Dubins\_ReedsSheeps算法

概述

本文档主要针对APS自动泊车路径规划部分；针对B样条曲线做补充，样条曲线能很好的规划控制点生成光滑曲线，但在样条曲线中，不能很好的表现车辆姿态。通过引入任意时刻车辆航向信息，搭建ReedsSheeps算法模型，理论上可以解决车辆任意姿态任意时刻的泊入泊出问题。

通过前文B样条曲线分析，针对每个循环调度时刻，根据车辆当前位姿与目标位姿，计算ReedsSheeps轨迹，然后取轨迹点生成光滑的B样条目标轨迹线。本文中未加车辆控制部分，车辆运动模型用初始轨迹参数计算；由坐标变换关系得，参数扰动由车辆当前位置误差累积，不累积计算目标车位误差，在定义的世界坐标系中，实时仿真车辆位置。

规划算法：通过ReedsSheeps算法计算车辆泊出入车位实时的圆弧直线轨迹，然后再B样条离散轨迹为连续光滑轨迹。

问题及不足：

1、现阶段只针对平行泊车轨迹规划部分，simulink中ReedsSheeps算法只加入CSC分段，车辆运动控制模型还未加入，误差扰动模块考虑的较为简单。后续需能适应多分段轨迹并加入运动学模型。

2、相比B样条只考虑轨迹位置，现阶段多考虑车辆姿态，但轨迹规划的速度和时间维度并未考虑。模型中只考虑了在理想情况下的转速和航向，在实际情况下，转向和速度含时间变化的运动规划还未考虑；

3、在算法函数完成的情况下，搭建转化生成符合MAAB标准的simulink模型还不熟悉，需要转化时间。

ReedsSheeps算法

Dubins曲线证明在车辆最小转弯半径的约束下，在不考虑车辆后退的前提下，车辆在任意位置以任意姿态均能通过Dubins曲线到达； ReedsSheeps是Dubins曲线的延申，考虑了车辆后退的情况；在满足Dubins曲线条件约束和车辆能后退的条件下，ReedsSheeps曲线是以最短路径到达目的地的轨迹曲线。

Dubins曲线

Dubins证明，一个最优路径一定是由分段圆弧(单位圆)和线段组成的平滑曲线，且最多3部分组成。对任意车辆两起始状态，最短路径组合只有6种，如下所示组合（L、R、S分别代表左转右转直行）。

$\displaystyle \{ LRL,\; RLR, \; LSL,\; LSR,\; RSL,\; RSR \} .$

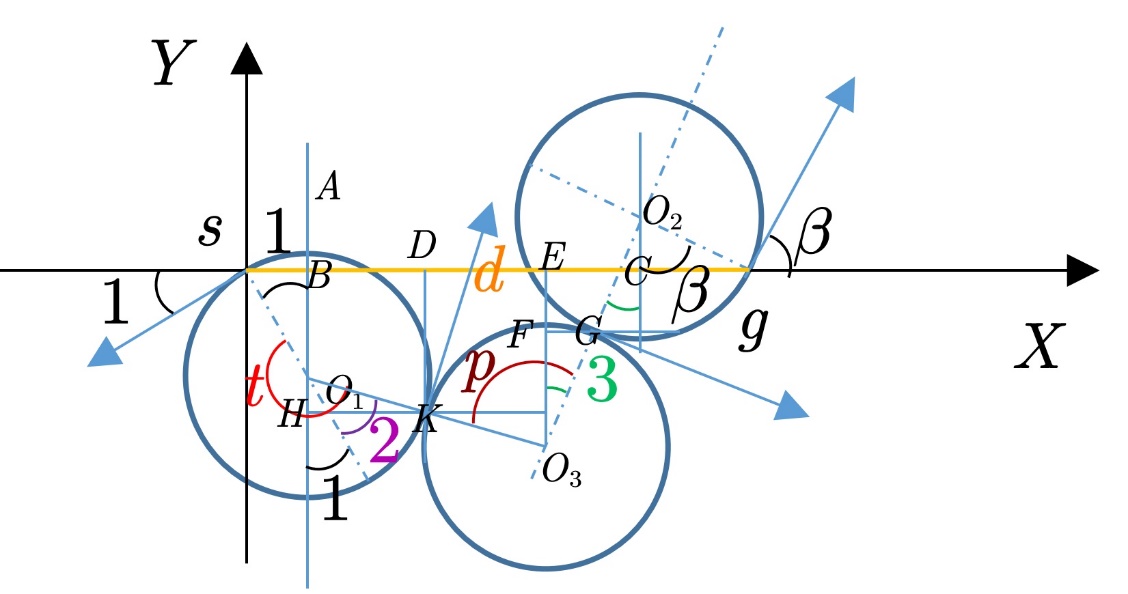
参考附件A Comprehensive, Step-by-Step Tutorial on

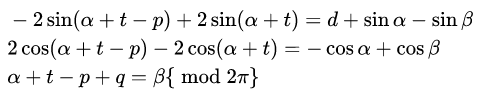
Computing Dubin’s Curves.pdf 论文。

Dubins曲线可应用于泊出或RRT离散点轨迹平滑处理。

Dubins只考虑两种形式，CCC,CSC；此处只计算分析了Dubins计算，根据几何知识，先将起始点坐标变换到原点，并旋转终点也落在X轴；变换后的几何关系如下所示，易推Dubins关键的6个函数，如下所示。此部分并未用于模型中，模型在ReedsSheeps部分分析使用。

