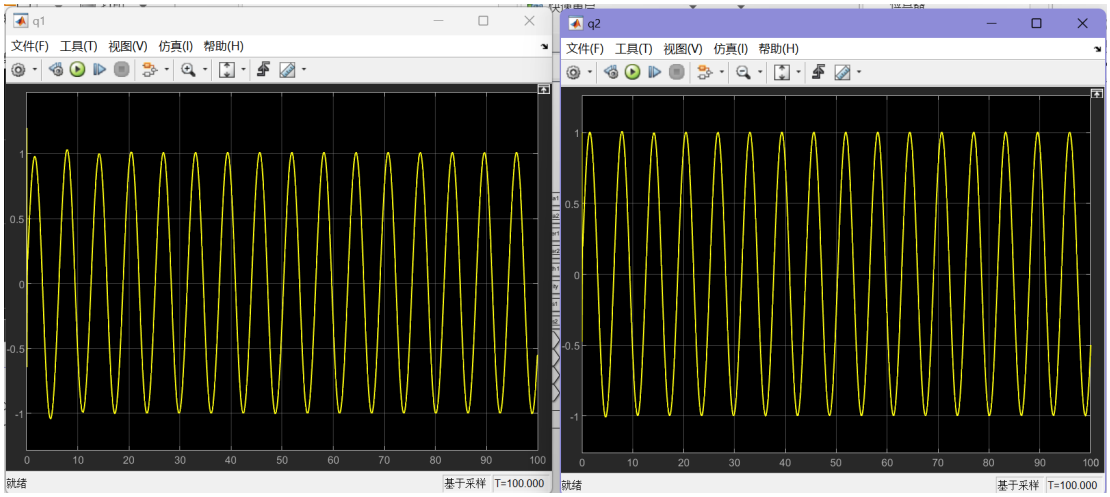
- Change the length 2 from 0.177 to 0.88.

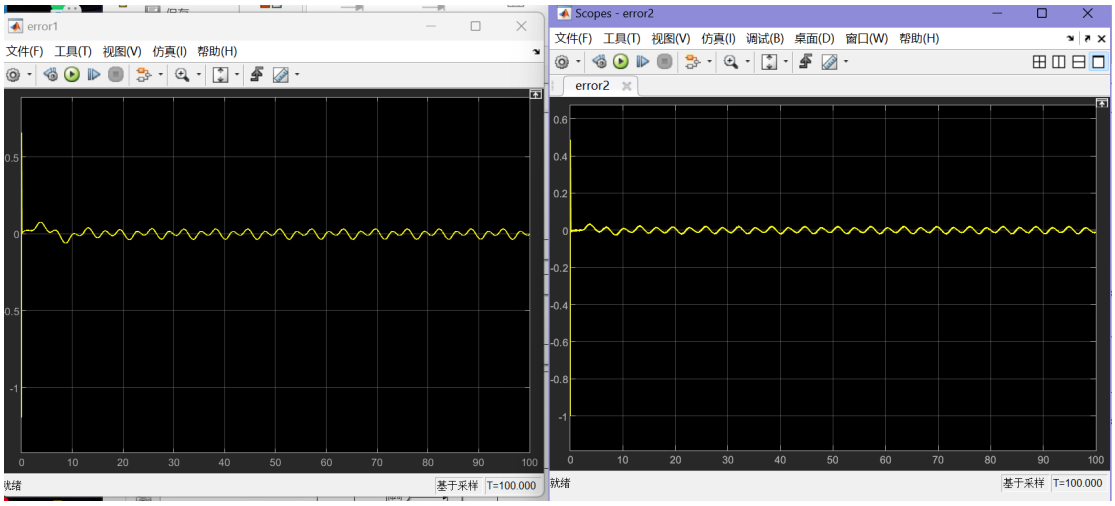


- Change the distance from the joint position 2 to the center of mass of link2 from 0.216 to 0.4



