Robot car LQR control

LQR (Linear Quadratic Regulator) is a commonly used high-order control method that can help the balancing car achieve stable movement.

LQR

LQR (Linear Quadratic Regulator) is a classic linear quadratic regulator used to design the optimal state feedback controller of continuous-time linear systems.

The performance of the system is optimized by minimizing a cost function, while considering the state and control input of the system.

Implementation ideas

System modeling

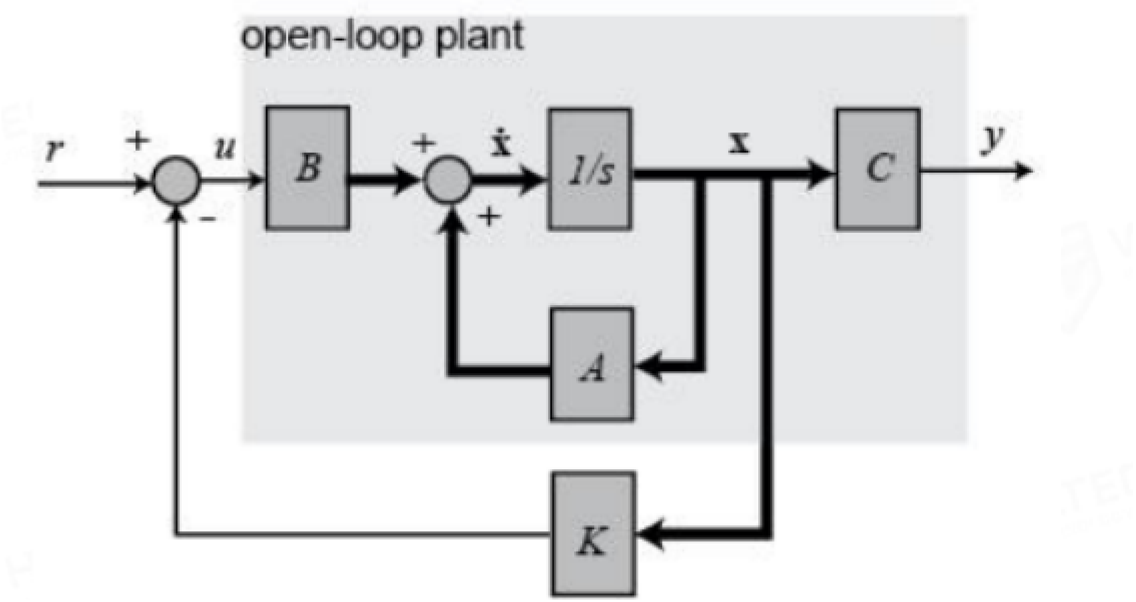
Establish a mathematical model of the system, usually using state space representation. Assume that the state space expression of the system (requires that the system is fully controllable) is:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

```
The state space equation can describe the dynamic characteristics of the system,
including state variables, input variables, system matrix, and output matrix.
```

Cost function design

It is used to evaluate the performance of the system under different states; usually the cost function consists of two parts: state error term and control input term.




```
The cost function can be designed according to specific application requirements,
such as maintaining stability, fast response, etc.
```

Solve the Riccati equation

The LQR method needs to solve the Riccati equation to calculate the optimal state feedback gain matrix.

```
The Riccati equation is a nonlinear algebraic equation that can be solved by
iteration or numerical methods
```

Calculate the controller

According to the optimal state feedback gain matrix obtained by solving, the optimal controller can be calculated.

```
The controller will adjust the control input in real time according to the
current system state to achieve the best performance of the system
```

Implementation and debugging

Implement the designed LQR controller into the actual system, and test and debug it.

```
By continuously optimizing parameters and adjusting strategies, the system can
achieve better stability, robustness and performance
```

Practical application

From the dynamic model of the balancing car:

$$A=\begin{pmatrix}0&1&0&0&0&0\\0&0&A_{23}&0&0&0\\0&0&0&1&0&0\\0&0&A_{43}&0&0&0\\0&0&0&0&0&1\\0&0&0&0&0&0\end{pmatrix},B=\frac{I}{r}\begin{pmatrix}0&0\\B_{21}&B_{22}\\0&0\\B_{41}&B_{42}\\0&0\\B_{61}&B_{62}\end{pmatrix},x=\begin{pmatrix}x\\\dot{x}\\\theta_P\\\dot{\theta}_P\\\delta\\\dot{\delta}\end{pmatrix},u=\begin{pmatrix}\dot{v_{LO}}\\\dot{v_{RO}}\end{pmatrix}$$

式中

$$A_{23}=-\frac{M^2l^2g}{Q_{eq}} \qquad A_{43}=\frac{Mlg\left(M+2m+\frac{2I}{r^2}\right)}{Q_{eq}}$$

$$B_{21}=\frac{J_P+Ml^2+Mlr}{Q_{eq}r} \qquad B_{22}=\frac{J_P+Ml^2+Mlr}{Q_{eq}r}$$

$$B_{41}=-\frac{\left(\frac{Ml}{r}+M+2m+\frac{2I}{r^2}\right)}{Q_{eq}} \qquad B_{42}=-\frac{\left(\frac{Ml}{r}+M+2m+\frac{2I}{r^2}\right)}{Q_{eq}}$$

$$B_{61}=\frac{1}{r\left(md+\frac{Id}{r^2}+\frac{2J_\delta}{d}\right)} \qquad B_{62}=-\frac{1}{r\left(md+\frac{Id}{r^2}+\frac{2J_\delta}{d}\right)}$$

其中

$$Q_{eq}=J_PM+(J_P+Ml^2)\left(2m+\frac{2I}{r^2}\right).$$

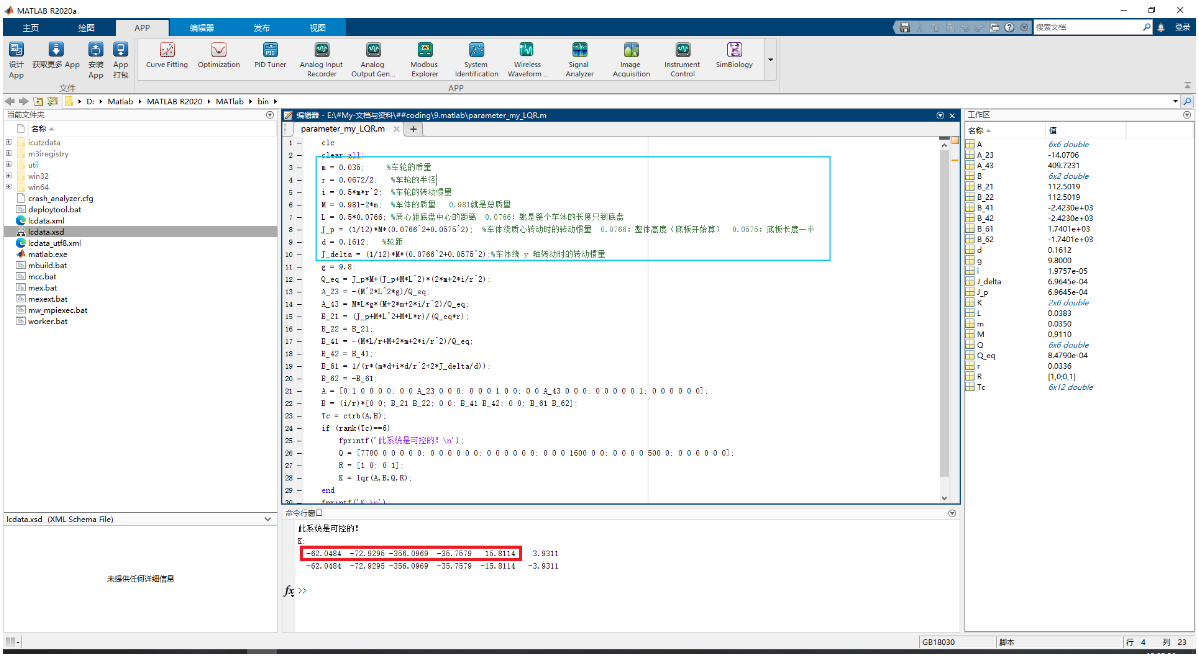
Use MATLAB's lqr function for calculation:

```
clc
clear all;
m = 0.035; %The mass of the wheel
r = 0.0672/2; %The radius of the wheel
i = 0.5*m*r^2; %The moment of inertia of the wheel
M = 0.981-2*m; %The mass of the car body 0.981 is the total mass
L = 0.5*0.0766; %The distance from the center of mass to the center of the
chassis 0.0766: that is, the length of the entire car body is only to the chassis
J_p = (1/12)*M*(0.0766^2+0.0575^2); %Moment of inertia when the vehicle rotates
around the center of mass 0.0766: Overall height (starting from the bottom plate)
0.0575: Half the length of the bottom plate
d = 0.1612; %Wheelbase
J_delta = (1/12)*M*(0.0766^2+0.0575^2); %Moment of inertia when the vehicle
rotates around the y axis
g = 9.8;
Q_eq = J_p*M+(J_p+M*L^2)*(2*m+2*i/r^2);
A_23 = -(M^2*L^2*g)/Q_eq;
A_43 = M*L*g*(M+2*m+2*i/r^2)/Q_eq;
B_21 = (J_p+M*L^2+M*L*r)/(Q_eq*r); B_22 = B_21; B_41 = -
(M*L/r+M+2*m+2*i/r^2)/Q_eq; B_42 = B_41; B_61 = 1/(r*(m*d+i*d/r^2+2*J_delta/d));
B_62 = -B_ 61; A = [0 1 0 0 0 0; 0 0 A_23 0 0 0; 0 0 0 1 0 0; 0 0 A_43 0 0 0; 0 0
0 0 0 1; 0 0 0 0 0 0]; B = (i/r)*[0 0; B_21 B_22; 0 0; B_41 B_42; 0 0; B_61
B_62];
Tc = ctrb(A,B);
if (rank(Tc)==6)
fprintf('This system is controllable!\n');
```

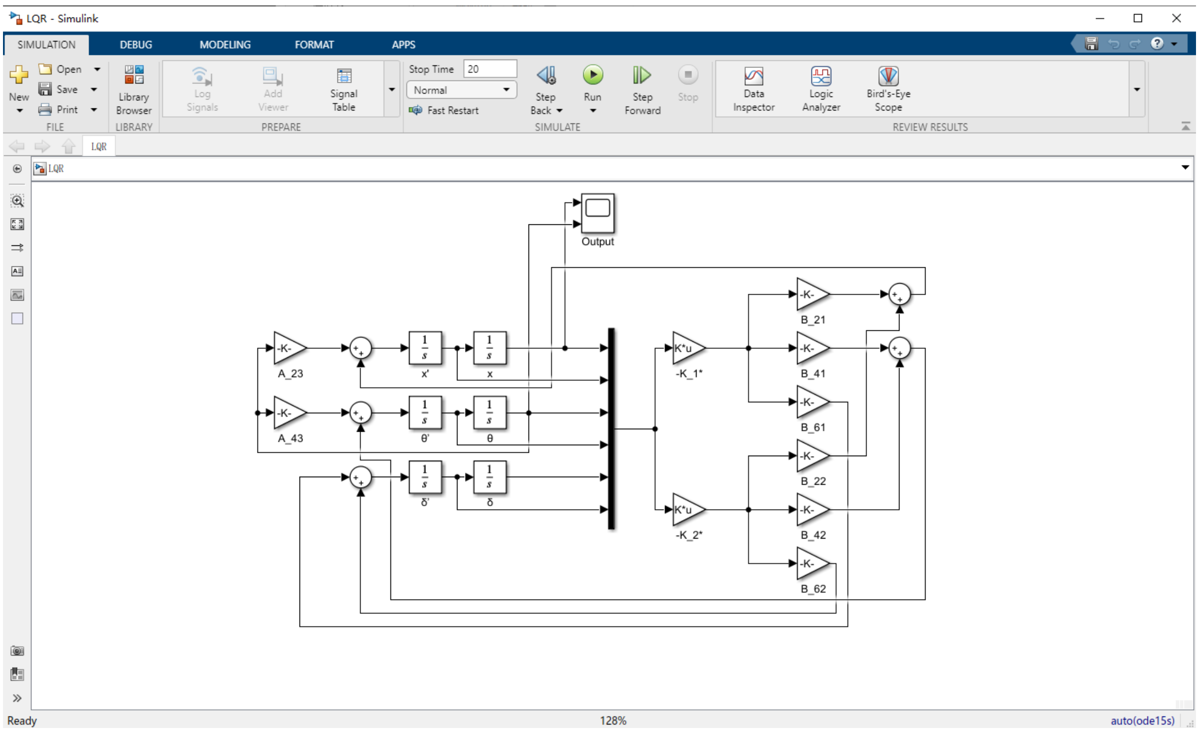
```
Q = [7700 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 1600 0 0; 0 0 0 0 500 0; 0 0
0 0 0 0];
R = [1 0; 0 1];
K = lqr(A,B,Q,R);
end
fprintf('K:\n');
disp(K);
```

The program's running results are as follows: You need to use MATLAB to run the program parameter_LQR.m, and modify the corresponding parameters in the program to adapt to your own car

