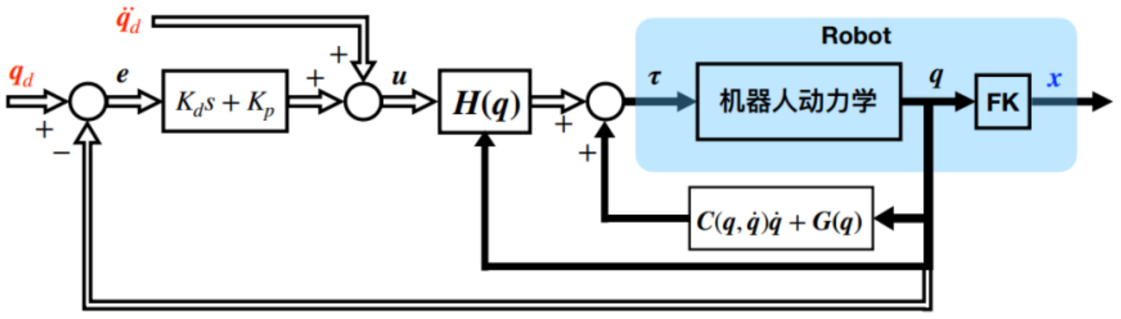
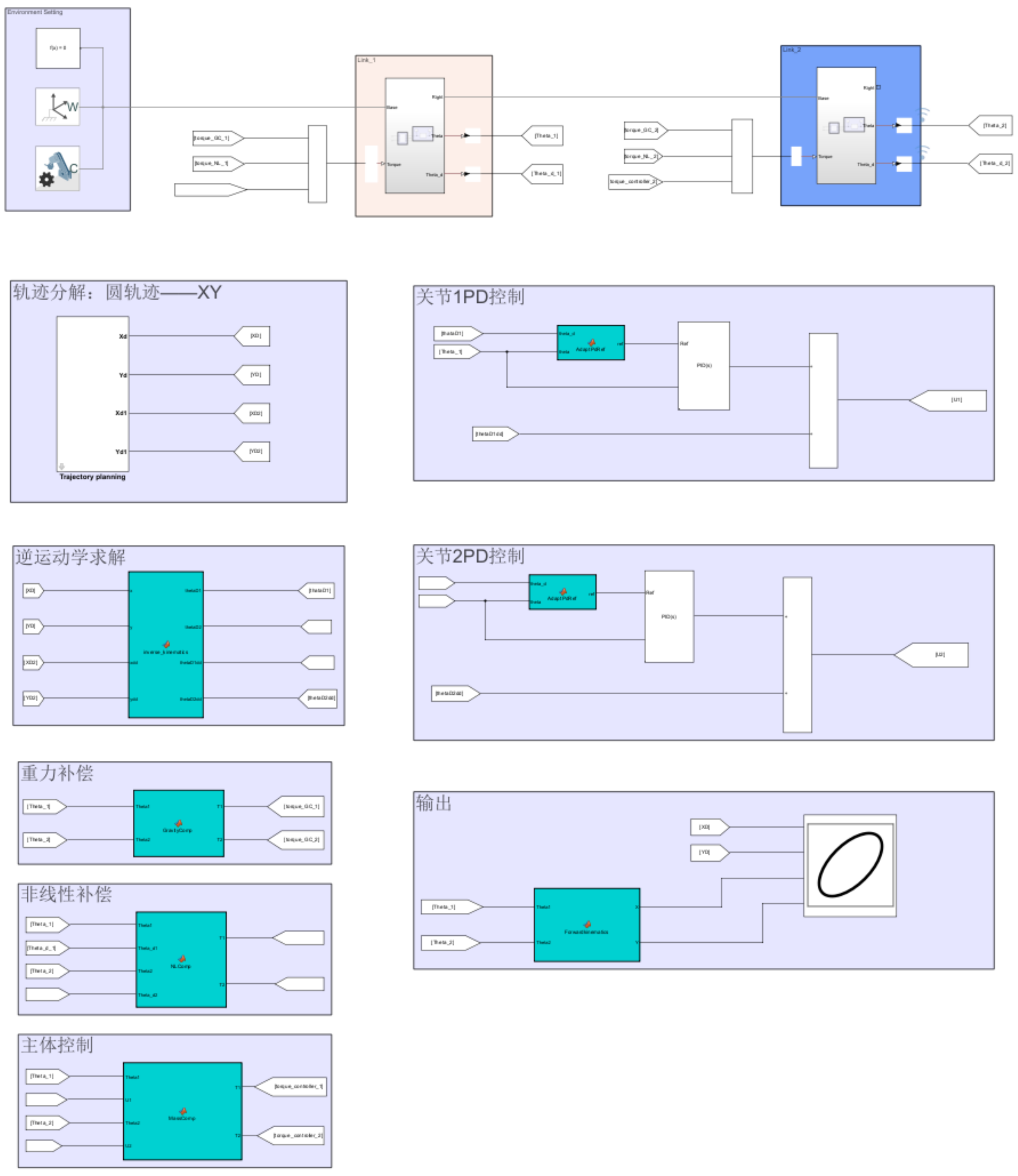
作业6：2DOF机械臂仿真

总览





逆运动学

求解加速度的雅可比矩阵以及其逆矩阵：

```
syms l1 l2 theta1 theta2 theta1d theta2d theta1dd theta2dd

% 定义机器人的坐标表示式
x = l1*cos(theta1) + l2*cos(theta1+theta2);
y = l1*sin(theta1) + l2*sin(theta1+theta2);

% 对坐标表示式分别求一阶导数，得到包含四个变量的 Jacobian 矩阵
J_v = jacobian([x, y],[theta1, theta2])
xd = J_v(1,:) * [theta1d; theta2d];
yd = J_v(2,:) * [theta1d; theta2d];

J_a = jacobian([xd, yd], [theta1d, theta2d])
xdd = J_a(1,:) * [theta1dd; theta2dd];
ydd = J_a(2,:) * [theta1dd; theta2dd];

J_v_inver = inv(J_v)
J_a_inver = inv(J_a)
```

求解逆运动学：

```
function [thetaD1,thetaD2,thetaD1dd,thetaD2dd] =
inverse_kinematics(x,y,xdd,ydd)

l1 = 0.25;
l2 = 0.25;
beta = atan2(y,x);
temp = (x^2 +y^2+l1^2-l2^2)/(2*l1*sqrt(x^2+y^2));
phi = acos(temp);
thetaD1 = beta - phi;
thetaD2 = 2*phi;

J_a_inv = [-cos(thetaD1 + thetaD2)/(l1*cos(thetaD1 + thetaD2)*sin(thetaD1) -
l1*sin(thetaD1 + thetaD2)*cos(thetaD1)), -sin(thetaD1 +
thetaD2)/(l1*cos(thetaD1 + thetaD2)*sin(thetaD1) - l1*sin(thetaD1 +
thetaD2)*cos(thetaD1));
(l2*cos(thetaD1 + thetaD2) + l1*cos(thetaD1))/(l1*l2*cos(thetaD1 +
thetaD2)*sin(thetaD1) - l1*l2*sin(thetaD1 + thetaD2)*cos(thetaD1)),
(l2*sin(thetaD1 + thetaD2) + l1*sin(thetaD1))/(l1*l2*cos(thetaD1 +
thetaD2)*sin(thetaD1) - l1*l2*sin(thetaD1 + thetaD2)*cos(thetaD1))];

thetaD1dd = J_a_inv(1,:) * [xdd;ydd];
thetaD2dd = J_a_inv(2,:) * [xdd;ydd];
```