1、CCC形式：（LRL）





代码：

function L = LRL(alpha,beta,d)

tmp\_lrl = (6. - d\*d + 2\*cos(alpha - beta) + 2\*d\*(- sin(alpha) + sin(beta))) / 8.;

if( abs(tmp\_lrl) > 1)

L = [inf inf inf inf];

else

p = mod(( 2\*pi - acos( tmp\_lrl ) ), 2\*pi);

t = mod((-alpha - atan2( cos(alpha)-cos(beta), d+sin(alpha)-sin(beta) ) + p/2), 2\*pi);

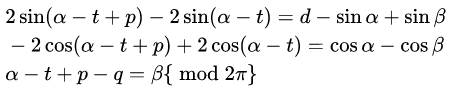
q = mod((mod(beta, 2\*pi) - alpha -t + mod(p, 2\*pi)), 2\*pi);

L=[t+p+q t p q];

end

end

2、CCC形式： (RLR)



function L = RLR(alpha,beta,d)

tmp\_rlr = (6. - d\*d + 2\*cos(alpha - beta) + 2\*d\*(sin(alpha)-sin(beta))) / 8.;

if( abs(tmp\_rlr) > 1)

L = [inf inf inf inf];

else

p = mod(( 2\*pi - acos( tmp\_rlr ) ), 2\*pi);

t = mod((alpha - atan2( cos(alpha)-cos(beta), d-sin(alpha)+sin(beta) ) + mod(p/2, 2\*pi)), 2\*pi);

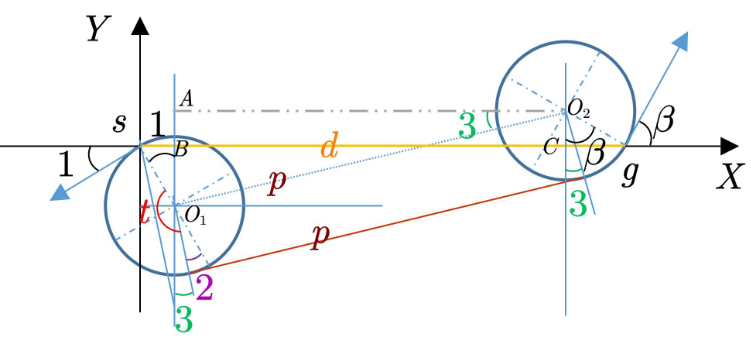
q = mod((alpha - beta - t + mod(p, 2\*pi)), 2\*pi);

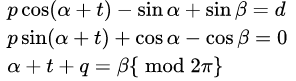
L=[t+p+q t p q];

end

end

3、CSC形式：（LSL）





function L = LSL(alpha,beta,d)

tmp0 = d + sin(alpha) - sin(beta);

p\_squared = 2 + (d\*d) -(2\*cos(alpha - beta)) + (2\*d\*(sin(alpha) - sin(beta)));

if( p\_squared < 0 )

L = [inf inf inf inf];

else

tmp1 = atan2( (cos(beta)-cos(alpha)), tmp0 );

t = mod((-alpha + tmp1 ), 2\*pi);

p = sqrt( p\_squared );

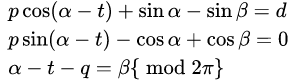
q = mod((beta - tmp1 ), 2\*pi);

L=[t+p+q t p q];

end

end

4、CSC形式（RSR）



function L = RSR(alpha,beta,d)

tmp0 = d-sin(alpha)+sin(beta);

p\_squared = 2 + (d\*d) -(2\*cos(alpha - beta)) + (2\*d\*(sin(beta)-sin(alpha)));

if( p\_squared < 0 )

L = [inf inf inf inf];

else

tmp1 = atan2( (cos(alpha)-cos(beta)), tmp0 );

t = mod(( alpha - tmp1 ), 2\*pi);

p = sqrt( p\_squared );

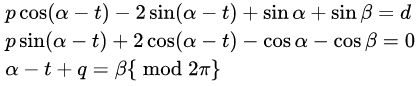
q = mod(( -beta + tmp1 ), 2\*pi);

L=[t+p+q t p q];

end

end

5、CSC形式（LSR）



function L = LSR(alpha,beta,d)

p\_squared = -2 + (d\*d) + (2\*cos(alpha - beta)) + (2\*d\*(sin(alpha)+sin(beta)));

if( p\_squared < 0 )

L = [inf inf inf inf];

else

p = sqrt( p\_squared );

tmp2 = atan2( (-cos(alpha)-cos(beta)), (d+sin(alpha)+sin(beta)) ) - atan2(-2.0, p);

t = mod((-alpha + tmp2), 2\*pi);

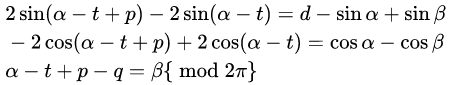
q = mod(( -mod((beta), 2\*pi) + tmp2 ), 2\*pi);

L=[t+p+q t p q];

end

end

6、CSC形式（RLR）



function L = RLR(alpha,beta,d)

tmp\_rlr = (6. - d\*d + 2\*cos(alpha - beta) + 2\*d\*(sin(alpha)-sin(beta))) / 8.;

if( abs(tmp\_rlr) > 1)

L = [inf inf inf inf];

else

p = mod(( 2\*pi - acos( tmp\_rlr ) ), 2\*pi);

t = mod((alpha - atan2( cos(alpha)-cos(beta), d-sin(alpha)+sin(beta) ) + mod(p/2, 2\*pi)), 2\*pi);

q = mod((alpha - beta - t + mod(p, 2\*pi)), 2\*pi);

L=[t+p+q t p q];

end

end

7、根据几何关系计算每个word（L\S\R）的轨迹函数分段点

function seg\_end = dubins\_segment(seg\_param, seg\_init, seg\_type)

if( seg\_type == 'L' )

seg\_end(1) = seg\_init(1) + sin(seg\_init(3)+seg\_param) - sin(seg\_init(3));

seg\_end(2) = seg\_init(2) - cos(seg\_init(3)+seg\_param) + cos(seg\_init(3));

seg\_end(3) = seg\_init(3) + seg\_param;

elseif( seg\_type == 'R' )

seg\_end(1) = seg\_init(1) - sin(seg\_init(3)-seg\_param) + sin(seg\_init(3));

seg\_end(2) = seg\_init(2) + cos(seg\_init(3)-seg\_param) - cos(seg\_init(3));

seg\_end(3) = seg\_init(3) - seg\_param;

elseif( seg\_type == 'S' )

seg\_end(1) = seg\_init(1) + cos(seg\_init(3)) \* seg\_param;

seg\_end(2) = seg\_init(2) + sin(seg\_init(3)) \* seg\_param;

seg\_end(3) = seg\_init(3);

end

end

任意起始点任意角度验证如下：

r=5;

p1 = [9 3 20\*pi/180];

p2 = [10 10 120\*pi/180];

dx = p2(1) - p1(1);

dy = p2(2) - p1(2);

d = sqrt( dx^2 + dy^2 ) / r;

theta = mod(atan2( dy, dx ), 2\*pi);

alpha = mod((p1(3) - theta), 2\*pi);

beta = mod((p2(3) - theta), 2\*pi);

L = zeros(6,4);

L(1,:) = LSL(alpha,beta,d);

L(2,:) = LSR(alpha,beta,d);

L(3,:) = RSL(alpha,beta,d);

L(4,:) = RSR(alpha,beta,d);

L(5,:) = RLR(alpha,beta,d);

L(6,:) = LRL(alpha,beta,d);

[~,ind] = min(L(:,1));

types=['LSL';'LSR';'RSL';'RSR';'RLR';'LRL'];

p\_start = [0 0 p1(3)];

mid1 = dubins\_segment(L(ind,2),p\_start,types(ind,1));

mid2 = dubins\_segment(L(ind,3), mid1,types(ind,2));

path=[];

for step=0:0.05:L(ind,1)\*r

t = step / r;

if( t < L(ind,2) )

end\_pt=dubins\_segment( t, p\_start,types(ind,1));

elseif( t < L(ind,2)+L(ind,3) )

end\_pt = dubins\_segment( t-L(ind,2),mid1,types(ind,2));

else

end\_pt = dubins\_segment( t-L(ind,2)-L(ind,3),mid2,types(ind,3));

end

end\_pt(1) = end\_pt(1) \* r + p1(1);

end\_pt(2) = end\_pt(2) \* r + p1(2);

end\_pt(3) = mod(end\_pt(3), 2\*pi);

path=[path;end\_pt];

end

部分角度及位姿仿真结果如下，

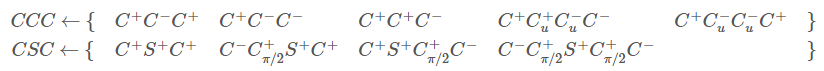


ReedsSheeps算法

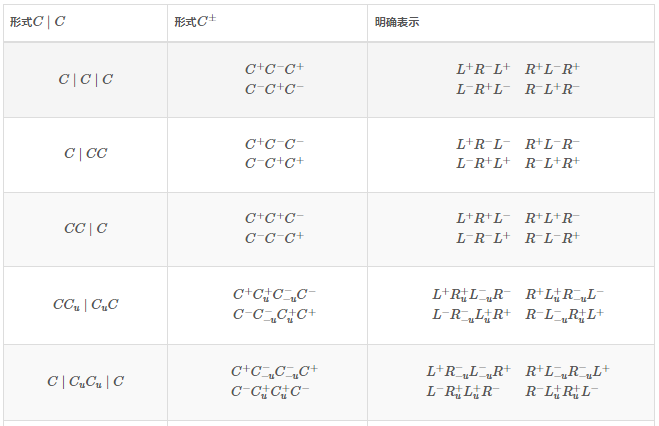
Dubins 曲线只能适用于车辆前进的轨迹规划，不能适应后退挡位切换路线规划，与车辆实际运行不符。 ReedsSheeps 曲线是 Dubins 的扩展， ReedsSheeps 所指代的轨迹是满足车辆最小转弯半径约束下，车辆从起始点能到达目的地终点的最短路径。

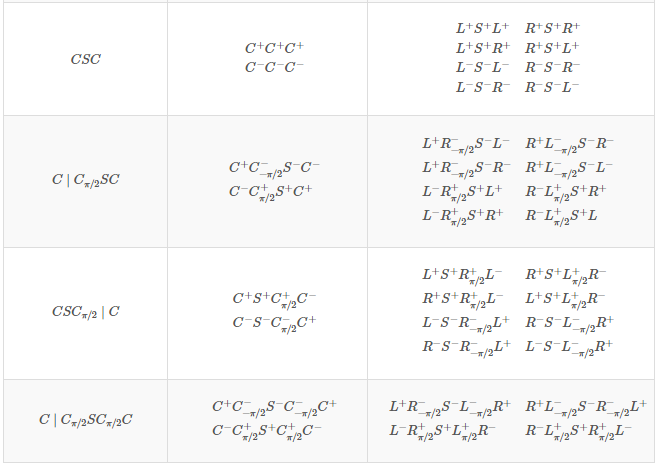
原文参考附件《OPTIMAL PATHS FOR A CAR THAT GOES BOTH》.pdf。

在车辆中，字母L、R、S参考方向盘转角，符号+和-参考挡位切换，使用C、S可表示ReedsSheeps路径集合；

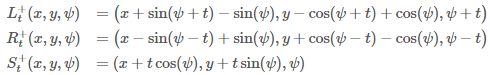


用C替换L或R，共48word，任何长度小于的最佳路径一定是CSC形式。任何最佳路径一定是由C和S的字段组成。使用一系列的特殊参数，可将所有有限的字段简化为CCC或CSC的形式。





归一化轨迹半径r = 1;设起始坐标（x,y,Ψ）,经过时间t，（L左转R右转S直行）行驶路径轨迹方程如下：

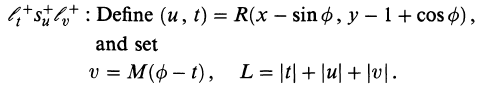


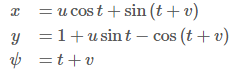
当t<0时，L、R、S为后退档左转右转直线。

由于车辆的起始位置和终点位置无法穷举，所以在计算路径之前，需要将车辆的位置和姿态作归一化处理。

ReedsSheeps证明，在轨迹字段处理中，由简单的几何关系可得，可以通过车辆前进后退和左转右转化简48种字段。通过时间变换timeflip可以化简车辆行进方向，反射变换reflect可以化简车辆方向盘左右转，逆向变换backwards同时化简左右转向和前后行进方向。通过三个变换，最终可保留9个字段。推导及公式参考 REEDS和 SHEEPS 论文 OPTIMAL PATHS FOR A CAR THAT GOES BOTH。

1、基函数 LpSpLp（左前直前左前）





极坐标变换求解，当求解的圆弧段t和v均为正时，满足几何条件，此处字段为true，不满足则字段为false。后续字段依次类推，直到找到全部可行字段，选取最短路径即为所求ReedsSheeps轨迹。

function [isok,t,u,v] = LpSpLp(x,y,phi)

[t,u] = cart2pol(x-sin(phi),y-1+cos(phi));

if t >= 0

v = mod2pi(phi-t);

if v >= 0

isok = true;

return

end

end

isok = false;

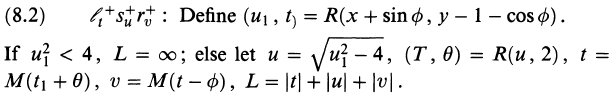
t = 0;

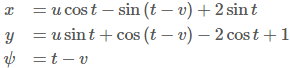
u = 0;

v = 0;

end

2、基函数 LpSpRp（左前直前右前）





极坐标变换求解，当求的解u1不满足几何条件时，此处字段为false。

function [isok,t,u,v] = LpSpRp(x,y,phi)

[t1,u1] = cart2pol(x+sin(phi),y-1-cos(phi));

if u1^2 >= 4

u = sqrt(u1^2-4);

theta = atan2(2,u);

t = mod2pi(t1+theta);

v = mod2pi(t-phi);

if t >= 0 && v >= 0

isok = true;

return

end

end

isok = false;

t = 0;

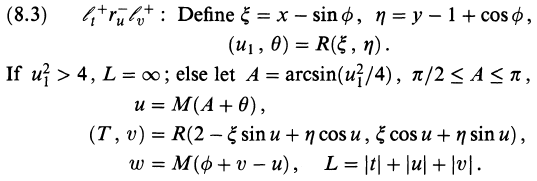
u = 0;

v = 0;

end

3、基函数 LpRmL

此处公式第一二字段LpRm为左前右后，最后一个字段合并，L应用于左转向前向后均可。



经反复推算及网上查证，论文中此处u1^2>4条件应该为u1>4,基函数代码如下所示，过程中用极坐标转换及几何关系验证得

function [isok,t,u,v] = LpRmL(x,y,phi)

xi = x-sin(phi);

eta = y-1+cos(phi);

[theta,u1] = cart2pol(xi,eta);

if u1 <= 4

u = -2\*asin(u1/4);

t = mod2pi(theta+u/2+pi);

v = mod2pi(phi-t+u);

if t >= 0 && u <= 0

isok = true;

return

end

end

isok = false;

t = 0;

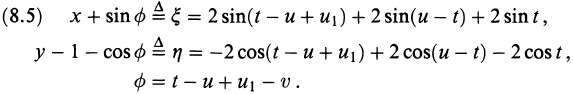
u = 0;

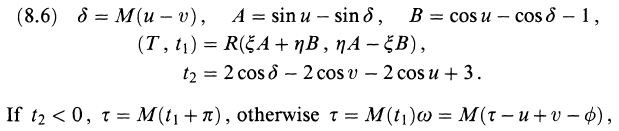
v = 0;

end

4、基函数 LpRupLumRm

此处公式中tauOmega并未全部推导出，主要参考论文中的公式；





function [tau,omega] = tauOmega(u,v,xi,eta,phi)

delta = mod2pi(u-v);

A = sin(u)-sin(delta);

B = cos(u)-cos(delta)-1;

t1 = atan2(eta\*A-xi\*B,xi\*A+eta\*B);

t2 = 2\*(cos(delta)-2\*cos(v)-2\*cos(u))+3;

if t2 < 0

tau = mod2pi(t1+pi);

else

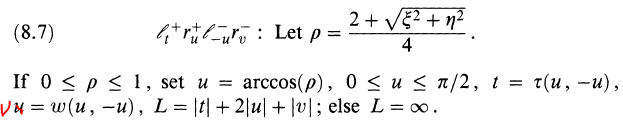
tau = mod2pi(t1);

end

omega = mod2pi(tau-u+v-phi);

end

求得tauOmega后，即可得基函数 LpRupLumRm



function [isok,t,u,v] = LpRupLumRm(x,y,phi)

xi = x+sin(phi);

eta = y-1-cos(phi);

rho = (2+sqrt(xi^2+eta^2))/4;

if rho <= 1

u = acos(rho);

[t,v] = tauOmega(u,-u,xi,eta,phi);

if t >= 0 && v <= 0

isok = true;

return

end

end

isok = false;

t = 0;

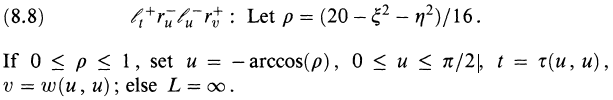
u = 0;

v = 0;

end

5、基函数 LpRumLumRp

由上可知tauOmega，公式如下，与上例中rho处计算不同，



function [isok,t,u,v] = LpRumLumRp(x,y,phi)

xi = x+sin(phi);

eta = y-1-cos(phi);

rho = (20-xi^2-eta^2)/16;

if rho >= 0 && rho <= 1

u = -acos(rho);

if u >= -pi/2

[t,v] = tauOmega(u,u,xi,eta,phi);

if t >=0 && v >=0

isok = true;

return

end

end

end

isok = false;

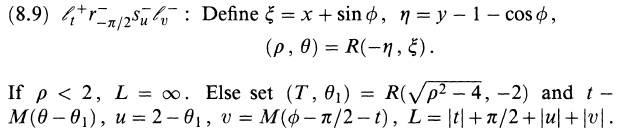
t = 0;

u = 0;

v = 0;

end

6、基函数 LpRmSmLm



function [isok,t,u,v] = LpRmSmLm(x,y,phi)

xi = x-sin(phi);

eta = y-1+cos(phi);

[theta,rho] = cart2pol(xi,eta);

if rho >= 2

r = sqrt(rho^2-4);

u = 2-r;

t = mod2pi(theta+atan2(r,-2));

v = mod2pi(phi-pi/2-t);

if t >= 0 && u <= 0 && v <= 0

isok = true;

return

end

end

isok = false;

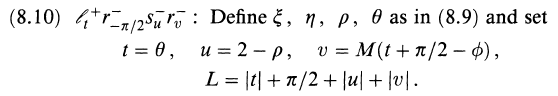
t = 0;

u = 0;

v = 0;

end

7、基函数 LpRmSmRm



function [isok,t,u,v] = LpRmSmRm(x,y,phi)

xi = x+sin(phi);

eta = y-1-cos(phi);

[theta,rho] = cart2pol(-eta,xi);

if rho >= 2

t = theta;

u = 2-rho;

v = mod2pi(t+pi/2-phi);

if t >= 0 && u <= 0 && v <= 0

isok = true;

return

end

end

isok = false;

t = 0;

u = 0;

v = 0;

end

8、组合函数及最短路径path

先定义含路径类型及长度的对象，通过三种变换组合，可以分类组合以上基函数形成5种word组合，CCC、CSC、CCCC、CCSC、CCSCC,所得5种path，最终选取最短path即为ReedsSheeps路径。对象及函数定义如下，

classdef RSPath

properties (Constant)

Types = [

'L', 'R', 'L', 'N', 'N' ; %1

'R', 'L', 'R', 'N', 'N' ; %2

'L', 'R', 'L', 'R', 'N' ; %3

'R', 'L', 'R', 'L', 'N' ; %4

'L', 'R', 'S', 'L', 'N' ; %5

'R', 'L', 'S', 'R', 'N' ; %6

'L', 'S', 'R', 'L', 'N' ; %7

'R', 'S', 'L', 'R', 'N' ; %8

'L', 'R', 'S', 'R', 'N' ; %9

'R', 'L', 'S', 'L', 'N' ; %10

'R', 'S', 'R', 'L', 'N' ; %11

'L', 'S', 'L', 'R', 'N' ; %12

'L', 'S', 'R', 'N', 'N' ; %13

'R', 'S', 'L', 'N', 'N' ; %14

'L', 'S', 'L', 'N', 'N' ; %15

'R', 'S', 'R', 'N', 'N' ; %16

'L', 'R', 'S', 'L', 'R' ; %17

'R', 'L', 'S', 'R', 'L' %18

];

end

properties

type = repmat('N',[1,5]);

t = 0;

u = 0;

v = 0;

w = 0;

x = 0;

totalLength = 0;

end

methods

function obj = RSPath(type,t,u,v,w,x)

obj.type = type;

obj.t = t;

obj.u = u;

obj.v = v;

obj.w = w;

obj.x = x;

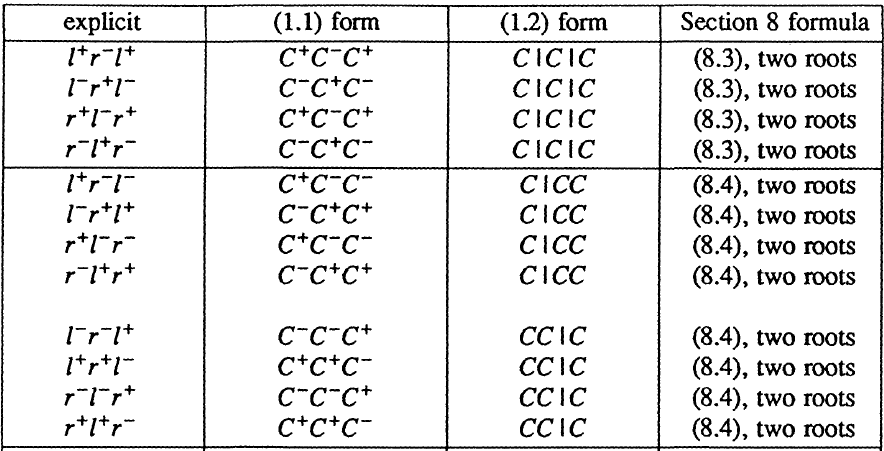
obj.totalLength = sum(abs([t,u,v,w,x]));

end

end

end

CCC字段函数由如下形式表示，



function [isok,path] = CCC(x,y,phi)

Lmin = inf;

type = repmat('N',[1,5]);

path = RSPath(type,0,0,0,0,0);

[isok,t,u,v] = LpRmL(x,y,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(1,:),t,u,v,0,0);

end

end

[isok,t,u,v] = LpRmL(-x,y,-phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(1,:),-t,-u,-v,0,0);

end

end

[isok,t,u,v] = LpRmL(x,-y,-phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(2,:),t,u,v,0,0);

end

end

[isok,t,u,v] = LpRmL(-x,-y,phi); % timeflip + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(2,:),-t,-u,-v,0,0);

end

end

% backwards

xb = x\*cos(phi)+y\*sin(phi); %

yb = x\*sin(phi)-y\*cos(phi); %LpRpL

[isok,t,u,v] = LpRmL(xb,yb,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(1,:),v,u,t,0,0);

end

end

[isok,t,u,v] = LpRmL(-xb,yb,-phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(1,:),-v,-u,-t,0,0);

end

end

[isok,t,u,v] = LpRmL(xb,-yb,-phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(2,:),v,u,t,0,0);

end

end

[isok,t,u,v] = LpRmL(-xb,-yb,phi); % timeflip + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(2,:),-v,-u,-t,0,0);

end

end

if Lmin == inf

isok = false;

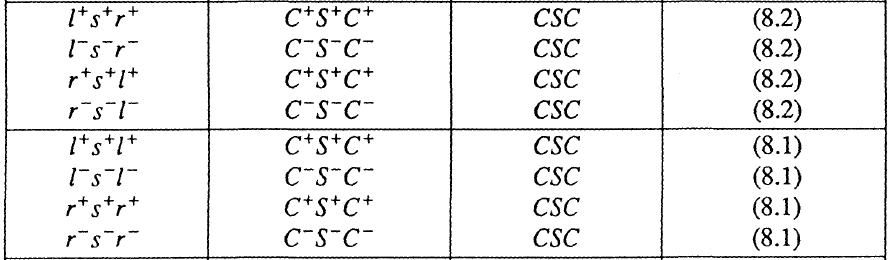
else

isok = true;

end

end

CSC字段函数由如下形式表示，



function [isok,path] = CSC(x,y,phi)

Lmin = inf;

type = repmat('N',[1,5]);

path = RSPath(type,0,0,0,0,0);

[isok,t,u,v] = LpSpLp(x,y,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(15,:),t,u,v,0,0);

end

end

[isok,t,u,v] = LpSpLp(-x,y,-phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(15,:),-t,-u,-v,0,0);

end

end

[isok,t,u,v] = LpSpLp(x,-y,-phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(16,:),t,u,v,0,0);

end

end

[isok,t,u,v] = LpSpLp(-x,-y,phi); % timeflp + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(16,:),-t,-u,-v,0,0);

end

end

[isok,t,u,v] = LpSpRp(x,y,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(13,:),t,u,v,0,0);

end

end

[isok,t,u,v] = LpSpRp(-x,y,-phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(13,:),-t,-u,-v,0,0);

end

end

[isok,t,u,v] = LpSpRp(x,-y,-phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(14,:),t,u,v,0,0);

end

end

[isok,t,u,v] = LpSpRp(-x,-y,phi); % timeflip + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(14,:),-t,-u,-v,0,0);

end

end

if Lmin == inf

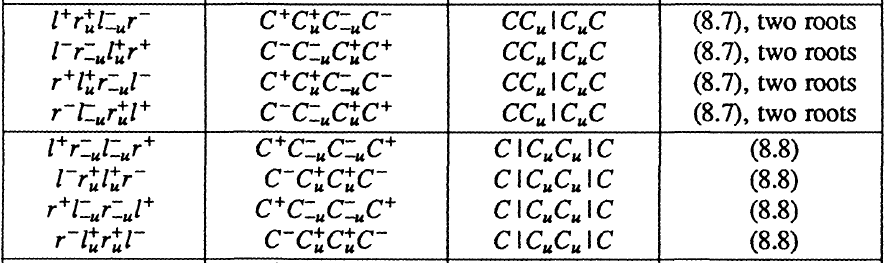
isok = false;

else

isok = true;

end

end

CCCC字段函数由如下形式表示，

function [isok,path] = CCCC(x,y,phi)

Lmin = inf;

type = repmat('N',[1,5]);

path = RSPath(type,0,0,0,0,0);

[isok,t,u,v] = LpRupLumRm(x,y,phi);

if isok

L = abs(t)+2\*abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(3,:),t,u,-u,v,0);

end

end

[isok,t,u,v] = LpRupLumRm(-x,y,-phi); % timeflip

if isok

L = abs(t)+2\*abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(3,:),-t,-u,u,-v,0);

end

end

[isok,t,u,v] = LpRupLumRm(x,-y,-phi); % reflect

if isok

L = abs(t)+2\*abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(4,:),t,u,-u,v,0);

end

end

[isok,t,u,v] = LpRupLumRm(-x,-y,phi); % timeflip + reflect

if isok

L = abs(t)+2\*abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(4,:),-t,-u,u,-v,0);

end

end

[isok,t,u,v] = LpRumLumRp(x,y,phi);

if isok

L = abs(t)+2\*abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(3,:),t,u,u,v,0);

end

end

[isok,t,u,v] = LpRumLumRp(-x,y,-phi); % timeflip

if isok

L = abs(t)+2\*abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(3,:),-t,-u,-u,-v,0);

end

end

[isok,t,u,v] = LpRumLumRp(x,-y,-phi); % reflect

if isok

L = abs(t)+2\*abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(4,:),t,u,u,v,0);

end

end

[isok,t,u,v] = LpRumLumRp(-x,-y,phi); % timeflip + reflect

if isok

L = abs(t)+2\*abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(4,:),-t,-u,-u,-v,0);

end

end

if Lmin == inf

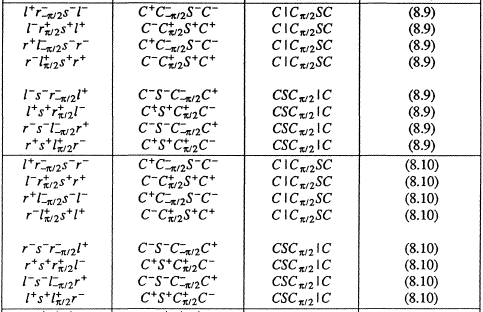
isok = false;

else

isok = true;

end

end

CCSC字段函数由如下形式表示，

function [isok,path] = CCSC(x,y,phi)

Lmin = inf;

type = repmat('N',[1,5]);

path = RSPath(type,0,0,0,0,0);

[isok,t,u,v] = LpRmSmLm(x,y,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(5,:),t,-pi/2,u,v,0);

end

end

[isok,t,u,v] = LpRmSmLm(-x,y,-phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(5,:),-t,pi/2,-u,-v,0);

end

end

[isok,t,u,v] = LpRmSmLm(x,-y,-phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(6,:),t,-pi/2,u,v,0);

end

end

[isok,t,u,v] = LpRmSmLm(-x,-y,phi); % timeflip + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(6,:),-t,pi/2,-u,-v,0);

end

end

[isok,t,u,v] = LpRmSmRm(x,y,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(9,:),t,-pi/2,u,v,0);

end

end

[isok,t,u,v] = LpRmSmRm(-x,y,-phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(9,:),-t,pi/2,-u,-v,0);

end

end

[isok,t,u,v] = LpRmSmRm(x,-y,-phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(10,:),t,-pi/2,u,v,0);

end

end

[isok,t,u,v] = LpRmSmRm(-x,-y,phi); % timeflip + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(10,:),-t,pi/2,-u,-v,0);

end

end

% backwards

此处经过推导当word字段倒序时，以下等式严格成立；

xb = x\*cos(phi)+y\*sin(phi);

yb = x\*sin(phi)-y\*cos(phi);

