

Piecewise Jerk Path Optimizer

该task的代码位于 `/modules/planning/tasks/optimizers/piecewise_jerk_path` ,主要靠调用 `modules/planning/math/piecewise_jerk` 下的类方法来帮助其完成二次规划问题的构造及求解工作。

```
1  /*
2  * @brief:
3  * FEM stands for finite element method.
4  * This class solve an optimization problem:
5  * x
6  * |
7  * |           P(s1, x1)  P(s2, x2)
8  * |           P(s0, x0)           ... P(s(k-1), x(k-1))
9  * |P(start)
10 * |
11 * |_____ s
12 *
13 * we suppose s(k+1) - s(k) == s(k) - s(k-1)
14 *
15 * Given the x, x', x'' at P(start), The goal is to find x0, x1, ... x(k-1)
16 * which makes the line P(start), P0, P(1) ... P(k-1) "smooth".
17 */
18 class PiecewiseJerkPathProblem : public PiecewiseJerkProblem {
19     ...
20 };
```

```
1  /modules/planning/tasks/optimizers/piecewise_jerk_path
2  .
3  |— BUILD
4  |— piecewise_jerk_path_ipopt_solver.cc
5  |— piecewise_jerk_path_ipopt_solver.h
6  |— piecewise_jerk_path_optimizer.cc
7  |— piecewise_jerk_path_optimizer.h
8
9  modules/planning/math/piecewise_jerk
10 .
11 |— BUILD
12 |— piecewise_jerk_path_problem.cc
13 |— piecewise_jerk_path_problem.h #派生类
14 |— piecewise_jerk_problem.cc
15 |— piecewise_jerk_problem.h #基类
16 |— piecewise_jerk_speed_problem.cc
17 |— piecewise_jerk_speed_problem.h #派生类
18
19 #path优化和speed优化的约束条件是一致的，都是在基类中实现的那个约束条件构造函数
```

横向轨迹优化

1 整体流程

1. **Process**
2. **OptimizePath** (task)
3. **set (Points,weight,DP_ref)**(目标点个数及坐标，各项优化目标权重，DP规划出来的path)
4. **Optimize**(优化函数的入口，设置默认迭代次数4000)
5. **FormulateProblem**(用于构造二次优化问题的具体矩阵，也就是将规划问题的求解条件转化为OSQP可求解形式的接口)
 1. **CalculateKernel()**
 2. **CalculateAffineConstraint()**
 3. **CalculateOffset()**
6. **SolverDefaultSettings**(默认配置的参数接口)
7. **osqp setup**(osqp库接口)
8. **osqp solve**(osqp求解接口)
9. **FreeData**(删除数据，释放内存)

```
1  //osqp求解步骤
2  OSQPData* data = FormulateProblem();
3  OSQPSettings* settings = SolverDefaultSettings();
4  settings->max_iter = max_iter;
5  OSQPWorkspace* osqp_work = osqp_setup(data, settings);
6  osqp_solve(osqp_work);
```

2 FormulateProblem()

二次规划问题中P、A是稀疏矩阵

1. CalculateKernel

构造二次项系数矩阵p的压缩矩阵

2. CalculateAffineConstraint

构造A矩阵以及上下边界lower_bounds和upper_bounds的压缩矩阵

3. CalculateOffset

构造一次项系数矩阵q的压缩矩阵

4. csc matrix

将上述转换得到的矩阵压入OSQPData中

```
1 data->n = kernel_dim;
2 data->m = num_affine_constraint;
3 data->P = csc_matrix(kernel_dim, kernel_dim, P_data.size(), CopyData(P_data),
4                   CopyData(P_indices), CopyData(P_indptr));
5 data->q = CopyData(q);
6 data->A = csc_matrix(num_affine_constraint, kernel_dim, A_data.size(),
7                   CopyData(A_data), CopyData(A_indices), CopyData(A_indptr));
```

如何构造一个最优化问题？

以四个点(p1、p2、p3、p4)为例构造最优化问题

二次规划的一般形式

$$\begin{aligned} \text{minimize } & \frac{1}{2} \cdot x^T \cdot P \cdot x + Q \cdot x \\ \text{s.t. } & LB \leq A \cdot x \leq UB \end{aligned} \quad (1)$$

CalculateKernel()构造目标函数矩阵

1. x 矩阵

- x矩阵即为需要优化的变量

$$x^T = [l_1 \ l_2 \ l_3 \ l_4 \ l'_1 \ l'_2 \ l'_3 \ l'_4 \ l''_1 \ l''_2 \ l''_3 \ l''_4] \quad (2)$$

2. p、q 矩阵

通过构造函数来构造 p、q 矩阵，其中代价函数分为三部分

- 曲线平滑 l, l', l'', l'''
- 与参考线的偏差 $(l - l_{ref})$
- 终点位置的软约束

总体代价函数公式

$$\begin{aligned} \text{cost function} = & w_l \cdot \sum_{i=0}^{n-1} l_i^2 + w_{l'} \cdot \sum_{i=0}^{n-1} l_i'^2 + w_{l''} \cdot \sum_{i=0}^{n-1} l_i''^2 + w_{l'''} \cdot \sum_{i=0}^{n-2} \left(\frac{l_{i+1}'' - l_i''}{\Delta s} \right)^2 + \\ & w_{end_l} \cdot (l_{n-1} - l_{endref})^2 + w_{end_{l'}} \cdot (l'_{n-1} - l'_{endref})^2 + w_{end_{l''}} \cdot (l''_{n-1} - l''_{endref})^2 + \\ & w_{ref} \cdot \sum_{i=0}^{n-1} (l_i - l_{ref})^2 + \\ & w_{end_l} \cdot (l_{n-1} - l_{endref})^2 + w_{end_{l'}} \cdot (l'_{n-1} - l'_{endref})^2 + w_{end_{l''}} \cdot (l''_{n-1} - l''_{endref})^2 \end{aligned} \quad (3)$$

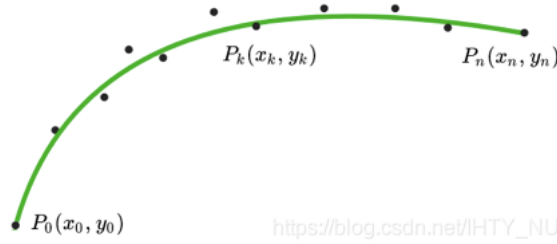
①曲线平滑的 cost

$$w_l \cdot \sum_{i=0}^{n-1} l_i^2 + w_{l'} \cdot \sum_{i=0}^{n-1} l_i'^2 + w_{l''} \cdot \sum_{i=0}^{n-1} l_i''^2 + w_{l'''} \cdot \sum_{i=0}^{n-2} \left(\frac{l_{i+1}'' - l_i''}{\Delta s} \right)^2 \quad (4)$$

- 转化为p矩阵(12*12), 记为 p1

$$p1 = \begin{bmatrix} w_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{l'} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{l'} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_{l'} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{l'} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{l''} + \frac{w_{l'''}}{\Delta s^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2\frac{w_{l'''}}{\Delta s^2} & w_{l''} + 2\frac{w_{l'''}}{\Delta s^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2\frac{w_{l'''}}{\Delta s^2} & w_{l''} + 2\frac{w_{l'''}}{\Delta s^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2\frac{w_{l'''}}{\Delta s^2} & w_{l''} + \frac{w_{l'''}}{\Delta s^2} \end{bmatrix} \quad (5)$$

②与参考线偏差



https://blog.csdn.net/IHTY_NUI

$$w_{ref} \cdot \sum_{i=0}^{n-1} (l_i - l_{ref})^2 \quad (6)$$

- 二次项转化为p矩阵(4*12), 记为 p2

$$p2 = \begin{vmatrix} w_{ref_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{ref_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{ref_3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{ref_4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \quad (7)$$

- 一次项转化为q矩阵(4*1), 记为 q1

注: 去掉上述约束方程的常量项

$$q1 = \begin{vmatrix} -2w_{ref_1} \cdot l_{ref_1} \\ -2w_{ref_2} \cdot l_{ref_2} \\ -2w_{ref_3} \cdot l_{ref_3} \\ -2w_{ref_4} \cdot l_{ref_4} \end{vmatrix} \quad (8)$$

③终点

$$w_{end_l} \cdot (l_{n-1} - l_{endref})^2 + w_{end_{dl}} \cdot (l'_{n-1} - l'_{endref})^2 + w_{end_{ddl}} \cdot (l''_{n-1} - l''_{endref})^2 \quad (9)$$

- 二次项转化为p矩阵(12*12), 记为 p3

$$p3 = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{end_l} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{end_{dl}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{end_{ddl}} \end{vmatrix} \quad (10)$$

- 一次项转化为q矩阵(12*1), 记为 q2

$$q2 = \begin{vmatrix} 0 \\ 0 \\ 0 \\ -2w_{end_l} \cdot endl \\ 0 \\ 0 \\ 0 \\ -2w_{end_{dl}} \cdot enddl \\ 0 \\ 0 \\ 0 \\ -2w_{end_{ddl}} \cdot endddl \end{vmatrix} \quad (11)$$

3.构造优化目标函数

综合 x, p, q 可得到 cost function:

$$\text{minimize } f(l(s)) = x_{(1*12)}^T (p_1^T p_1 + p_2^T p_2 + p_3^T p_3)_{(12*12)} x_{(12*1)} + (q_1^T + q_2^T)_{(1*12)} x_{(12*1)} \quad (12)$$

记:

$$\begin{aligned} P_{(12*12)} &= p_1^T p_1 + p_2^T p_2 + p_3^T p_3 \\ Q_{(1*12)} &= q_1^T + q_2^T \end{aligned} \quad (13)$$

得到最终目标函数的表达式:

$$\text{minimize } f(l(s)) = x^T P x + Q x \quad (14)$$

其中:

$$x_{(1*12)}^{I'} = |l_1 \ l_2 \ l_3 \ l_4 \ l'_1 \ l'_2 \ l'_3 \ l'_4 \ l''_1 \ l''_2 \ l''_3 \ l''_4| \quad (15)$$

$$P_{(12*12)} = \begin{bmatrix} w_l + w_{ref_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_l + w_{ref_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_l + w_{ref_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_l + w_{ref_1} + w_{end_l} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{l'} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{l'} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_{l'} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{l'} + w_{end_{dl}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{l''} + \frac{w_{l''}}{\Delta s^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2\frac{w_{l''}}{\Delta s^2} & w_{l''} + 2\frac{w_{l''}}{\Delta s^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2\frac{w_{l''}}{\Delta s^2} & w_{l''} + 2\frac{w_{l''}}{\Delta s^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2\frac{w_{l''}}{\Delta s^2} & w_{l''} + 2\frac{w_{l''}}{\Delta s^2} \end{bmatrix}$$

$$Q_{(12*1)} = \begin{bmatrix} -2w_{ref_1} \cdot l_{ref_1} \\ -2w_{ref_2} \cdot l_{ref_2} \\ -2w_{ref_3} \cdot l_{ref_3} \\ -2w_{ref_4} \cdot l_{ref_4} - 2w_{end_l} \cdot endl \\ 0 \\ 0 \\ 0 \\ -2w_{end_{dl}} \cdot enddl \\ 0 \\ 0 \\ 0 \\ -2w_{end_{ddl}} \cdot endddl \end{bmatrix} \quad (17)$$

扩展到n个点？

假设约束维度任然是二阶(l、l'、l'')，那么上述的P矩阵为(3n* 3n)，Q矩阵为(3n* * 1)

$$x^T = |l_1 \ \dots \ l_n \ l'_1 \ \dots \ l'_n \ l''_1 \ \dots \ l''_n|_{3n \times 1} \quad (18)$$

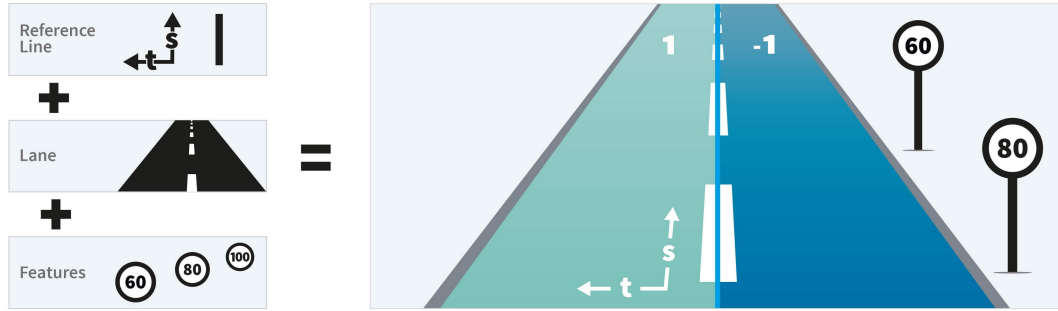
$$P = \begin{bmatrix} \begin{bmatrix} w_l + w_{ref_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & w_l + w_{ref_1} + w_{end_l} \end{bmatrix}_{n \times n} & \begin{bmatrix} w_{l'} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & w_{l'} + w_{end_{dl}} \end{bmatrix}_{n \times n} & \begin{bmatrix} w_{l''} + \frac{w_{l''}}{\Delta s^2} & 0 & \dots & \dots \\ -2\frac{w_{l''}}{\Delta s^2} & w_{l''} + \frac{w_{l''}}{\Delta s^2} & 0 & \dots \\ 0 & -2\frac{w_{l''}}{\Delta s^2} & \ddots & \dots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & -2\frac{w_{l''}}{\Delta s^2} & w_{l''} + \frac{w_{l''}}{\Delta s^2} + w_{end_{ddl}} \end{bmatrix}_{n \times n} \end{bmatrix}$$

$$Q = \begin{bmatrix} \begin{bmatrix} -2w_{ref_1} \cdot l_{ref_1} \\ \vdots \\ -2w_{ref_4} \cdot l_{ref_4} - 2w_{end_l} \cdot endl \end{bmatrix}_{n \times 1} \\ \begin{bmatrix} 0 \\ \vdots \\ -2w_{end_{dl}} \cdot enddl \end{bmatrix}_{n \times 1} \\ \begin{bmatrix} 0 \\ \vdots \\ -2w_{end_{ddl}} \cdot endddl \end{bmatrix}_{n \times 1} \end{bmatrix}_{3n \times 1} \quad (20)$$

CalculateAffineConstraint()构造约束矩阵

1.对l的约束

车辆行驶位置，即对道路边界的约束



```
1 DEFINE_double(longitudinal_jerk_lower_bound, -4.0,
2               "The lower bound of longitudinal jerk.");
3 DEFINE_double(longitudinal_jerk_upper_bound, 2.0,
4               "The upper bound of longitudinal jerk.");
```

构造矩阵(4*12):

$$\begin{bmatrix} lb_{s1} \\ lb_{s2} \\ lb_{s3} \\ lb_{s4} \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x \leq \begin{bmatrix} ub_{s1} \\ ub_{s2} \\ ub_{s3} \\ ub_{s4} \end{bmatrix} \quad (21)$$

2.对l'的约束

轨迹的一阶导为heading，可以近似理解为横向运动的“速度”，希望不要横向走的太快

```
1 DEFINE_double(lateral_derivative_bound_default, 2.0,
2               "the default value for lateral derivative bound.");
```

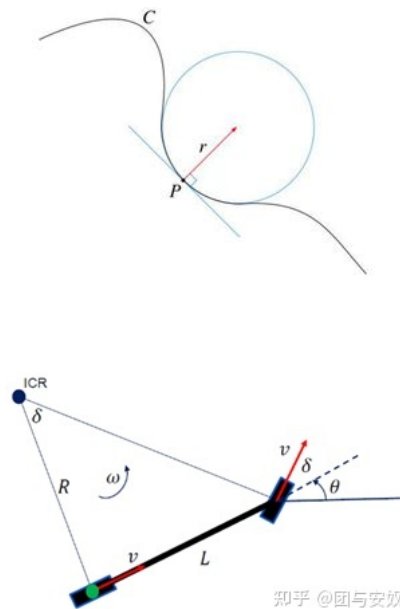
构造矩阵(4*12):

$$\begin{bmatrix} -2.0 \\ -2.0 \\ -2.0 \\ -2.0 \end{bmatrix} \leq \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} x \leq \begin{bmatrix} 2.0 \\ 2.0 \\ 2.0 \\ 2.0 \end{bmatrix} \quad (22)$$

3.对l''的约束

轨迹的二阶导可以近似理解为横向运动的“加速度”，希望方向盘不要打得太猛

因为车辆存在最小的转弯半径,即存在最大行驶曲率，所以要对车辆运动学进行限制。



$$K_{max} = \frac{1}{R} = \frac{\tan(\delta_{max})}{L} = \frac{\tan(\frac{maxSteerAngle}{steerRadio})}{L} \quad (23)$$

其中： R 为车辆的最大转弯半径， L 为车辆轴距
 δ_{max} 为前轮最大转角， $steerRadio$ 为车辆转向传动比

问题(bug)

代码中的意思是直接把 κ_r 等价于当前轨迹的曲率 k ，而实际的 k 值计算如下文所示

- 在百度发表的论文中对车辆曲率的约束如下: [Optimal Vehicle Path Planning Using Quadratic Optimization for Baidu Apollo Open Platform](#)

The most important factor for kinematic feasibility is the curvature of the path. According to the frame conversion equations in [14], the curvature of one point in path is defined as the following equation:

$$\kappa = \frac{\left(\frac{((l'' + (\kappa_r l + \kappa_r l') \tan \Delta \theta) \cos^2 \Delta \theta) + \kappa_r}{1 - \kappa_r l} \right) \cos \Delta \theta}{1 - \kappa_r l}$$

where κ_r and κ_r' are the curvature and its change rate of the corresponding point p_r on the driving guide line, $\Delta \theta$ is the angle difference between vehicle heading direction and the tangent direction of point r .

To simply the complex relation, we make the following assumptions:

- 1) The vehicle is nearly parallel to the driving guide line, i.e., the vehicle's heading angle is assume to be the same as the direction of the guide line at the corresponding point, thus $\Delta \theta = 0$.
- 2) The lateral "acceleration" l'' is numerically small (in the order of 10^{-2}) and is assumed to be 0.

Based on these, κ is approximated as follows:

$$\kappa \approx \frac{\kappa_r}{1 - \kappa_r * l}$$

Given the kinematic model of the vehicle (see Fig.2) and vehicle's maximal steer angle α_{max} , the maximal curvature for the vehicle can be computed:

$$\kappa_{max} = \frac{\tan(\alpha_{max})}{L}$$

Thus, we add the linear constraint for l as follows to the optimization procedure for kinematic feasibility:

$$\tan(\alpha_{max}) * \kappa_r * l - \tan(\alpha_{max}) + |\kappa_r| * L \leq 0$$

κ_r 和 κ_r' 是参考线在 p_r 处的曲率和曲率变化率, $\Delta \theta$ 是车辆和参考线点 p_r 处切线方向的角度差。

简化: 假设车辆几乎在沿着道路方向行驶, 因此 $\Delta \theta = 0$; "横向加速度" l'' 是很小的, 数量级在 10^{-2} , 因此 $l'' = 0$

(24)

- 百度公开课讲解的曲率约束内容如下

Piecewise Jerk Path Optimization

Constraints


- 对车辆运动学限制的近似
- Assumption:
 $\Delta \theta \approx 0$, 车辆朝向与reference line平行
 $l'' \approx 0$, 横向二阶导几乎为零

$$\kappa = \frac{\left(\frac{((l'' + (\kappa_r l + \kappa_r l') \tan \Delta \theta) \cos^2 \Delta \theta) + \kappa_r}{1 - \kappa_r l} \right) \cos \Delta \theta}{1 - \kappa_r l}$$

κ_r is the curvature along reference line

Curvature constraints by rear centered kinematic model

$$\kappa \approx \frac{\kappa_r}{1 - \kappa_r * l} \quad \kappa < \kappa_{max}$$

$$\kappa_{max} = \frac{\tan(\alpha_{max})}{L}$$


$$\tan(\alpha_{max}) * \kappa_r * l - \tan(\alpha_{max}) + |\kappa_r| * L \leq 0$$

- 但是在代码中A矩阵的赋值根据代码直接设为1, 也就是说 κ_r 直接等于了当前车辆的行驶曲率 k
 - 给A矩阵赋值

```
1  int constraint_index = 0;
2  // set x, x', x'' bounds
3  for (int i = 0; i < num_of_variables; ++i) {
4      if (i < n) {
5          variables[i].emplace_back(constraint_index, 1.0);
6          lower_bounds->at(constraint_index) =
7              x_bounds_[i].first * scale_factor_[0];
8          upper_bounds->at(constraint_index) =
9              x_bounds_[i].second * scale_factor_[0];
10     } else if (i < 2 * n) {
11         variables[i].emplace_back(constraint_index, 1.0);
12     }
13     lower_bounds->at(constraint_index) =
```

```

14     dx_bounds_[i - n].first * scale_factor_[1];
15     upper_bounds->at(constraint_index) =
16     dx_bounds_[i - n].second * scale_factor_[1];
17 } else {
18     variables[i].emplace_back(constraint_index, 1.0);
19     lower_bounds->at(constraint_index) =
20     ddx_bounds_[i - 2 * n].first * scale_factor_[2];
21     upper_bounds->at(constraint_index) =
22     ddx_bounds_[i - 2 * n].second * scale_factor_[2];
23 }
24 ++constraint_index;
25 }
26
27 //给A矩阵赋值
28 int ind_p = 0;
29 for (int i = 0; i < num_of_variables; ++i) {
30     A_indptr->push_back(ind_p);
31     for (const auto& variable_nz : variables[i]) {
32         // coefficient
33         A_data->push_back(variable_nz.second);
34
35         // constraint index
36         A_indices->push_back(variable_nz.first);
37         ++ind_p;
38     }

```

- 上下边界赋值

```

1 //车辆运动学约束，由车轮最大转角推导行驶过程中的最大曲率
2 const double lat_acc_bound =
3     std::tan(veh_param.max_steer_angle() / veh_param.steer_ratio()) /
4     veh_param.wheel_base();
5
6 //要考虑道路的曲率，所以要减去道路的kappa值
7 double kappa = reference_line.GetNearestReferencePoint(s).kappa();
8     ddl_bounds.emplace_back(-lat_acc_bound - kappa, lat_acc_bound - kappa);

```

根据代码构造出矩阵如下(4*12):

$$\begin{bmatrix} -lat_acc_bound - kappa_{s1} \\ -lat_acc_bound - kappa_{s2} \\ -lat_acc_bound - kappa_{s3} \\ -lat_acc_bound - kappa_{s4} \end{bmatrix} \leq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x \leq \begin{bmatrix} lat_acc_bound - kappa_{s1} \\ lat_acc_bound - kappa_{s2} \\ lat_acc_bound - kappa_{s3} \\ lat_acc_bound - kappa_{s4} \end{bmatrix} \quad (25)$$

上述矩阵相当于直接把 l'' 等价于曲率,而实际k的约束如论文中所示, 两个地方约束计算并不相同。

$$\tan(\delta_{max}) \times k_r \times l - \tan(\delta_{max}) + k_r \times L \leq 0 \quad (26)$$

4.对 l''' 的约束

由差分求导可得到轨迹的三阶导数,可以理解为人打方向盘的加速度,此时是对 jerk 的约束, `delta_s_ = 1.0;`

$$l''' = \frac{l''_{i+1} - l''_i}{\Delta s} \quad (27)$$

横摆角速度:

$$yaw_rate = \frac{(w_1 - w_2) \cdot R_r}{A} \quad (28)$$

```

1 double PiecewiseJerkPathOptimizer::EstimateJerkBoundary(
2     const double vehicle_speed, const double axis_distance,
3     const double max_yaw_rate) const {
4     return max_yaw_rate / axis_distance / vehicle_speed;
5 }

```

$$\begin{bmatrix} -jerk_1 * \Delta s \\ -jerk_2 * \Delta s \\ -jerk_3 * \Delta s \end{bmatrix} \leq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} x \leq \begin{bmatrix} jerk_1 * \Delta s \\ jerk_2 * \Delta s \\ jerk_3 * \Delta s \end{bmatrix} \quad (29)$$

5.对起点p1的约束

起点必须在初始点的位置

$$\begin{bmatrix} ego_l \\ ego_{dl} \\ ego_{ddl} \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} x \leq \begin{bmatrix} ego_l \\ ego_{dl} \\ ego_{ddl} \end{bmatrix} \quad (30)$$

6.路径连续性约束

```

1 // x(i+1)' - x(i)' - 0.5 * delta_s * x(i)'' - 0.5 * delta_s * x(i+1)'' = 0
2 for (int i = 0; i + 1 < n; ++i) {
3     ...
4     lower_bounds->at(constraint_index) = 0.0;
5     upper_bounds->at(constraint_index) = 0.0;
6     ++constraint_index;

```

```

7 }
8
9 // x(i+1) - x(i) - delta_s * x(i)'
10 // - 1/3 * delta_s^2 * x(i)'' - 1/6 * delta_s^2 * x(i+1)''
11 auto delta_s_sq_ = delta_s_ * delta_s_;
12 for (int i = 0; i + 1 < n; ++i) {
13     ..._{(12*1)}
14
15     lower_bounds->at(constraint_index) = 0.0;
16     upper_bounds->at(constraint_index) = 0.0;
17     ++constraint_index;
18 }

```

对上述代码中的公式推导：

将零阶状态用一二阶状态进行线性表示，使其更为合理地表示各界状态的关联关系，确保路径的连续性。轨迹的三阶导数会随着二阶导数的变化而变化，但两点的三阶导保持相等

泰勒公式：

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots + \frac{f^n(x_0)}{n!}(x - x_0)^n + R_n(x) \quad (1-1)$$

其中：

$$R_n(x) = o((x - x_0)^n) \quad (1-2)$$

对于状态转移来说：

i 位置所对应的状态： (S_i, X_i) , $d(S_i, X_i)$, $dd(S_i, X_i)$

i+1 位置所对应的状态： (S_{i+1}, X_{i+1}) , $d(S_{i+1}, X_{i+1})$, $dd(S_{i+1}, X_{i+1})$

对于(1-1)忽略二阶无穷小可得：

$$f(S) = f(S_0) + f'(S_0)(S - S_0) + \frac{f''(S_0)}{2!}(S - S_0)^2 \quad (1-3)$$

忽略三阶无穷小可得：

$$f(S) = f(S_0) + f'(S_0)(S - S_0) + \frac{f''(S_0)}{2!}(S - S_0)^2 + \frac{f'''(S_0)}{3!}(S - S_0)^3 \quad (1-4)$$

对于(1-3)：

令 $X_{i+1} = f(S)$, $S_{i+1} = S$, $f(S_0) = X_i$, $S_i = S_0$ 则：

$$X_{i+1} - X_i = \dot{X}_i \cdot \Delta S + \frac{1}{2} \ddot{X}_i \cdot \Delta S^2 \quad (1-5)$$

对于(1-4)

令 $X_{i+1} = f(S_0)$, $S_{i+1} = S_0$, $f(S) = X_i$, $S_i = S$ 则：

$$X_i - X_{i+1} = \dot{X}_{i+1} \cdot (-\Delta S) + \frac{1}{2} \ddot{X}_{i+1} \cdot \Delta S^2$$

(1-6)

(1-5) + (1-6) 得：

$$0 = (\dot{X}_i - \dot{X}_{i+1}) + \frac{1}{2}(\ddot{X}_i \cdot \Delta S^2) + \frac{1}{2}(\ddot{X}_{i+1} \cdot \Delta S^2)$$

由于 ΔS 非零，左右同时除以 ΔS ，并将所有项移动到等号左边可得：

$$(\dot{X}_i - \dot{X}_{i+1}) + \frac{1}{2}(\ddot{X}_i \cdot \Delta S^2) + \frac{1}{2}(\ddot{X}_{i+1} \cdot \Delta S^2) \quad (1-6)$$

同样对于(1-4)

令 $X_{i+1} = f(S)$, $S_{i+1} = S$, $f(S_0) = X_i$, $S_i = S_0$ 则：

$$X_{i+1} - X_i = \dot{X}_i \cdot \Delta S + \frac{1}{2} \ddot{X}_i \cdot \Delta S^2 + \frac{1}{6} \ddot{X}_i \Delta S^3 \quad (1-7)$$

因为三阶状态可由二阶状态表示：

$$\ddot{X}_i = \frac{\ddot{X}_{i+1} - \ddot{X}_i}{\Delta S}$$

代入上式(1-7)可得：

$$X_{i+1} - X_i = \dot{X}_i \cdot \Delta S + \frac{1}{2} \ddot{X}_i \cdot \Delta S^2 + \frac{1}{6} \frac{\ddot{X}_{i+1} - \ddot{X}_i}{\Delta S} \Delta S^3$$

整理之后得：

$$X_{i+1} - X_i = \dot{X}_i \cdot \Delta S + \frac{1}{2} \ddot{X}_i \cdot \Delta S^2 + \frac{1}{6}(\ddot{X}_{i+1} - \ddot{X}_i) \Delta S^2$$

$$X_{i+1} - X_i = \dot{X}_i \cdot \Delta S + \frac{1}{3} \ddot{X}_i \cdot \Delta S^2 + \frac{1}{6} \ddot{X}_{i+1} \Delta S^2$$

知乎 @团与安奴

知乎 @团与安奴

通过上述(1-6)(1-8)可得到:

$$l'_{i+1} - l'_i - \frac{1}{2}\Delta s * l''_i - \frac{1}{2}\Delta s * l''_{i+1} = 0 \quad (31)$$

$$l_{i+1} - l_i - \Delta s \cdot l'_i - \frac{1}{3}\Delta s^2 \cdot l''_i - \frac{1}{6}\Delta s^2 \cdot l''_{i+1} = 0$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -\frac{1}{2} \cdot \Delta s & -\frac{1}{2} \cdot \Delta s & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -\frac{1}{2} \cdot \Delta s & -\frac{1}{2} \cdot \Delta s & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -\frac{1}{2} \cdot \Delta s & -\frac{1}{2} \cdot \Delta s \end{bmatrix} x \leq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (32)$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} -1 & 1 & 0 & 0 & -\Delta s & 0 & 0 & 0 & -\frac{1}{3}\Delta s^2 & -\frac{1}{6}\Delta s^2 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & -\Delta s & 0 & 0 & 0 & -\frac{1}{3}\Delta s^2 & -\frac{1}{6}\Delta s^2 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & -\Delta s & 0 & 0 & 0 & -\frac{1}{3}\Delta s^2 & -\frac{1}{6}\Delta s^2 \end{bmatrix} x \leq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (33)$$

7.构造约束条件

$$x^T = [l_1 \ l_2 \ l_3 \ l_4 \ l'_1 \ l'_2 \ l'_3 \ l'_4 \ l''_1 \ l''_2 \ l''_3 \ l''_4]_{1 \times 12} \quad (34)$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -\frac{1}{2} \cdot \Delta s & -\frac{1}{2} \cdot \Delta s & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -\frac{1}{2} \cdot \Delta s & -\frac{1}{2} \cdot \Delta s & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -\frac{1}{2} \cdot \Delta s & -\frac{1}{2} \cdot \Delta s \\ -1 & 1 & 0 & 0 & -\Delta s & 0 & 0 & 0 & -\frac{1}{3}\Delta s^2 & -\frac{1}{6}\Delta s^2 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & -\Delta s & 0 & 0 & 0 & -\frac{1}{3}\Delta s^2 & -\frac{1}{6}\Delta s^2 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & -\Delta s & 0 & 0 & 0 & -\frac{1}{3}\Delta s^2 & -\frac{1}{6}\Delta s^2 \end{bmatrix}_{24 \times 12} \quad (35)$$

$$LB = \begin{bmatrix} lb_{s1} \\ lb_{s2} \\ lb_{s3} \\ lb_{s4} \\ -2.0 \\ -2.0 \\ -2.0 \\ -2.0 \\ -lat_acc_bound - kappa_{s1} \\ -lat_acc_bound - kappa_{s2} \\ -lat_acc_bound - kappa_{s3} \\ -lat_acc_bound - kappa_{s4} \\ -jerk_1 * \Delta s \\ -jerk_2 * \Delta s \\ -jerk_3 * \Delta s \\ ego_l \\ ego_{dl} \\ ego_{ddl} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}_{24 \times 1} = \begin{bmatrix} Lbsi_{(4 \times 1)} \\ Heading1_{(4 \times 1)} \\ Kappa1_{(4 \times 1)} \\ Jerk1_{(3 \times 1)} \\ Ego1_{(3 \times 1)} \\ Continuous1_{(6 \times 1)} \end{bmatrix}; UB = \begin{bmatrix} ub_{s1} \\ ub_{s2} \\ ub_{s3} \\ ub_{s4} \\ 2.0 \\ 2.0 \\ 2.0 \\ 2.0 \\ lat_acc_bound - kappa_{s1} \\ lat_acc_bound - kappa_{s2} \\ lat_acc_bound - kappa_{s3} \\ lat_acc_bound - kappa_{s4} \\ jerk_1 * \Delta s \\ jerk_2 * \Delta s \\ jerk_3 * \Delta s \\ ego_l \\ ego_{dl} \\ ego_{ddl} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}_{24 \times 1} = \begin{bmatrix} Ubsi_{(4 \times 1)} \\ Heading2_{(4 \times 1)} \\ Kappa2_{(4 \times 1)} \\ Jerk2_{(3 \times 1)} \\ Ego2_{(3 \times 1)} \\ Continuous2_{(6 \times 1)} \end{bmatrix} \quad (36)$$

综上得到约束条件:

$$LB_{(12 \times 1)} \leq A_{(24 \times 12)} x_{(12 \times 1)} \leq UB_{(12 \times 1)} \quad (37)$$

扩展到n个点?

$$A = \begin{bmatrix} 1 & \dots & \dots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & 1 \\ & & & & 1 & \dots & \dots & 0 \\ & & & & \vdots & \ddots & & \vdots \\ & & & & \vdots & & \ddots & \vdots \\ & & & 0 & \dots & \dots & 1 \\ & & & -1 & 1 & \dots & 0 \\ & & & \vdots & \ddots & \ddots & \vdots \\ & & & \vdots & & \ddots & \ddots \\ & & & 0 & \dots & -1 & 1 \end{bmatrix} \quad (38)$$

$$LB = \begin{bmatrix} Lbsi_{(n*1)} \\ Heading1_{(n*1)} \\ Kappa1_{(n*1)} \\ Jerk1_{((n-1)*1)} \\ Ego1_{(3*1)} \\ Continuous1_{((2n-2)*1)} \end{bmatrix}_{6n \times 1} \quad UB = \begin{bmatrix} Ubsi_{(n*1)} \\ Heading2_{(n*1)} \\ Kappa2_{(n*1)} \\ Jerk2_{((n-1)*1)} \\ Ego2_{(3*1)} \\ Continuous2_{((2n-2)*1)} \end{bmatrix}_{6n \times 1} \quad (39)$$

总结

假设有n个点，优化维度为三维(l、l'、l'')，通过构造P,Q,A, LB, UB 矩阵方程，将此问题转化为二次规划问题

$$\begin{aligned} \text{minimize } f(l(s)) &= x^T P x + Q x \\ \text{s.t. } LB &\leq A x \leq UB \end{aligned} \quad (40)$$

其中:

$$\begin{aligned} x^T &= [l_1 \dots l_n \ l'_1 \dots l'_n \ l''_1 \dots l''_n]_{3n \times 1} \\ P &= [\dots]_{3n \times 3n} \\ Q &= [\dots]_{3n \times 1} \\ A &= [\dots]_{6n \times 3n} \\ LB &= [\dots]_{6n \times 1} \\ UB &= [\dots]_{6n \times 1} \end{aligned} \quad (41)$$

Picewise Jerk Speed Optimizer

纵向速度轨迹优化

SL规划保证车辆的横向偏移足够平滑，ST规划保证车辆的前进方向速度变化足够平滑。

1. x 矩阵

x矩阵即为需要优化的变量

$$x^T = |s_1 \dots s_n \ s'_1 \dots s'_n \ s''_1 \dots s''_n| \tag{42}$$

2. p、q 矩阵

跟path optimizer区别在于p矩阵，speed多了对参考线偏差的一阶偏差约束

① 曲线平滑

$$w_s \cdot \sum_{i=0}^{n-1} s_i^2 + w_{ds} \cdot \sum_{i=0}^{n-1} (s'_i)^2 + w_{dds} \cdot \sum_{i=0}^{n-2} (\frac{s''_{i+1} - s''_i}{\Delta s^2})^2 \tag{43}$$

② 与参考线偏差

$$w_{xref} \cdot \sum_{i=0}^{n-1} (s_i - s_{ref})^2 + w_{dxref} \cdot \sum_{i=0}^{n-1} (s'_i - s'_{ref})^2 \tag{44}$$

③ 终点

$$w_{endi} \cdot (s_{n-1} - s_{endref})^2 + w_{enddi} \cdot (s'_{n-1} - s'_{endref})^2 + w_{endddi} \cdot (s''_{n-1} - s''_{endref})^2 \tag{45}$$

3.约束条件

约束分为六个部分(6n*1)

对变量的约束(3n): $LowerBounds < s, s', s'', s''' < upperBounds$

对Jerk的约束(n-1): $LowerBounds < s''' < upperBounds$ 对起点的约束(3): $ego_1 \leq s_1, s'_1, s''_1 \leq ego_1$

连续性约束(2n-2): $s_{i+1} - s_i - \Delta s \cdot s'_i - \frac{1}{3} \Delta s^2 \cdot s''_i - \frac{1}{6} \Delta s^3 \cdot s'''_i = 0$

$s'_{i+1} - s'_i - \frac{1}{2} \Delta s \cdot s''_i - \frac{1}{2} \Delta s \cdot s'''_i = 0$

$s''_{i+1} - s''_i - \Delta s \cdot s'''_i = 0$

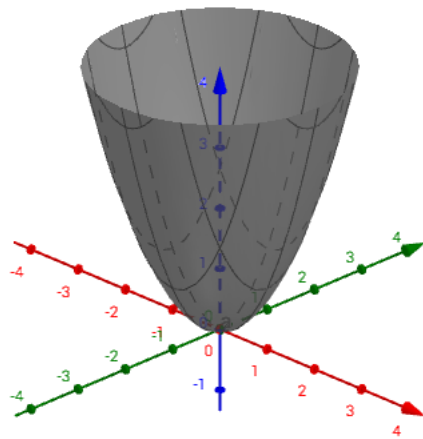
(46)

思考

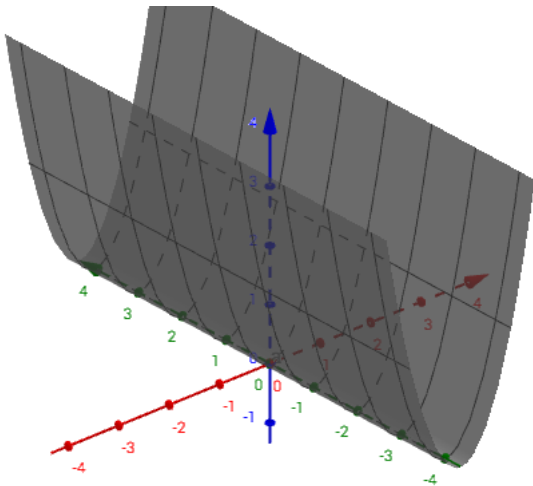
1 为什么需要正定矩阵？

- 如果P是半正定矩阵，那么f(x)是一个凸函数。相应的二次规划为凸二次规划问题；此时若约束条件定义的可行域不为空，且目标函数在此可行域有下界，则问题有全局最小值。
- 如果P是正定矩阵，则该问题有唯一的全局最小值。
- 若P为非正定矩阵，则目标函数是有多个平稳点和局部极小点的NP难问题。
- 如果P=0，二次规划问题就变成线性规划问题。

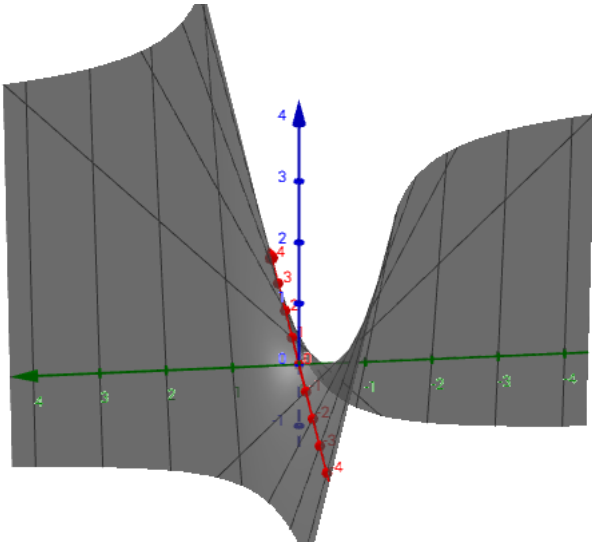
正定是对二次函数有效的一个定义，对方程无效。



$f(x) > 0, x \neq 0$, 则 x 为正定二次型, P 为正定矩阵 (47)



$f(x) \geq 0, x \neq 0$, 则 f 为半正定二次型, P 为半正定矩阵 (48)



不定 (49)

2.压缩矩阵有哪几种方法？

参考[英伟达](#)的介绍