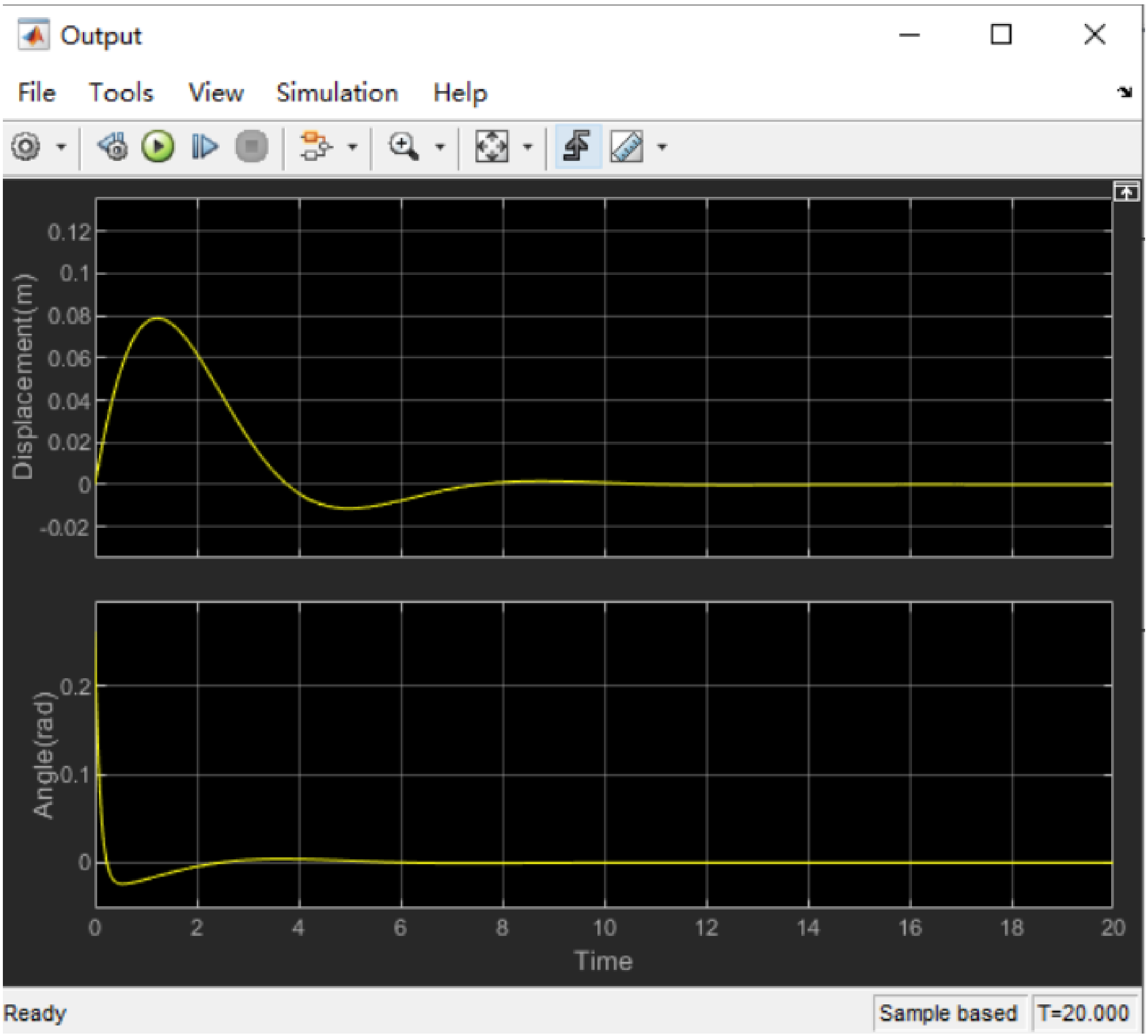
```
Product supporting materials source code path: Attachment → Source code summary →
6.LQR → Matlab
```



Use SIMULINK to simulate it:



Simulation results:



Code implementation

When running this case code, the balancing car can only retain the standard package accessories, and other advanced accessories need to be removed; keep the balancing car vertical to the ground at 90° before starting.

```
Product supporting information source code path: Attachment → Source code summary → 6.LQR → STM32_code
```

> **Implementation code**

```
//LQR state feedback coefficient
float K1= -62.1608, K2= -72.9295, K3=-356.0969, K4=-35.7579, K5=15.8114,
K6=15.8114;
float K5OLD=15.8114, K6OLD=15.8114; L_accel=-(K1*x_pose+K2*(x_speed-
Target_x_speed)+K3*(angle_x-Target_angle_x)+K4*gyro_x+K5*angle_z+K6*(gyro_z-
Target_gyro_z)); R_accel=-(K1*x_pose+K2*(x_speed-Target_x_ speed)+K3*(angle_x-
Target_angle_x)+K4*gyro_x-K5*angle_z-K6*(gyro_z-Target_gyro_z)); ```
```