深蓝学院
shenlanxueyuan.com

第四章作业分享

主讲人 阿小关

纲要

- 第一部分：作业内容
- 第二部分：思考与展望

作业内容

- 建立状态空间表达式
- 连续系统的离散化
- 状态量的计算与获取
- 前馈控制量的计算
- 反馈控制量的计算

建立状态空间表达式

Rewrite it as: $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_1\delta + \boldsymbol{B}_2 r_{des}$, where $\boldsymbol{x} = \left(e_{cg}\ \ \dot{e}_{cg}\ \ e_\theta\ \ \dot{e}_\theta\right)^T$.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{(c_f + c_r)}{mv} & \frac{c_f + c_r}{m} & \frac{l_r c_r - l_f c_f}{mv} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{l_r c_r - l_f c_f}{I_z v} & \frac{l_r c_r - l_f c_f}{I_z} & -\frac{l_f^2 c_f + l_r^2 c_r}{I_z v} \end{bmatrix}, \boldsymbol{B}_1 = \begin{bmatrix} 0 \\ \frac{c_f}{m} \\ 0 \\ \frac{l_f c_f}{I_z} \end{bmatrix}, \boldsymbol{B}_2 = \begin{bmatrix} 0 \\ \frac{l_r c_r - l_f c_f}{mv} - v \\ 0 \\ -\frac{l_f^2 c_f + l_r^2 c_r}{I_z v} \end{bmatrix}$$

常数项：与车速无关

非常数项：与车速有关

连续系统的离散化

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y = Cx(t) + Du(t) \end{cases} \xrightarrow{\text{离散化}} \begin{cases} x_{k+1} = A_d x_k + B_d u_k \\ y_k = C_d x_k + D_d u_k \end{cases}$$

Continuous	Forward Euler	Backward Euler	Tustin	Exact
A	$A_d = I + TA$	$(I - TA)^{-1}$	$(I - \frac{AT}{2})^{-1}(I + \frac{AT}{2})$	e^{AT}
B	$B_d = TB$	$T(I - TA)^{-1}B$	$(I - \frac{AT}{2})^{-1}B\sqrt{T}$	$\int_0^T e^{A\tau}Bd\tau$
C	$C_d = C$	$C(I - TA)^{-1}$	$\sqrt{T}C(I - \frac{AT}{2})^{-1}$	C
D	$D_d = D$	$D + C(I - TA)^{-1}BT$	$D + C(I - \frac{AT}{2})^{-1}B\frac{T}{2}$	D
$P(s)$	$s = \frac{1}{T}(z - 1)$	$s = \frac{1}{T}\frac{(z-1)}{z}$	$s = \frac{2}{T}\frac{(z-1)}{(z+1)}$	$(1 - z^{-1})Z[\frac{P(s)}{s}]$


```
// 初始化B矩阵
matrix_b_ = Matrix::Zero(basic_state_size_, 1);
matrix_bd_ = Matrix::Zero(basic_state_size_, 1);
matrix_b_(1, 0) = cf_ / mass_;
matrix_b_(3, 0) = lf_ * cf_ / iz_;
matrix_bd_ = matrix_b_ * ts_;
// B_d = B*T
// 表示离散化采用的是前向欧拉法
// 对应 A_d = I + T*A
```

前向欧拉法

后向欧拉法

中点欧拉法

状态量的计算与获取

```cpp
// TODO 03 计算误差
void LqrController::ComputeLateralErrors(const double x, const double y, const double theta, const double vx, const double vy,
    const double angular_v, const double linear_a, LateralControlErrorPtr &lat_con_err) {
    TrajectoryPoint TargetPoint = QueryNearestPointByPosition(x,y);

    // **确定侧向误差的符号**
    // 将相对位置矢量转换到参考点对应的Frenet坐标系下，根据相对位置矢量在n轴上的投影确定相对位置的符号
    const double dx = -1*(x - TargetPoint.x);
    const double dy = -1*(y - TargetPoint.y);
    const double theta_r = TargetPoint.heading;
    const double kappa_r = TargetPoint.kappa;

    lat_con_err->heading_error = -1*(NormalizeAngle(theta - theta_r));  // 航向误差
    lat_con_err->heading_error_rate = -1 * (angular_v - kappa_r*vx); // 航向误差变化率
    lat_con_err->lateral_error = dy*cos(theta_r) - dx*sin(theta_r);  // 侧向误差
    lat_con_err->lateral_error_rate = -1 * (vy + vx*tan(lat_con_err->heading_error)); //侧向误差变化率

}
```

前馈控制量的计算

```
// TODO 07 前馈控制，计算横向转角的反馈量
double LqrController::ComputeFeedForward(const VehicleState &localization, double ref_curvature) {
    double steer_angle_FF;
    double v = localization.vx;
    steer_angle_FF = ref_curvature*(wheelbase_-lr_*matrix_k_(0,2)-mass_*v*v/wheelbase_*(-lr_/cf_+lf_/cr_-lf_*matrix_k_(0,2)/cr_));

    return -steer_angle_FF;
}
```

由式（3-9）可以看出侧向位置误差 e_1 可以通过合理的选择 δ_{ff} 值而被置为0。然而，如式（3-9）所示，δ_{ff} 不影响稳态偏航误差。不论前馈转向角如何选择，方向角误差总存在不可修正的稳态项。稳态方向角误差为：

$$e_{2\_ss}=\frac{1}{2RC_{\alpha r}(l_f+l_r)}\left[-2C_{\alpha r}l_f l_r-2C_{\alpha r}l_r^2+l_f m v_x^2\right]$$
$$=-\frac{l_r}{R}+\frac{l_f}{2C_{\alpha r}(l_f+l_r)}\times\frac{mv_x^2}{R} \quad (3\text{-}10)$$

如果选择前馈转向角为以下值，稳态侧向位置误差可以为0。

$$\boxed{\delta_{ff}=\frac{mv_x^2}{RL}\left[\frac{l_r}{2C_{\alpha f}}-\frac{l_f}{2C_{\alpha r}}+\frac{l_f}{2C_{\alpha r}}k_3\right]+\frac{L}{R}-\frac{l_r}{R}k_3} \quad (3\text{-}11)$$

车辆动力学及控制[M].机械工业出版社,拉杰什·拉贾马尼,2018

反馈控制量的计算

深蓝学院 shenlanxueyuan.com

```cpp
// TODO 05:求解LQR方程
void LqrController::SolveLQRProblem(const Matrix &A, const Matrix &B, const Matrix &Q, const Matrix &R,
    const double tolerance, const uint max_num_iteration, Matrix *ptr_K) {
    // 防止矩阵的维数出错导致后续的运算失败
    if (A.rows() != A.cols() || B.rows() != A.rows() || Q.rows() != Q.cols() || Q.rows() != A.rows() || R.rows() != R.cols() || R.rows() != B.cols()) {
        std::cout << "LQR solver: one or more matrices have incompatible dimensions." << std::endl;
        return;
    }
    // 离散系统LQR
    Eigen::MatrixXd AT = A.transpose();
    Eigen::MatrixXd BT = B.transpose();
    // 给矩阵P赋初值
    Eigen::MatrixXd P_iter = Q;
    Eigen::MatrixXd P_pre = P_iter;

    // 基于动态规划DP的LQR求解方法
    // 参考:https://stanford.edu/class/ee363/lectures/dlqr.pdf 1-24
    uint num_iteration = 0; // 迭代次数
    double diff = std::numeric_limits<double>::max();   // 相邻两次迭代结果的差值，与设置的迭代精度相比较

    while(num_iteration < max_num_iteration && diff > tolerance){
        P_iter = Q + AT*P_pre*A - AT*P_pre*B*((R+BT*P_pre*B).inverse())*BT*P_pre*A;
        diff = fabs((P_iter - P_pre).maxCoeff());
        P_pre = P_iter;
        num_iteration++;
    }
    // 给指向矩阵K的指针赋值
    *ptr_K = ((R + BT*P_iter*B).inverse())*BT*P_iter*A;
}
```

Summary of LQR solution via DP

1. set $P_N := Q_f$
2. for $t = N, \ldots, 1$,

$$P_{t-1} := Q + A^T P_t A - A^T P_t B (R + B^T P_t B)^{-1} B^T P_t A$$

3. for $t = 0, \ldots, N-1$, define $K_t := -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$
4. for $t = 0, \ldots, N-1$, optimal u is given by $u_t^{\mathrm{lqr}} = K_t x_t$

- optimal u is a linear function of the state (called *linear state feedback*)
- recursion for min cost-to-go runs backward in time

https://stanford.edu/class/ee363/lectures/dlqr.pdf

纲要

- 第一部分：作业内容
- 第二部分：思考与展望

思考与展望

- 前馈控制量

```
// **思考：0.9是否相当于前馈系数，调整前馈控制量的叠加程度，防止高速工况下前馈过大导致系统失稳
// double steer_angle = steer_angle_feedback - 0.9 * steer_angle_feedforward;
double FF_coef = 3.0;
double steer_angle = steer_angle_feedback + FF_coef * steer_angle_feedforward;
```

- 控制算法实时性

Apply LQR in this situation, we have

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

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

Objective cost function to be minimized by the control is

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

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

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

在线问答

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening