基于 LQR 的平衡小车

1.LQR 控制器

1.1 概述

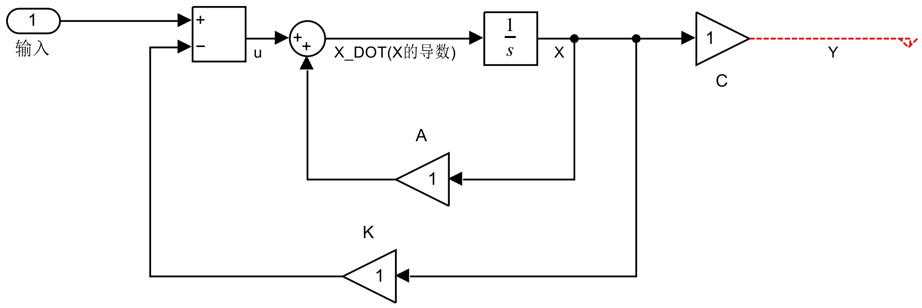
LQR 即线性二次型调节器，其基于贝尔曼最优原则，是一种用于线性系统的最优控制器。

设一个系统完全能控的状态方程为：

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y = Cx + Du$$

通常情况下，最基本的控制系统是需要全状态反馈控制的，而目标一般是设计一个状态反馈控制器，即 u=-Kx 来控制系统的表现：



1.2 最优控制

然而 K 并不是唯一的，那么什么情况下 K 才是最优的呢？

在这样的前提下需要引入一个评价标准，对于控制系统，一般使用代价函数（Quadratic Cost Function）来作为其的评价标准，代价函数越小，说明性能指标越好，反之则越差。

代价函数的一般形式如下:

$$J = \int_0^{+\infty} (x^{\mathrm{T}} Q x + u^{\mathrm{T}} R u)\, dt$$

其中 Q 为 n*n 的半正定的状态加权矩阵，R 为 n*n 的正定的控制加权矩阵。Q,R 在工程中常为兑成对角矩阵，其对角线上的元素a_i表示对矩阵中对应分量x_i的重视程度，a_i越大则对x_i的重视程度越大，其在代价函数中便会更快趋向更小的值。

LQR 的思路是设计一个反馈控制器 u=-Kx,使得代价函数最小，即min_J, K 矩阵的最小值的表达形式是这样的

已知有状态方程和代价函数:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$J = \int_0^{\infty} \left(x^{\mathrm{T}}(t) Q x(t) + u^{\mathrm{T}}(t) R u(t)\right) dt$$

所以对于最优的 K 矩阵可以表示为如下:

$$K = R^{-1} B^T P$$

其中 P 是如下方程的解

$$A^T P + PA + Q - PBR^{-1} B^T P = 0$$

在实际应用中，出于效率的考量，往往不使用人工的方法计算 K，一般使用计算机软件辅助计算 K，在 matlab 中一般使用库函数来计算

2.小车的数学模型

两轮小车的结构由车体和双轮组成，可以简化的看作一个移动倒立摆，分别对小车的车轮和车体进行力学分析，可以通过其建立系统的状态空间表达式。

2.1 小车的状态方程

在对小车进行物理分析后可以建立小车的状态方程，因为在小车的运动全过程中小车的倾角一直很小，所以可以认为 $sin\theta_p$=1,$cos\theta_p = 0$,$\dot{\theta}_p$=0，所以有如下状态方程：

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta}_p \\ \ddot{\theta}_p \\ \dot{\delta} \\ \ddot{\delta} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_{23} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & A_{43} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \theta_p \\ \dot{\theta}_p \\ \delta \\ \dot{\delta} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ B_{21} & B_{22} \\ 0 & 0 \\ B_{41} & B_{42} \\ 0 & 0 \\ B_{61} & B_{62} \end{pmatrix} \begin{pmatrix} T_L \\ T_R \end{pmatrix}$$

其中状态变量从上至下依次表示 小车的位移，前进速度，车体的倾角，车体的角速度，小车的转向角，转向速度

电机的扭矩和车轮的加速度有线性关系如下，因此可以转化为：

$\dot{v_{LO}}$=rT_L/I

$\dot{v_{RO}}$=rT_R/I

其中

v_{LO} 左轮无摩擦力时的线速度大小（rad/s）

v_{RO} 右轮无摩擦力时的线速度大小（rad/s）

故而，对于系统的状态空间表达式

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y = Cx + Du$$

其中各参数的值如下

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=(l/r)$\begin{pmatrix} 0 & 0 \\ B_{21} & B_{22} \\ 0 & 0 \\ B_{41} & B_{42} \\ 0 & 0 \\ B_{61} & B_{62} \end{pmatrix}$ C=$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$ D=0

x=$\begin{pmatrix} x \\ \dot{x} \\ \theta_p \\ \dot{\theta_p} \\ \delta \\ \dot{\delta} \end{pmatrix}$ u=$\begin{pmatrix} \dot{v_{LO}} \\ \dot{v_{RO}} \end{pmatrix}$

A_{23}=$-M^2l^2g/Q_{eq}$

A_{43}=$Mlg(M+2m+2I/r^2)/Q_{eq}$

B_{21}=$(J_P+Ml^2+Mlr)/Q_{eq}$

B_{22}=$(J_P+Ml^2+Mlr)/Q_{eq}$

B_{41}=$\big((Ml/r)+M+2m+(2I/r^2)\big)/Q_{eq}$

B_{42}=$\big((Ml/r)+M+2m+(2I/r^2)\big)/Q_{eq}$

B_{61}=$1/\big(r(md+Id/r^2+2J_\delta/d)\big)$

B_{62}=$1/\big(r(md+Id/r^2+2J_\delta/d)\big)$

Q_{eq}=$J_PM+(J_P+Ml^2)(2m+2I/r^2)$

m 车轮的质量（kg）

r 车轮的半径（m）

x_R 车轮的水平位移（m）

H_{fR} 车轮收到地面的摩擦力（N）

H_R 车轮收到车体作用力的水平方向的分力（N）

T_R 车轮电机输出的转矩（N*m）

I 车轮的转动惯量（kg*m^2）

ω_R 车轮的角速度（rad/s）

M 车轮的质量（kg）

l 质心距离底盘中心的距离（m）

J_P 车体绕质心转动时的转动惯量（kg*m^2）

θ_P 车体与竖直方向的夹角（rad）

d 轮距（m）

J_δ 车体绕 y 轴转动时的转动惯量（kg*m^2）

δ 小车的偏航角（rad）

v_{LO} 左轮无摩擦力时的线速度大小（rad/s）

v_{RO} 右轮无摩擦力时的线速度大小（rad/s）

3.matlab 计算 K 矩阵 simulink 仿真以及 keil5 的 C 代码

3.1 Matlab 代码和 Simulink 仿真

反馈矩阵很难通过手动计算来得出，因此往往利用 matlab 的函数来进行计算，程序如下:

```
Clc

m = 0.035;

r = 0.0672/2;

i = 0.5*m*r^2;

M = 0.737-2*m;

L = 0.5*0.0903;

J_p = (1/12)*M*(0.0903^2+0.0530^2);

d = 0.1612;

J_delta = (1/12)*M*(0.0930^2+0.0530^2);

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('此系统是可控的！ \n');

    Q = [1000 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 1000 0 0; 0 0 0 0 1000 0; 0 0 0 0 0 0];

    R = [1 0; 0 1];

    K = lqr(A,B,Q,R);

End
```

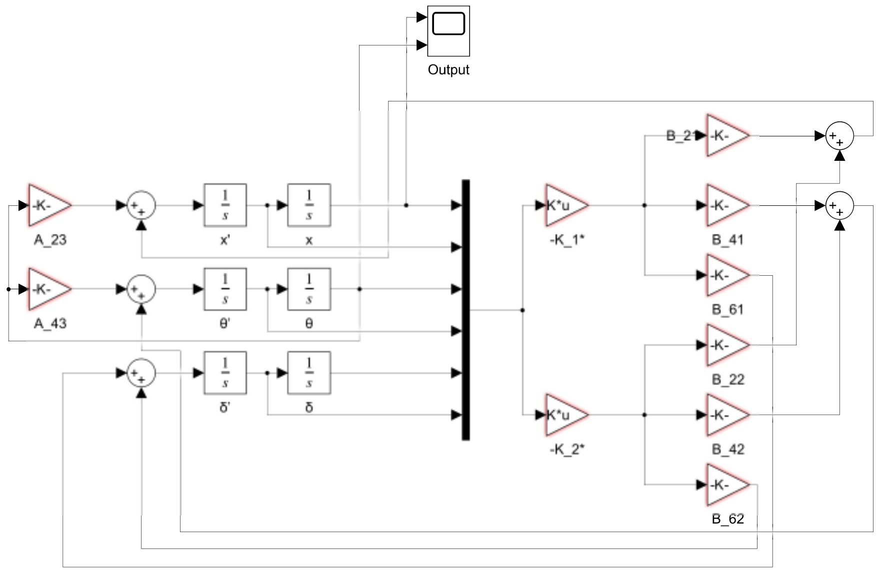
最终可以得出与之相对应的 K 矩阵：

K

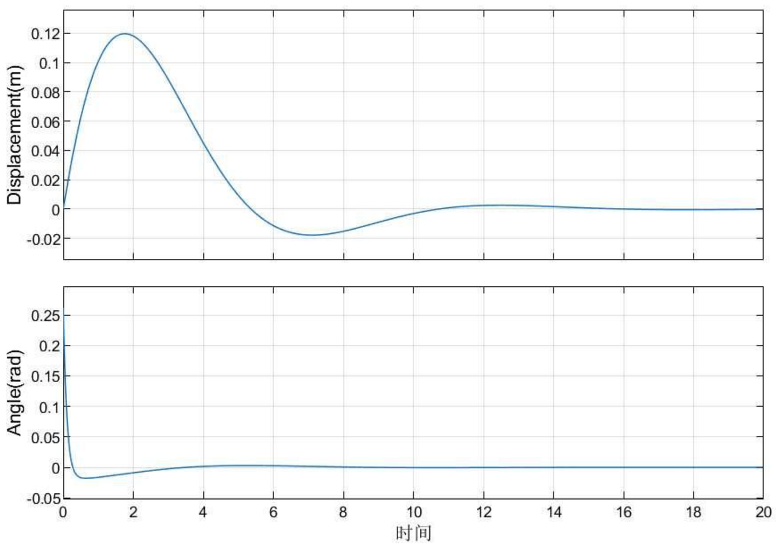
2x6 double

	1	2	3	4	5	6	7
1	-22.3607	-36.1269	-269.7865	-27.7650	22.3607	4.5727	
2	-22.3607	-36.1269	-269.7865	-27.7650	-22.3607	-4.5727	

之后使用 simulink 进行仿真，系统如下



仿真结果:


Displacement(m)
Angle(rad)
时间
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
2
4
6
8
10
12
14
16
18
20

3.2keil5 代码（c 语言）

```c
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));
```

LQR 控制器的效果是使得所有的状态变量都变为 0， LQR 控制器的效果是使得所有的状态变量都变为 0，那么，可以把“x_speed-Target_x_speed”作为新的状态变量，使它变为 0 即等价于使得 x_speed=Target_x_speed。）