spiral_reference_line_smoother

简介

1. 掌握**基于优化方法利用分段多项式螺旋线平滑参考线的建模过程**
2. 了解**基于样条曲线**和**基于螺旋曲线**平滑曲线之间的优劣
3. 了解Apollo中三种平滑器的优劣

螺旋曲线 vs 样条线

样条线

五次样条(quintic splines)是最常用的一种，它描述的是**车辆x和y位置的五次多项式函数**，对于车辆一条轨迹，五次样条有12个参数，其中x方程为6个，y方程为6个，变量t可以任意设置，方程表示为：

$$\begin{aligned} x(t) &= at^5 + bt^4 + ct^3 + dt^2 + et + f \\ y(t) &= a_1t^5 + b_1t^4 + c_1t^3 + d_1t^2 + e_1t + f_1 \end{aligned} \tag{1}$$

优势

对于给定$(x, y, \theta, kappa)$的边界条件，都存在一组满足关系的样条系数，其实现起来比使用迭代优化方法更加便捷.

劣势

很难实现把曲率限制在一定范围内，因为曲率是弧长的函数，其形式并不是多项式，此时**很可能引入尖点**，甚至会导致曲率的不连续。这使得很难在五次样条的整个范围内大致满足曲率约束。

$$k = \frac{d\theta}{ds} = \frac{x'y'' - y'x''}{(x'^2 + y')^{\frac{3}{2}}} \tag{2}$$

螺旋曲线

多项式螺旋(polynomial spiral)，由相对于弧长的多项式函数给出，会沿其弧长的每个点的曲率提供了一个闭合形式方程

$$\begin{aligned} \theta(s) &= as^5 + bs^4 + cs^3 + ds^2 + es + f \\ k(s) &= 5as^4 + 4bs^3 + 3cs^2 + 2ds + e \end{aligned} \tag{3}$$

优势

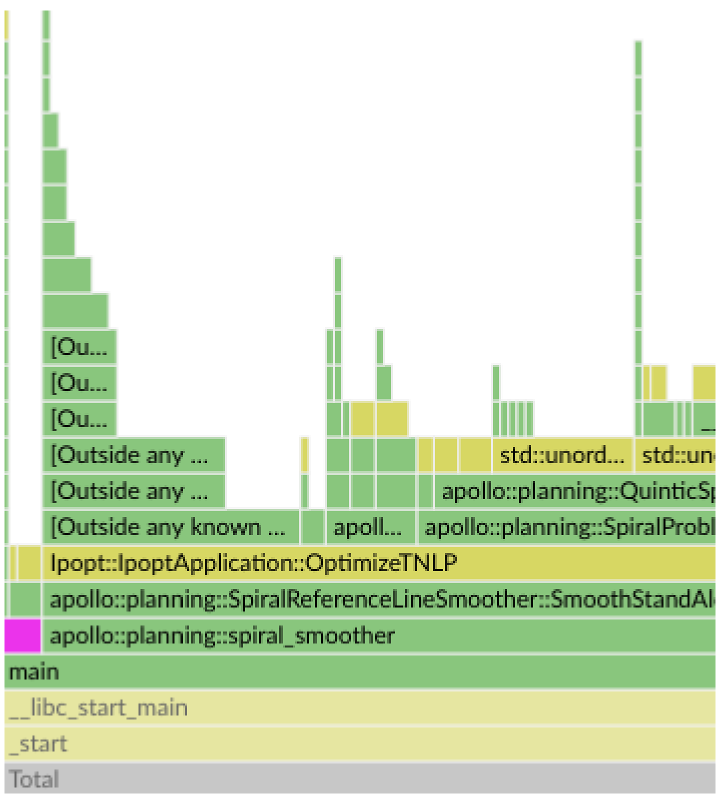
其结构**非常容易满足路径规划问题中需要的曲率约束**。由于螺旋线是曲率的多项式函数，因此曲率值不会像在五次样条曲线中那样迅速变化。通过限制螺旋线和螺旋线中仅几个点的曲率，就很可能满足了整个曲线上的曲率约束