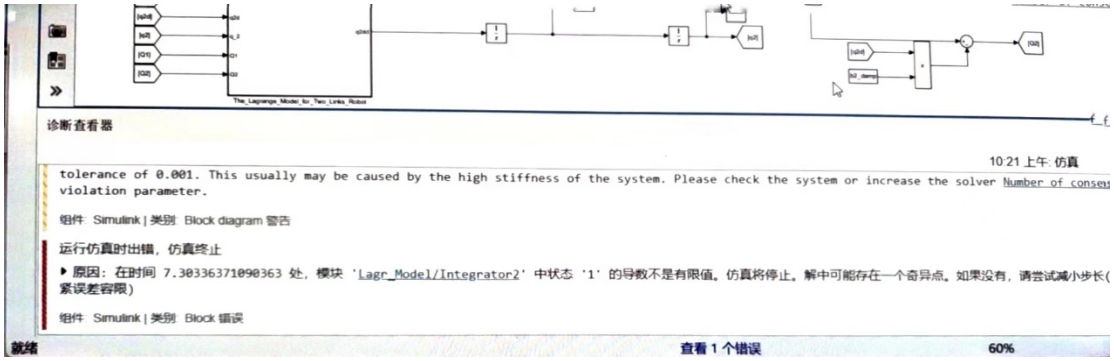
After adding the payload, the result does not have a big difference. But the error changed, and it is not stable anymore.





Model Evaluation

1. Some parameters of PID cannot do simulation because the solve may have a singular point.

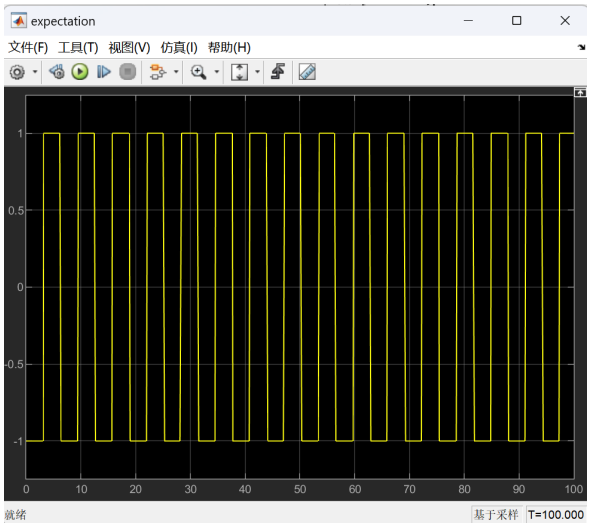


2. The Kd is very large. Although the larger the Kd is, the closer the steady state will be. There always exists some difference value.
3. When we use square input, the system is unlikely to be stable. We choose different

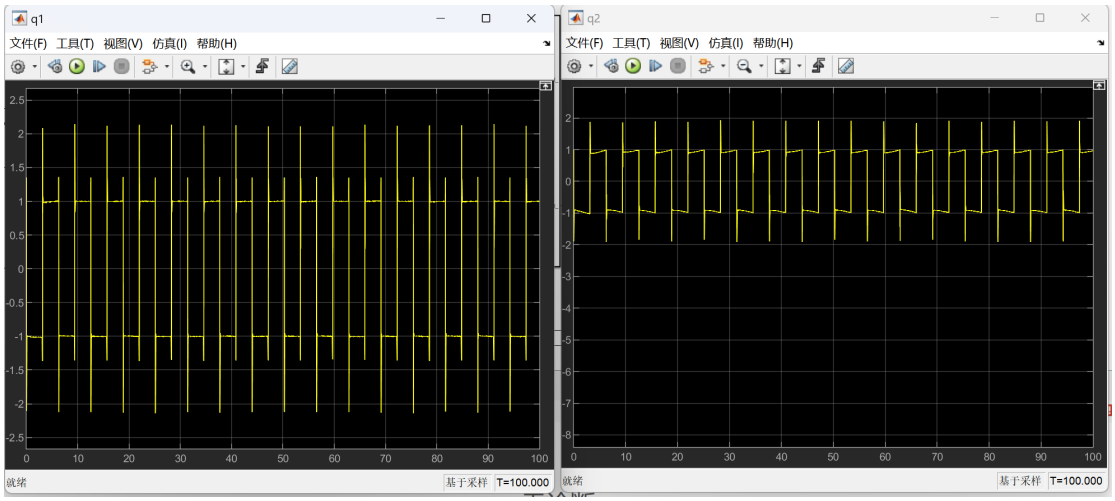
PID, and it is hard to meet the ideal state.

You can see the model in the file: PID_control_Model_square_input.slx

Input:



The two curves: when Kp=1000,



No matter how to increase Kd, the curve is almost the same.

And the error of the two joints is like that: