第四章作业 LQR 控制

第四章作业是使用 LQR 的方法实现横向的控制，详细的任务说明参考作业 PDF 文档，主要是将 To-Do 的内容编写完成

较难编写的部分主要是相关矩阵的求解，其中矩阵 A 的离散化采用双线性变换得到

$A_t = (\frac{I + T/2*A}{I - T/2*A})$ ，At来源见右边公式

$$z = e^{sT} = \frac{e^{sT/2}}{e^{-sT/2}} \approx \frac{1 + sT/2}{1 - sT/2}$$

控制矩阵的计算主要是参考课件中给出的公式，

Apply LQR in this situation, we have

$$\delta^*(k) = -\boldsymbol{K}\boldsymbol{x}(k)$$

Where $\boldsymbol{K} = (R + \boldsymbol{B}_d^T \boldsymbol{P} \boldsymbol{B}_d)^{-1} \boldsymbol{B}_d^T \boldsymbol{P} \boldsymbol{A}_d$.

Objective cost function to be minimized by the control is

$$J = \sum_{k=0}^{\infty} \boldsymbol{x}^T(k) \boldsymbol{Q} \boldsymbol{x}(k) + \delta(k) R \delta(k)$$

where $\boldsymbol{P}$ satisfies the matrix difference Riccati equation

$$\boldsymbol{P} = \boldsymbol{A}_d^T \boldsymbol{P} \boldsymbol{A}_d - \boldsymbol{A}_d^T \boldsymbol{P} \boldsymbol{B}_d (R + \boldsymbol{B}_d^T \boldsymbol{P} \boldsymbol{B}_d)^{-1} \boldsymbol{B}_d^T \boldsymbol{P} \boldsymbol{A}_d + \boldsymbol{Q}$$

只不过在计算 P 的时候要注意是使用的迭代法，即先给 P 一个初值，然后按照上式迭代计算，达到收敛即停止，如下式

```
Matrix P = Q;
uint num_iteration = 0;
double diff = std::numeric_limits<double>::max();
while (num_iteration++ < max_num_iteration && diff > tolerance) {
  Matrix P_next =
      AT * P * A -
      (AT * P * B) * (R + BT * P * B).inverse() * (BT * P * A) + Q;
  // check the difference between P and P_next
  diff = fabs((P_next - P).maxCoeff());
  P = P_next;
}
```

另外计算角度误差的时候要注意方向，如下式，如果不确定的话可以先仿真，发现一直往相反的方向行驶或者转圈，即知方向反了。

```
lat_con_err->heading_error=-NormalizeAngle(Nearest_Point.heading-theta);
```

前馈控制加和不加在作业的地图中区别不大

编写完成代码之后，使用 ROS 的 catkin_make 编译，编译完成没有错误之后，使用作业指导里面的命令运行，可能要加一行 source devel/setup.bash

注意要在代码的工作空间目录运行

最终运行的效果参考附件视频，车辆会沿着设定的轨迹绕行一圈，并且在距离终点位置附近停止。