深蓝学院

shenlanxueyuan.com

自动驾驶控制与规划
第四章思路提示

主讲人 助教-邱润其

思路提示

深蓝学院
shenlanxueyuan.com

- 车辆线性误差状态方程

State Space Model:

$$\dot{x} = Ax + B_1\delta + B_r r_{des}, \quad x = (e_{cg}, \dot{e}_{cg}, e_\theta, \dot{e}_\theta)$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\dfrac{c_f + c_r}{mv} & \dfrac{c_f + c_r}{mv} & \dfrac{l_r c_r - l_f c_f}{mv} \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{l_r c_r - l_f c_f}{I_z v} & \dfrac{l_r c_r - l_f c_f}{I_z} & -\dfrac{l_f^2 c_f + l_r^2 c_r}{I_z v} \end{bmatrix} \quad B_1 = \begin{bmatrix} 0 \\ \dfrac{c_f}{m} \\ 0 \\ \dfrac{l_f c_f}{I_z} \end{bmatrix} \quad B_2 = \begin{bmatrix} 0 \\ \dfrac{l_r c_r - l_f c_f}{m} - v \\ 0 \\ -\dfrac{l_f^2 c_f + l_r^2 c_r}{I_z v} \end{bmatrix}$$

思路提示

- 车辆线性误差状态方程

State Space Model:

Matrix A

matrix_a_(0, 1) = 1.0;
matrix_a_(1, 2) = (cf_ + cr_) / mass_;
matrix_a_(2, 3) = 1.0;
matrix_a_(3, 2) = (lf_ * cf_ - lr_ * cr_) / iz_;
 	
matrix_a_coeff_(1, 1) = -(cf_ + cr_) / mass_;
matrix_a_coeff_(1, 3) = (lr_ * cr_ - lf_ * cf_) / mass_;
matrix_a_coeff_(3, 1) = (lr_ * cr_ - lf_ * cf_) / iz_;
matrix_a_coeff_(3, 3) = -1.0 * (lf_ * lf_ * cf_ + lr_ * lr_ * cr_) / iz_;

Matrix B
matrix_b_(1, 0) = cf_ / mass_; matrix_b_(3, 0) = lf_ * cf_ / iz_;

思路提示

- 矩阵离散化方法

➢ 前向欧拉法：

$$x(t+dt) = (I + Adt)x(t)$$

➢ 前向欧拉法：

$$x(t+dt) = (I - Adt)^{-1}x(t)$$

➢ 中点欧拉法：

$$x(t+dt) = \left(I - \frac{Adt}{2}\right)^{-1}\left(I - \frac{Adt}{2}\right)x(t)$$

Discretize Matrix B

```
matrix_bd_ = matrix_b_ * ts_;
```

Discretize Matrix A

```
double v;
v = std::max(vehicle_state.velocity,
minimum_speed_protection_);
matrix_a_(1,1) = matrix_a_coeff_(1,1) / v;
matrix_a_(1,3) = matrix_a_coeff_(1,3) / v;
matrix_a_(3,1) = matrix_a_coeff_(3,1) / v;
matrix_a_(3,3) = matrix_a_coeff_(3,3) / v;
Matrix matrix_i =
Matrix::Identity(matrix_a_.cols(),
matrix_a_.cols());
matrix_ad_ = (matrix_i - ts_ * 0.5 *
matrix_a_).inverse() * (matrix_i + ts_ *
0.5 * matrix_a_);
```

思路提示

- 计算横向误差

Discretize Matrix A

```
// 查询距离当前位置距离最近的tagret_point
target_point =
QueryNearestPointByPosition(x, y);

target_point.heading << endl;

// x轴误差 & y轴误差
const double dx = target_point.x - x;
const double dy = target_point.y - y;

const double cos_target_heading =
std::cos(target_point.heading);

const double sin_target_heading =
std::sin(target_point.heading);
```

```
double lateral_error = cos_target_heading * dy -
sin_target_heading * dx;  //横向误差

lat_con_err->lateral_error = lateral_error;

// 计算朝向角并归化到[-M_PI, M_PI]
double heading_error =
NormalizeAngle(target_point.heading – theta);

lat_con_err->heading_error = heading_error;

auto lateral_error_dot = linear_v *
std::sin(heading_error);

lat_con_err->lateral_error_rate = lateral_error_dot;

double ref_heading_rate = target_point.kappa *
target_point.v;

lat_con_err->heading_error_rate = angular_v -
ref_heading_rate;
```

思路提示

- 计算前馈控制量

参考文档：

Automatic Steering Method for Autonomous Automobile Path Tracking

Compute Lateral Error

```
const double kv = lr_ * mass_ / 2 / cf_ / wheelbase_ - lf_ *
mass_ / 2 / cr_ / wheelbase_;

// Calculate the feedforward term of the lateral controller;
then change it
// from rad to %
const double v = localization.velocity;
double steer_angle_feedforwardterm;

steer_angle_feedforwardterm = (wheelbase_ * ref_curvature + kv *
v * v * ref_curvature - matrix_k_(0, 2) * (lr_ * ref_curvature -
lf_ * mass_ * v * v * ref_curvature / 2 / cr_ / wheelbase_));

return steer_angle_feedforwardterm;
```

思路提示

深蓝学院 shenlanxueyuan.com

- Riccati 方程求解

Compute Lateral Error

```
Matrix AT = A.transpose();  // 状态矩阵A的转置
Matrix BT = B.transpose();  // 状态矩阵B的转置
Matrix P = Q;
uint num_iteration = 0;  // 迭代次数
double diff = std::numeric_limits<double>::max();

// 求解Riccati Equation
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;
}
```

```
if (num_iteration >=
max_num_iteration) {
  std::cout << "LQR solver cannot
converge to a solution " << diff <<
std::endl;
} else {
  std::cout << "LQR solver
converged at iteration: " <<
num_iteration << ", max consecutive
result diff.: " << diff <<
std::endl;
}

std::cout << std::endl;

// u = -kx, x为状态向量， u为车轮转角
*ptr_K = (R + BT * P * B).inverse()
* (BT * P * A);
```

参考资料

深蓝学院
shenlanxueyuan.com

- B站up主 忠厚老实的老王 — “自动驾驶控制算法”系列视频
- B站up主 Dr.CAN — “【Advanced 控制理论】8_LQR 控制器”
- Apollo 控制算法部分源码：

  apollo/lat_controller.cc at master · ApolloAuto/apollo (github.com)

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听!

Thanks for Listening