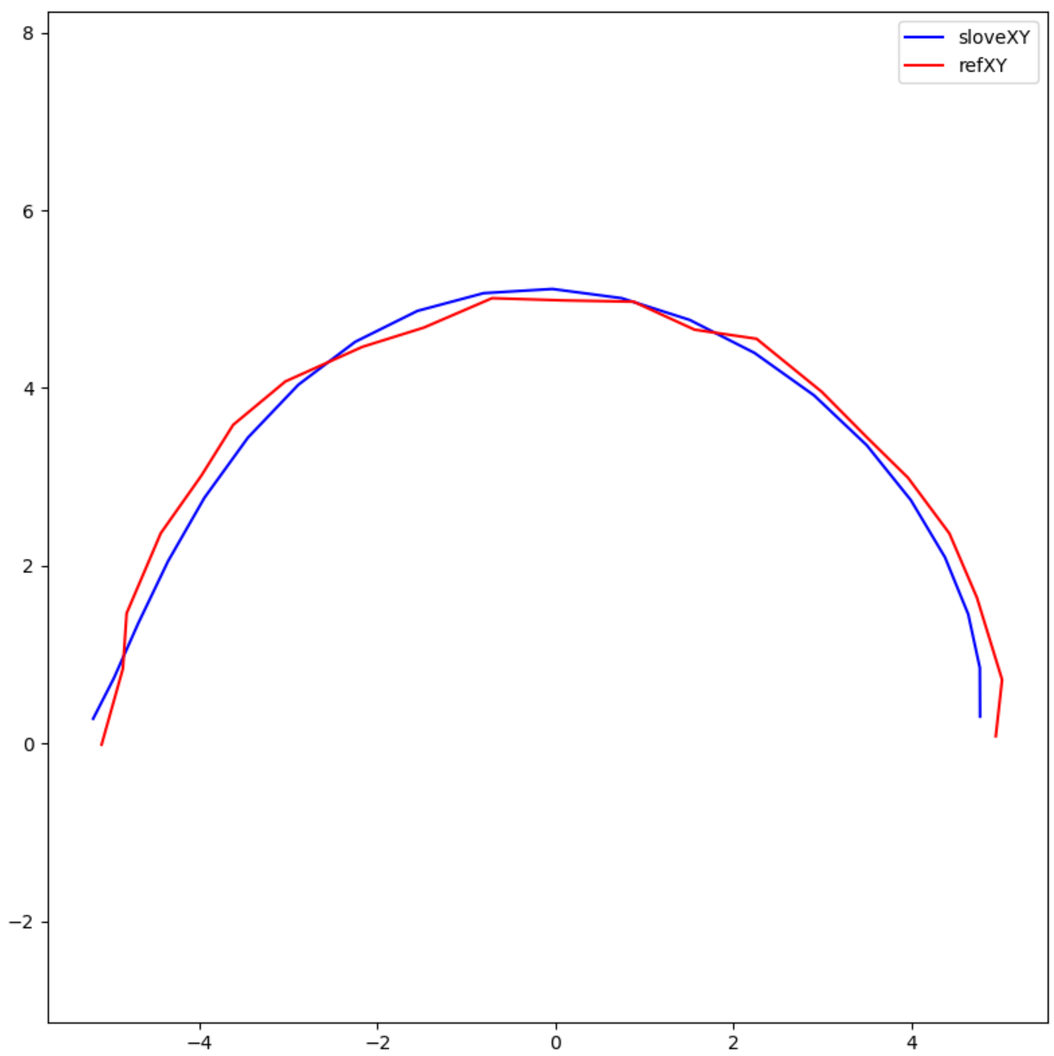
劣势

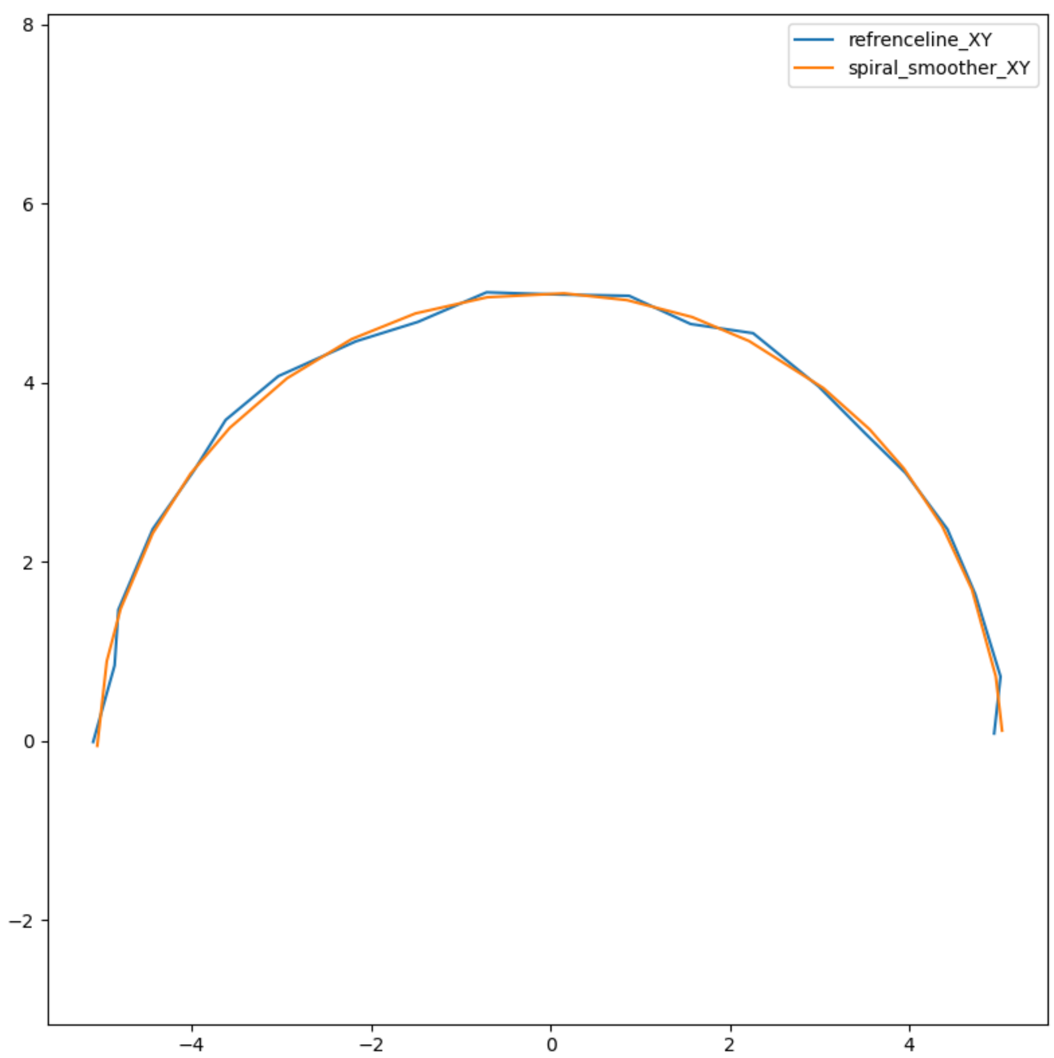
螺旋的位置不能进行闭式求解，这与五次样条中的情况不同。因此，必须进行迭代优化才能生成一个螺旋来满足边界条件。从下面的方程可以看出，位置方程得出的菲涅耳积分没有封闭形式的解。因此，需要使用数值逼近来计算螺旋曲线的端点

$$\begin{aligned} x_{i+1} &= x_i + \int_0^{\Delta s_i} \cos(\theta(s))ds \\ y_{i+1} &= y_i + \int_0^{\Delta s_i} \sin(\theta(s))ds \end{aligned} \tag{4}$$

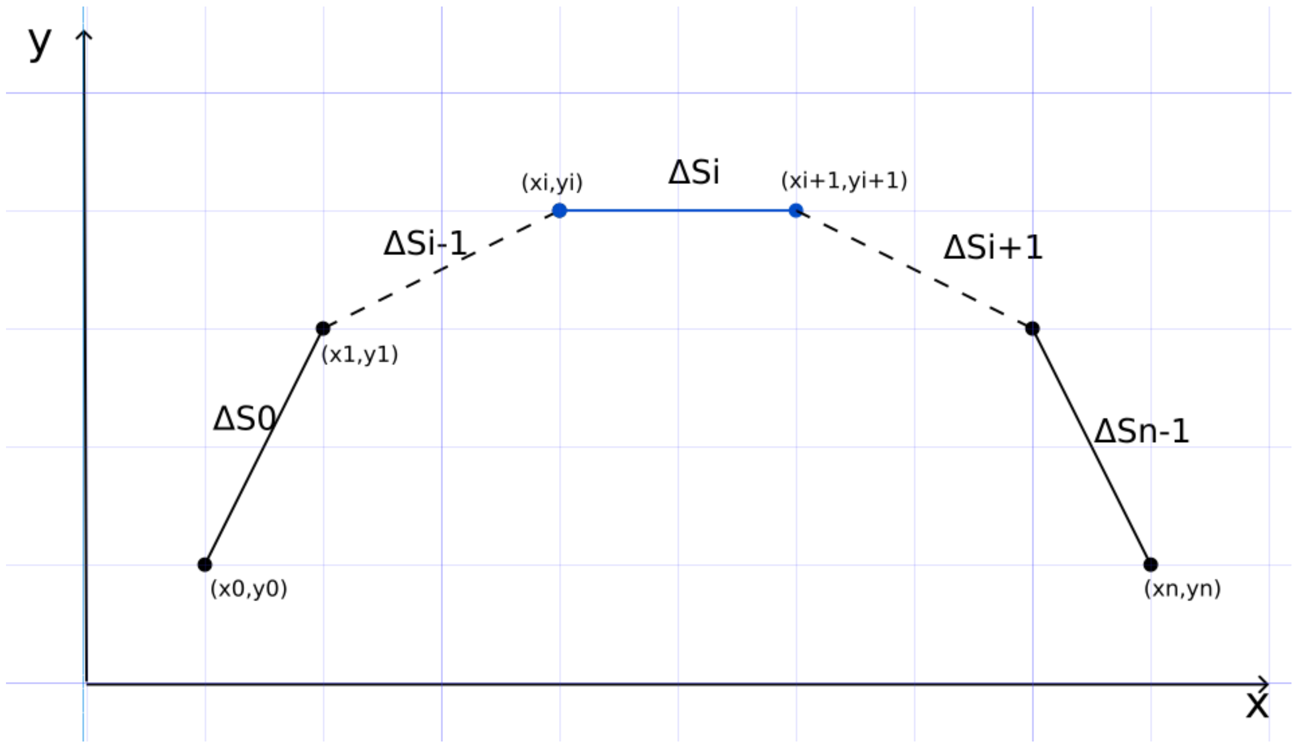
二者对比







分段多项式螺旋曲线平滑



n个点将参考线分割为n-1段，其中**每段多项式螺旋曲线**的参数方程描述为

$$\theta(s) = as^5 + bs^4 + cs^3 + ds^2 + e^s + f \tag{5}$$

其中：$\theta(s)$ 为**螺旋线上的切线方向**，s表示沿**螺旋曲线的弧长**，那么在给定区间$s \in [0, s_n]$即可完整的描述一段曲线

单条五次多项式螺旋曲线

确定过程跟五次多项式一样，每段多项式螺旋线有6个未知系数系数a, b, c, d, e, f，可由曲线的起点和终点的6个方程确定：

$$\begin{aligned} \text{起点状态：} \quad & \theta_i(0) = \theta_i \\ & \dot{\theta}_i(0) = \dot{\theta}_i \\ & \ddot{\theta}_i(0) = \ddot{\theta}_i \\ \text{终点状态：} \quad & \theta_i(\Delta s) = \theta_{i+1} \\ & \dot{\theta}_i(\Delta s) = \dot{\theta}_{i+1} \\ & \ddot{\theta}_i(\Delta s) = \ddot{\theta}_{i+1} \end{aligned} \tag{6}$$

其中：θ_i为起点方向，$\dot{\theta}_i$为起点曲率(对s求导)，$\ddot{\theta}_i$为起点的曲率导数，终点状态类似。

可提前将方程 $AX = b$ 进行逆运算求解，提高计算效率，参数方程为：

$$\begin{aligned} a_i &= \frac{-6\theta_i}{s^5} - \frac{3\dot{\theta}_i}{s^4} - \frac{\ddot{\theta}_i}{2s^3} + \frac{6\theta_{i+1}}{s^5} - \frac{3\dot{\theta}_{i+1}}{s^4} + \frac{\ddot{\theta}_{i+1}}{2s^3} \\ b_i &= \frac{15\theta_i}{s^4} + \frac{8\dot{\theta}_i}{s^3} + \frac{3\ddot{\theta}_i}{2s^2} - \frac{15\theta_{i+1}}{s^4} + \frac{7\dot{\theta}_{i+1}}{s^3} - \frac{\ddot{\theta}_{i+1}}{s^2} \\ c_i &= \frac{-10\theta_i}{s^3} - \frac{6\dot{\theta}_i}{s^2} - \frac{3\ddot{\theta}_i}{2s} + \frac{10\theta_{i+1}}{s^3} - \frac{4\dot{\theta}_{i+1}}{s^2} + \frac{\ddot{\theta}_{i+1}}{2s} \\ d_i &= \frac{\ddot{\theta}_i}{2}; \\ e_i &= \dot{\theta}_i \\ f_i &= \theta_i \end{aligned} \tag{7}$$

分段五次多项式螺旋曲线

为了确保第i段与第$i+1$段螺旋曲线之间的连续，需要保证满足如下关系：

1. 位置连续

> 通坐标变换,得到笛卡尔坐标系下的位置(x_i, y_i)

$$\begin{aligned} x_{i+1} &= x_i + \int_0^{\Delta s_i} \cos(\theta(s)) ds \\ y_{i+1} &= y_i + \int_0^{\Delta s_i} \sin(\theta(s)) ds \end{aligned} \tag{8}$$

2. 方向连续

$$\theta_{i+1} = \theta_{i+1}(\Delta s) \tag{9}$$

3. 曲率连续

$$\dot{\theta}_{i+1} = \dot{\theta}_{i+1}(\Delta s) \tag{10}$$

4. 曲率变化率连续

$$\ddot{\theta}_{i+1} = \ddot{\theta}_{i+1}(\Delta s) \tag{11}$$

数学模型

优化变量

通过上述分析得到此问题的优化变量为：位置、方向、曲率、曲率变化率以及分段弧长s

$$\vec{q} = [\vec{\theta}, \vec{\dot{\theta}}, \vec{\ddot{\theta}}, \vec{x}, \vec{y}, \vec{\Delta s}] \tag{12}$$

每个元素均为n行向量，如$\vec{\Delta s} = [\Delta s_0 \quad \Delta s_1 \quad \cdots \quad \Delta s_{n-1}]$，其中$\Delta s_0 = 0$

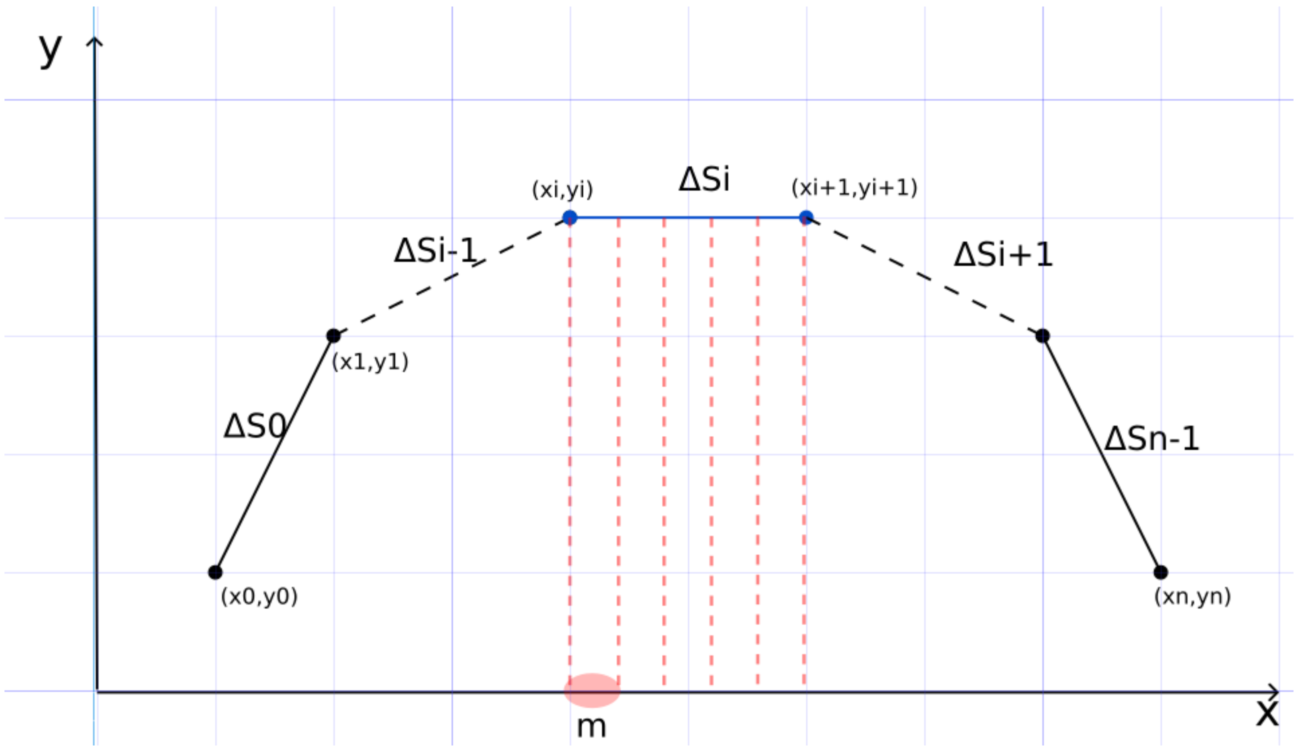
目标函数

1. 分段弧长
2. 曲率
3. 曲率变化率

$$cost\ \ function = w_{length} \cdot \sum_{i=0}^{n-1} \Delta s_i + w_{dkappa} \cdot \sum_{i=0}^{n-1}\sum_{i=0}^{m-1} \left(\dot{\theta}(s_j)\right)^2 + w_{ddkappa} \cdot \sum_{i=0}^{n-1}\sum_{i=0}^{m-1} \left(\ddot{\theta}(s_j)\right)^2 \tag{13}$$

式中的m是什么？

m表示将第i段螺旋曲线等分成m个子段，这个过程是为了提高数值积分的计算精度



约束条件

> 约束问题分为变量的 **bound** 和 约束条件**constraints**

- **起点等式约束**(bound)

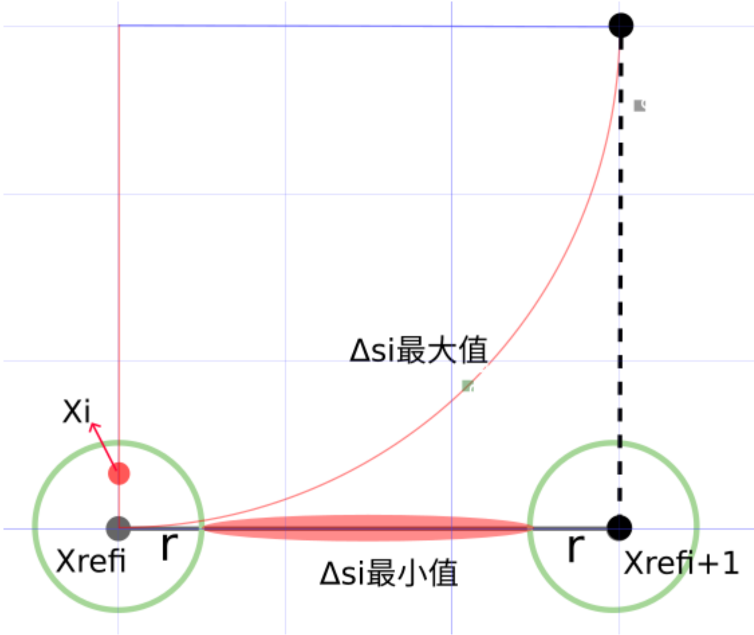
$$\begin{aligned} \theta_0 &= \theta_{start} \\ \kappa_0 &= \kappa_{start} \\ \dot{\kappa_0} &= \dot{\kappa_{start}} \\ x_0 &= x_{ref_0} \\ y_0 &= y_{ref_0} \end{aligned} \tag{14}$$

- **中间点**
  - 动力学约束(bound)

$$\begin{aligned} \text{最大转角：} & \theta_{i-1} - \frac{\pi}{2} \le \theta_i \le \theta_{i-1} + \frac{\pi}{2} \\ \text{曲率：} & -0.25 \le \kappa \le +0.25 \\ \text{曲率变化率：} & -0.02 \le \dot{\kappa} \le +0.02 \end{aligned} \tag{15}$$

  - 中间点边界约束
    - 规定连续两点之间最大转向为四分之一圆弧(防止转弯太急)(bound)

$$\begin{aligned} & x_{ref_i} - r_i \le x_i \le x_{ref_i} + r_i \\ & y_{ref_i} - r_i \le y_i \le y_{ref_i} + r_i \\ & D_i - 2r_i \le \Delta s_i \le D_i \frac{\pi}{2} \\ & \text{其中}, D_i = \sqrt{(x_{ref_i} - x_{ref_{i+1}})^2 + (y_{ref_i} - y_{ref_{i+1}})^2} \end{aligned} \tag{16}$$



- 分段之间的连接点等式约束(bound + constraints)

$$
\begin{aligned}
x_{i+1} &= x_i + \int_0^{\Delta s_i} \cos\left(\theta(s)\right) ds \qquad (constraints) \\
y_{i+1} &= y_i + \int_0^{\Delta s_i} \sin\left(\theta(s)\right) ds \qquad (constraints) \\
\theta_{i+1} &= \theta_i(\Delta s_i) \\
\dot{\theta_{i+1}} &= \dot{\theta}_i(\Delta s_i) \\
\ddot{\theta_{i+1}} &= \ddot{\theta}_i(\Delta s_i)
\end{aligned}
\tag{17}
$$

- **终点等式约束(bound)**

$$
\begin{aligned}
\theta_{n-1} &= \theta_{end} \\
\kappa_{n-1} &= \kappa_{end} \\
\dot{\kappa_{n-1}} &= \dot{\kappa_{end}} \\
x_{n-1} &= x_{ref_{n-1}} \\
y_{n-1} &= y_{ref_{n-1}}
\end{aligned}
\tag{18}
$$

- **位置平移非等式约束**

$$
(x_i - x_{ref_i})^2 + (y_i - y_{ref_i})^2 \le r_i^2 \qquad (constraints) \tag{19}
$$

数学求解库

问题规模

- 约束变量个数

  每个点的约束变量为5个优化变量和1个分段弧长变量，设路径点的数量为n，则总的约束变量个数为：

$$
NumsVariable = 5n + n = 6n \tag{20}
$$

  矩阵方程为：

$$
[\theta_0, \dot{\theta}_0, \ddot{\theta}_0, x_0, y_0, \theta_1, \dot{\theta}_1, \ddot{\theta}_1, x_1, y_1, \cdots, \theta_n, \dot{\theta}_n, \ddot{\theta}_n, x_n, y_n, \Delta s_0, \Delta s_1, \cdots, \Delta s_n]^T_{6n\times 1} \tag{21}
$$

- 约束的数量

  除起点外，每个点包含有 x, y 两个维度的约束和每个点的平移约束，则总的约束数量为：

$$
NumConstraints = 2n + n = 3n \tag{22}
$$

```
bool SpiralProblemInterface::get_nlp_info(int& n, int& m, int& nnz_jac_g,
                                          int& nnz_h_lag,
                                          IndexStyleEnum& index_style) {
  // number of variables
  n = num_of_points_ * 5 + num_of_points_ - 1;
  num_of_variables_ = n;

  // number of constraints
  // b. positional equality constraints;
  // totally 2 * (num_of_points - 1) considering x and y separately
  m = (num_of_points_ - 1) * 2;
  // a. positional movements; totally num_of_points
  m += num_of_points_;
  num_of_constraints_ = m;
  ...
}
```

问题约束边界

设路径点个数为n，对每个优化变量设置其上下边界，，则自变量的边界**bound**约束个数为：

> 起终点边界:10 个
> 动力学约束边界:$3n$ 个
> 中间点范围边界: $3n$ 个

$$NumVarBound = 2 * (10 + 3n + 3n) = 12n + 20 \tag{23}$$

```cpp
bool SpiralProblemInterface::get_bounds_info(int n, double* x_l, double* x_u,
                                              int m, double* g_l, double* g_u) {
    has_fixed_start_point_
    has_fixed_end_point_
    // theta
    x_l[index] = theta_lower;
    x_u[index] = theta_upper;

    // kappa
    x_l[index + 1] = kappa_lower;
    x_u[index + 1] = kappa_upper;

    // dkappa
    x_l[index + 2] = dkappa_lower;
    x_u[index + 2] = dkappa_upper;

    // x
    x_l[index + 3] = x_lower;
    x_u[index + 3] = x_upper;

    // y
    x_l[index + 4] = y_lower;
    x_u[index + 4] = y_upper;

    // delta_s
    x_l[variable_offset + i] =
      point_distances_[i] - 2.0 * default_max_point_deviation_;
    x_u[variable_offset + i] = point_distances_[i] * M_PI * 0.5;
    ...
```

约束条件**constraints** 的上下边界约束个数为:

> 连接点等式约束: $3n$ 个

$$NumConstraints = 2 * 3n = 6n \tag{24}$$

```cpp
  ...
  // constraints
  // a. positional equality constraints
  for (int i = 0; i + 1 < num_of_points_; ++i) {
    // for x
    //将约束方程转化为 g(x) = 0 的形式
    g_l[i * 2] = 0.0;
    g_u[i * 2] = 0.0;

    // for y
    g_l[i * 2 + 1] = 0.0;
    g_u[i * 2 + 1] = 0.0;
  }
  // b. positional deviation constraints
  int constraint_offset = 2 * (num_of_points_ - 1);
  for (int i = 0; i < num_of_points_; ++i) {
    g_l[constraint_offset + i] = 0.0;
    g_u[constraint_offset + i] =
        default_max_point_deviation_ * default_max_point_deviation_;
  }
}
```

则总的约束个数为:

$$Num = 12n + 20 + 6n = 18n + 20 \tag{25}$$

设置初始值

优化变量的初始值为原始参考线的状态

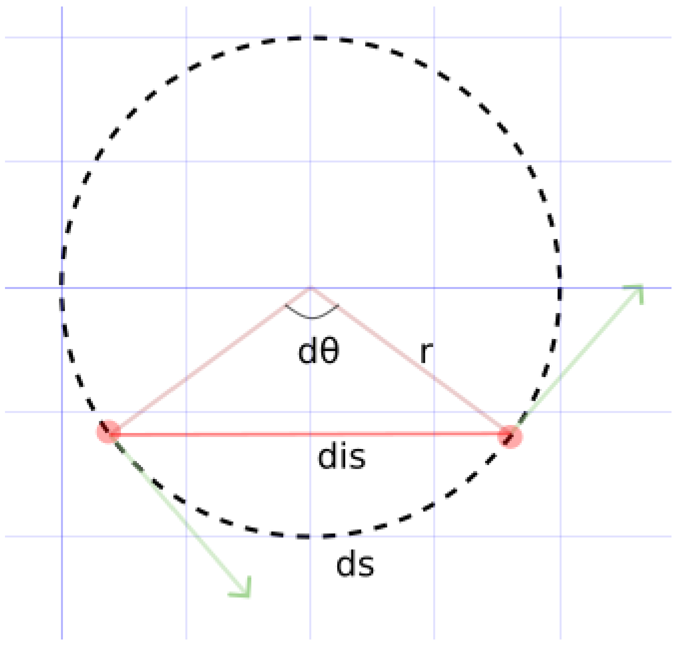
- 非第一个点的曲率变化率定义为0，第一个点的曲率、曲率变化率根据实际给出
- 计算弧长

$$\begin{aligned}
&\text{由正弦定理:} \ \frac{sin\theta}{dis} = \frac{sin(\frac{\pi-\theta}{2})}{r} \\
&=> dis = \frac{sin\theta \cdot r}{cos\frac{\theta}{2}}, \qquad \text{又} s = \theta \cdot r \\
&=> s = \frac{dis \cdot \theta}{2 \cdot sin\frac{\theta}{2}}
\end{aligned} \tag{26}$$

> 注:
>
> `x[variable_offset + i] = point_distances_[i] / std::cos(0.5 * delta_theta);`
>
> 上面计算s的推导跟代码中的表达有些出入