重力补偿G

重力补偿力矩：

```
function [T1,T2] = GravityComp(Theta1, Theta2)

g = 9.80665;

m1 = 1.0;
L1 = 0.25;
Lc1 = 0.125;

m2 = 1.0;
L2 = 0.25;
Lc2 = 0.125;

T2 = m2 * g * Lc2 * cos(Theta1 + Theta2);
T1 = T2 + (m1 * Lc1 + m2 * L1) *cos(Theta1) * g;
```

非线性补偿C

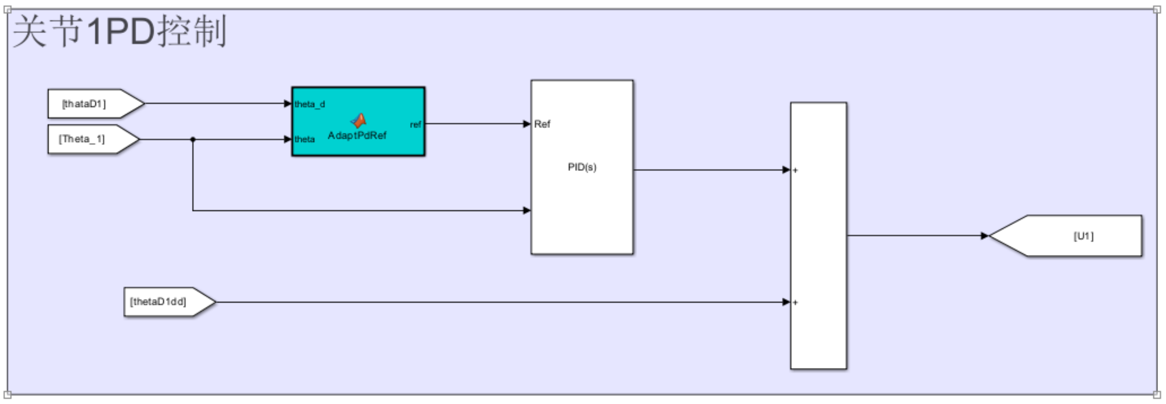
非线性补偿力矩：

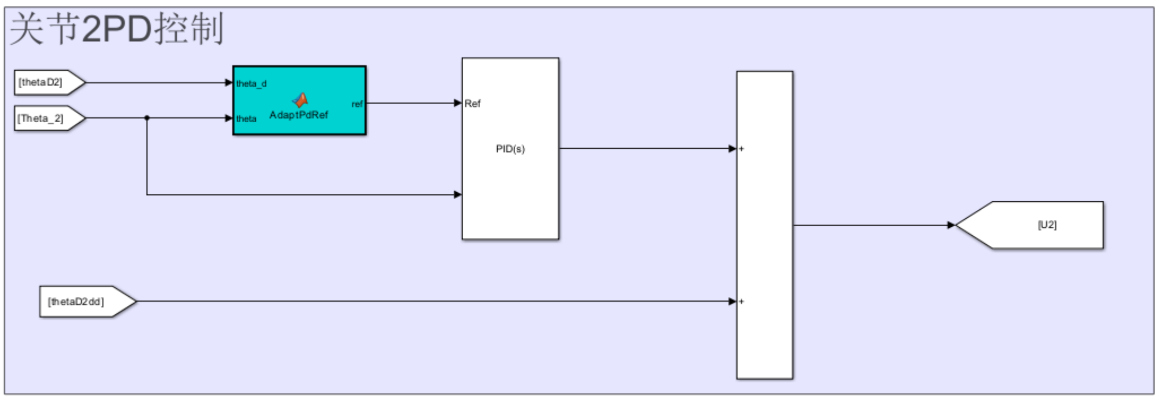
```
function [T1, T2] = NLComp(Theta1, Theta_d1, Theta2, Theta_d2)

g = 9.80665;

m1 = 1.0;
L1 = 0.25;
Lc1 = 0.125;

m2 = 1.0;
L2 = 0.25;
Lc2 = 0.125;

T1 = -2 * Theta_d2 * Theta_d1 * L1 * Lc2 * m2 * sin(Theta2) - 2 * (Theta_d1 +
Theta_d2) * Theta_d2 * L1 * Lc2 * m2 * sin(Theta2);
T2 = 2 * m2 * L1 * Lc2 * sin(Theta2) * Theta_d1^2;
```

主体控制H

```
function [T1, T2] = MassComp(Theta1, U1, Theta2, U2)

g = 9.80665;

m1 = 1.0;
L1 = 0.25;
Lc1 = 0.125;
Ic1 = 0.020833;

m2 = 1.0;
L2 = 0.25;
Lc2 = 0.125;
Ic2 = 0.020833;

T1 = (m2*L1^2 + 2*m2*cos(Theta2)*L1*Lc2 + m1*Lc1^2 + m2*Lc2^2 + Ic1 + Ic2) * U1
+ (m2*Lc2^2 + L1*m2*cos(Theta2)*Lc2 + Ic2) * U2;
T2 = (m2*Lc2^2 + L1*m2*cos(Theta2)*Lc2 + Ic2) * U1 + (m2*Lc2^2 + Ic2) * U2;
```

PD控制器

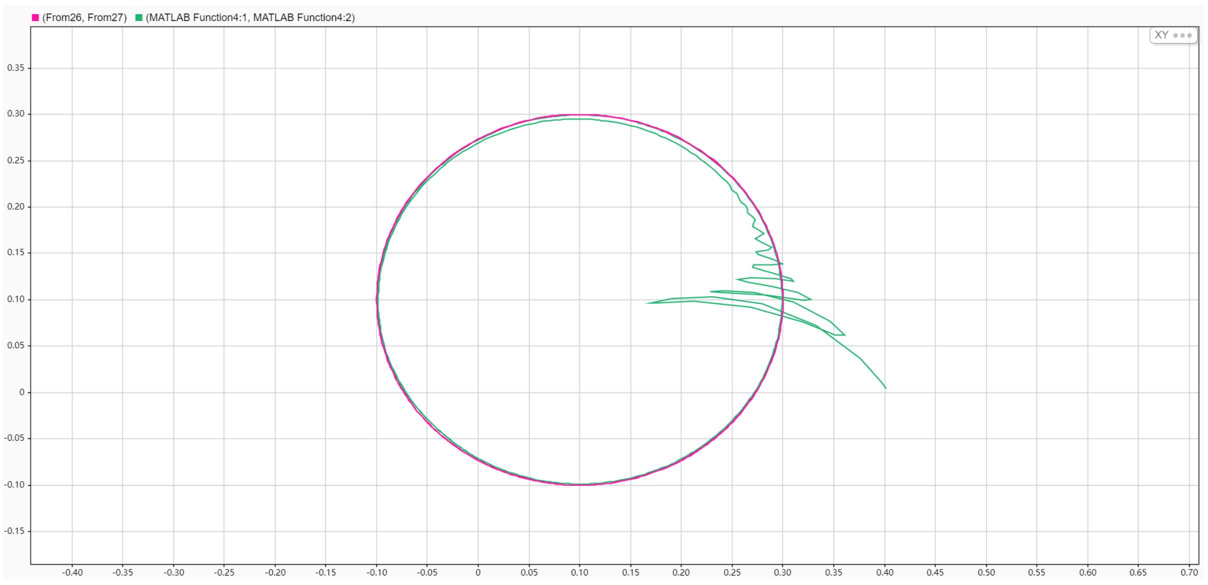




角度范围控制：

```
function ref = AdaptPdRef(theta_d, theta)

while (theta_d - theta) >= pi
theta_d = theta_d - 2 * pi;
end

while (theta_d - theta) <= -pi
theta_d = theta_d + 2 * pi;
end

ref = theta_d;
```

最终结果



可以看到从初始位置先向圆的初始位置靠近，最后稳定于圆轨迹上，完成了控制目标