- 计算曲率

$$\Delta s = \theta \cdot r => kappa = \frac{1}{r} = \frac{\theta}{s} \tag{27}$$




```
bool SpiralProblemInterface::get_starting_point(int n, bool init_x, double* x,
                                                bool init_z, double* z_L,
                                                double* z_U, int m,
                                                bool init_lambda,
                                                double* lambda) {
  ...
  for (int i = 0; i < num_of_points_; ++i) {
    int index = i * 5;
    x[index] = relative_theta_[i];
    x[index + 1] = 0.0;
    x[index + 2] = 0.0;
    x[index + 3] = init_points_[i].x();
    x[index + 4] = init_points_[i].y();
  }

  int variable_offset = num_of_points_ * 5;
  for (int i = 0; i + 1 < num_of_points_; ++i) {
    double delta_theta = relative_theta_[i + 1] - relative_theta_[i];
    x[variable_offset + i] = point_distances_[i] / std::cos(0.5 * delta_theta);
  }

  for (int i = 0; i + 1 < num_of_points_; ++i) {
    double delta_theta = relative_theta_[i + 1] - relative_theta_[i];
    x[(i + 1) * 5 + 1] = delta_theta / x[variable_offset + i];
  }
  x[1] = x[6];
  //设置起点
  if (has_fixed_start_point_) {
    x[0] = start_theta_;
    x[1] = start_kappa_;
    x[2] = start_dkappa_;
  }
  return true;
}
```

设置目标函数

$$cost\ \ function = w_{length} \cdot \sum_{i=0}^{n-1} \Delta s_i + w_{kappa} \cdot \sum_{i=0}^{n-1} \sum_{i=0}^{m-1} (\dot{\theta}(s_j))^2 + w_{ddkappa} \cdot \sum_{i=0}^{n-1} \sum_{i=0}^{m-1} (\ddot{\theta}(s_j))^2 \tag{28}$$

```
bool SpiralProblemInterface::eval_f(int n, const double* x, bool new_x,
                                    double& obj_value) {
  CHECK_EQ(n, num_of_variables_);
  if (new_x) {
    update_piecewise_spiral_paths(x, n);
  }

  obj_value = 0.0;
  for (int i = 0; i + 1 < num_of_points_; ++i) {
    const auto& spiral_curve = piecewise_paths_[i];
    double delta_s = spiral_curve.ParamLength();

    obj_value += delta_s * weight_curve_length_;
    // num_of_internal_points_为目标函数中的m,定义为5
    // 对每两个节点之间均分出了5个内部节点, 分别将内部节点的曲率和曲率变化率加权求和,提高计算精度
    for (int j = 0; j < num_of_internal_points_; ++j) {
      double ratio =
          static_cast<double>(j) / static_cast<double>(num_of_internal_points_);
      double s = ratio * delta_s;

```

```
        double kappa = spiral_curve.Evaluate(1, s);
        obj_value += kappa * kappa * weight_kappa_;

        double dkappa = spiral_curve.Evaluate(2, s);
        obj_value += dkappa * dkappa * weight_dkappa_;
      }
    }
    return true;
}
```

设置目标梯度

略复杂

```
bool SpiralProblemInterface::eval_grad_f(int n, const double* x, bool new_x,
                                         double* grad_f) {
  ...
    grad_f[variable_offset + i] += weight_curve_length_ * 1.0;
  ...
    grad_f[variable_offset + i] += weight_kappa_ * 2.0 * kappa *
    spiral_curve.DeriveKappaDerivative(
    DELTA_S, j, num_of_internal_points_);
  ...
    grad_f[variable_offset + i] += weight_dkappa_ * 2.0 * dkappa *
    spiral_curve.DeriveDKappaDerivative(
    DELTA_S, j, num_of_internal_points_);
}
```

设置约束函数

1. 分段曲线连接点的等式约束

$$
\begin{aligned}
g_{1i} &= \left(x_{i+1} - x_i - \int_0^{\Delta s_i} \cos(\theta(s))ds\right)^2 \\
g_{2i} &= \left(y_{i+1} - y_i - \int_0^{\Delta s_i} \sin(\theta(s))ds\right)^2
\end{aligned}
\tag{29}
$$

```
bool SpiralProblemInterface::eval_g(int n, const double* x, bool new_x, int m,
                                    double* g) {
  ...
// first, fill in the positional equality constraints
  for (int i = 0; i + 1 < num_of_points_; ++i) {
    int index0 = i * 5;
    int index1 = (i + 1) * 5;

    const auto& spiral_curve = piecewise_paths_[i];
    double delta_s = spiral_curve.ParamLength();

    double x_diff = x[index1 + 3] - x[index0 + 3] -
                    spiral_curve.ComputeCartesianDeviationX(delta_s);
    g[i * 2] = x_diff * x_diff;

    double y_diff = x[index1 + 4] - x[index0 + 4] -
                    spiral_curve.ComputeCartesianDeviationY(delta_s);
    g[i * 2 + 1] = y_diff * y_diff;
  }
  ...
```

2. 位置平移非等式约束(将非等式转化为等式约束)

$$
g_{i3} = (x_i - x_{ref_i})^2 + (y_i - y_{ref_i})^2 \tag{30}
$$

```
  ...
// second, fill in the positional deviation constraints
  int constraint_offset = 2 * (num_of_points_ - 1);
  for (int i = 0; i < num_of_points_; ++i) {
    int variable_index = i * 5;
    double x_cor = x[variable_index + 3];
    double y_cor = x[variable_index + 4];

    double x_diff = x_cor - init_points_[i].x();
    double y_diff = y_cor - init_points_[i].y();

    g[constraint_offset + i] = x_diff * x_diff + y_diff * y_diff;
  }
  return true;
}
```

3.

设置Jacobian矩阵

略

```
bool SpiralProblemInterface::eval_jac_g(int n, const double* x, bool new_x,
                                        int m, int nele_jac, int* iRow,
                                        int* jCol, double* values) {
  ...
}
```

设置Hessian矩阵

调用拟牛顿法，无需求解Hessian矩阵

```
app->Options()->SetStringValue("hessian_approximation","limited-memory");
...
bool SpiralProblemInterface::eval_h(int n, const double* x, bool new_x,
                                    double obj_factor, int m,
                                    const double* lambda, bool new_lambda,
                                    int nele_hess, int* iRow, int* jCol,
                                    double* values) {
  ACHECK(false);
  return true;
}
```

如何求解等式约束中的位置积分？

螺旋曲线笛卡尔坐标变换

对于任意$s \in [0, s_i]$,任意坐标$x(s)$,$y(s)$表示为：

$$
\begin{aligned}
x(s) &= x_i + \int_0^s cos(\theta(s))ds \\
y(s) &= y_i + \int_0^s sin(\theta(s))ds
\end{aligned}
\tag{31}
$$

上述积分求不出其解析解，实际计算利用数值积分的方法计算积分，Apollo默认使用**高斯-勒让德求积公式**进行求解（调用接口为`ComputeCartesianDeviationX()`）：

$$
\begin{aligned}
x(s) &= x_i + \frac{s}{2} \cdot \sum_{i=0}^{n} w_i \cdot cos(\theta(\frac{s}{2} \cdot \zeta_i + \frac{s}{2})) \\
y(s) &= y_i + \frac{s}{2} \cdot \sum_{i=0}^{n} w_i \cdot sin(\theta(\frac{s}{2} \cdot \zeta_i + \frac{s}{2}))
\end{aligned}
\tag{32}
$$

求解步骤：

1. 将自变量区间$[0, \Delta s_i]$转换到$\xi_i \in [-1, 1]$
2. Apollo中采用五点高斯积分，故$n = 4$，查表得到在$n = 4$处的x_i取值和权重w_i，计算出对应的函数值$f(x_i)$与权重的乘积再累加

Gauss-Legendre求积公式

> 只需计算特定几个点处的多项式函数值的加权和，即可逼近积分值, 并且能保证较满意的精度.

1. **积分法则**

   考虑实数区间 $[-1, 1]$上的积分

   $$\int_{-1}^{1} f(\xi)\, d\xi. \tag{1}$$

   一个数值积分法则 $(\xi_i, w_i), i = 1, \ldots, n$,即积分点和积分权重, 具有如下形式

   $$\int_{-1}^{1} f(\xi)\, d\xi \approx \sum_{i=1}^{n} \omega_i f(\xi_i). \tag{2}$$

   希望选取的积分法则具有尽可能高的数值精度, 即数值积分 (2) 对尽可能高阶的多项式准确成立. 一般来说, 2n 个参数可以唯一确定一个 2n−1 次多项式. 因此希望可以找到积分法则使得下面的等式成立

   $$\int_{-1}^{1} p(\xi)\, d\xi = \sum_{i=1}^{n} \omega_i p(\xi_i), \quad \forall p \in \mathbb{P}_{2n-1}. \tag{3}$$

   式中：

   $$
   \begin{aligned}
   &\xi_i, i = 1, 2, \ldots, n\text{为}P_n(\xi_i)\text{的}n\text{个根，权重}\omega_i = \int_{-1}^{1} L_i(\xi)\, d\xi \\
   &\text{其中}L_i(\xi) = \sum_{j=1, j\neq i}^{n} \frac{\xi - \xi_j}{\xi_i - \xi_j}\text{为}Lagrange\text{插值基函数}
   \end{aligned}
   \tag{33}
   $$

2. **一般区间上的数值积分**

通常的做法是将一般区间$[a,b], a,b \in R$上的积分线性变换到参考单元$[-1,1]$上.

$$\begin{aligned}\int_a^b f(x)dx &= \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}\xi + \frac{a+b}{2}\right)d\xi \\ &\approx \frac{b-a}{2}\sum_i \omega_i f\left(\frac{b-a}{2}\xi_i + \frac{a+b}{2}\right)\end{aligned} \tag{34}$$

因此, 区间 $[a,b]$上的积分法则(x_i, w_i) 为

$$x_i = \frac{b-a}{2}\xi_i + \frac{a+b}{2}, \quad w_i = \frac{b-a}{2}\omega_i, \tag{35}$$

即

$$\int_a^b f(x)dx \approx \sum_i w_i f(x_i). \tag{36}$$

通过Gauss-Legendre 积分积分点与权重系数表，求得数值积分值

n	x_i	w_i
1	0.000000000000000	2.000000000000000
2	-0.577350269189626	1.000000000000000
	0.577350269189626	1.000000000000000
3	-0.774596669241484	0.555555555555556
	0.000000000000000	0.888888888888889
	0.774596669241484	0.555555555555556
4	-0.861136311594053	0.347854845137454
	-0.339981043584856	0.652145154862546
	0.339981043584856	0.652145154862546
	0.861136311594053	0.347854845137454
5	-0.906179845938664	0.236926885056189
	-0.538469310105683	0.478628670499366
	0.000000000000000	0.568888888888889
	0.538469310105683	0.478628670499366
	0.906179845938664	0.236926885056189

Apollo三种平滑器对比

未完待续...