[isok,t,u,v] = LpRmSmLm(xb,yb,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(7,:),v,u,-pi/2,t,0);

end

end

[isok,t,u,v] = LpRmSmLm(-xb,yb,-phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(7,:),-v,-u,pi/2,-t,0);

end

end

[isok,t,u,v] = LpRmSmLm(xb,-yb,-phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(8,:),v,u,-pi/2,t,0);

end

end

[isok,t,u,v] = LpRmSmLm(-xb,-yb,phi); % timeflip + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(8,:),-v,-u,pi/2,-t,0);

end

end

[isok,t,u,v] = LpRmSmRm(xb,yb,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(11,:),v,u,-pi/2,t,0);

end

end

[isok,t,u,v] = LpRmSmRm(-xb,yb,-phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(11,:),-v,-u,pi/2,-t,0);

end

end

[isok,t,u,v] = LpRmSmRm(xb,-yb,-phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(12,:),v,u,-pi/2,t,0);

end

end

[isok,t,u,v] = LpRmSmRm(-xb,-yb,phi); % timeflip + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(12,:),-v,-u,pi/2,-t,0);

end

end

if Lmin == inf

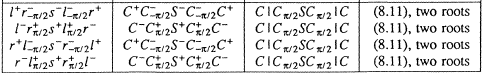
isok = false;

else

isok = true;

end

end

CCSCC字段函数由如下形式表示，

function [isok,path] = CCSCC(x,y,phi)

Lmin = inf;

type = repmat('N',[1,5]);

path = RSPath(type,0,0,0,0,0);

[isok,t,u,v] = LpRmSLmRp(x,y,phi);

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(17,:),t,-pi/2,u,-pi/2,v);

end

end

[isok,t,u,v] = LpRmSLmRp(x,y,phi); % timeflip

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(17,:),-t,pi/2,-u,pi/2,-v);

end

end

[isok,t,u,v] = LpRmSLmRp(x,y,phi); % reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(18,:),t,-pi/2,u,-pi/2,v);

end

end

[isok,t,u,v] = LpRmSLmRp(x,y,phi); % timeflip + reflect

if isok

L = abs(t)+abs(u)+abs(v);

if Lmin > L

Lmin = L;

path = RSPath(RSPath.Types(18,:),-t,pi/2,-u,pi/2,-v);

end

end

if Lmin == inf

isok = false;

else

isok = true;

end

end

车辆任意点任意姿态做ReedsSheeps曲线，在计算的上述函数中，选取字段最短路径，FindRSPath函数如下所示

function path = FindRSPath(x,y,phi,veh)

rmin = veh.MIN\_CIRCLE; %minimum turning radius

x = x/rmin;

y = y/rmin;

[isok1,path1] = CSC(x,y,phi);

[isok2,path2] = CCC(x,y,phi);

[isok3,path3] = CCCC(x,y,phi);

[isok4,path4] = CCSC(x,y,phi);

[isok5,path5] = CCSCC(x,y,phi);

isoks = [isok1, isok2, isok3, isok4, isok5];

paths = {path1, path2, path3, path4, path5};

Lmin = inf;

for i = 1:5

if isoks(i) == true

elem = paths{i};

if Lmin > elem.totalLength

Lmin = elem.totalLength;

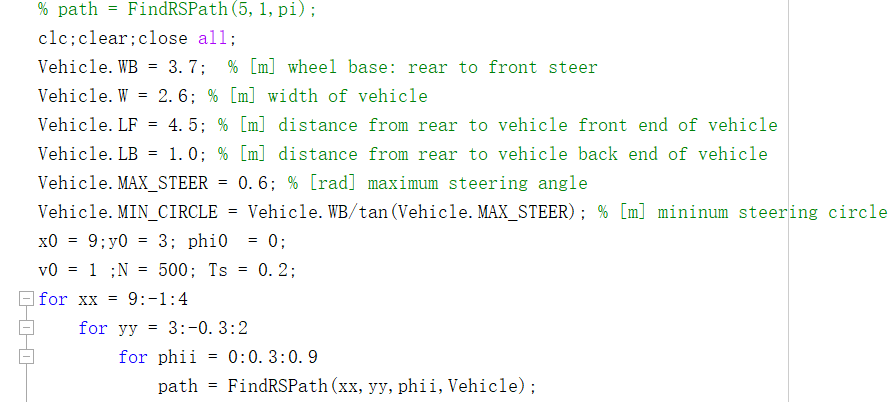
path = elem;

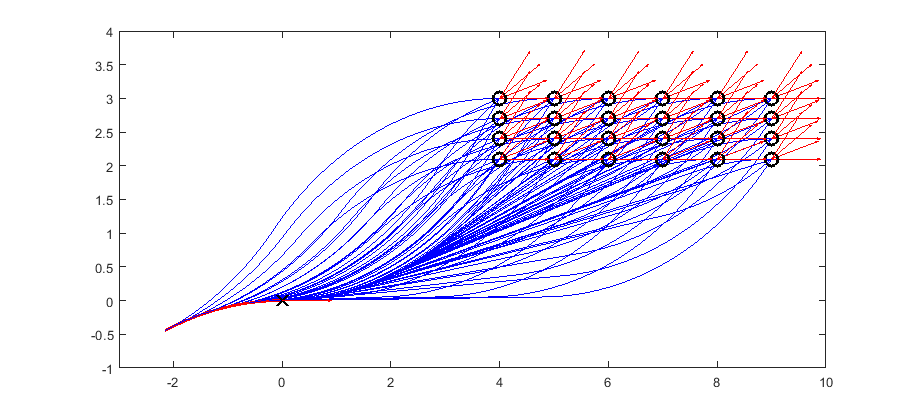
end

end

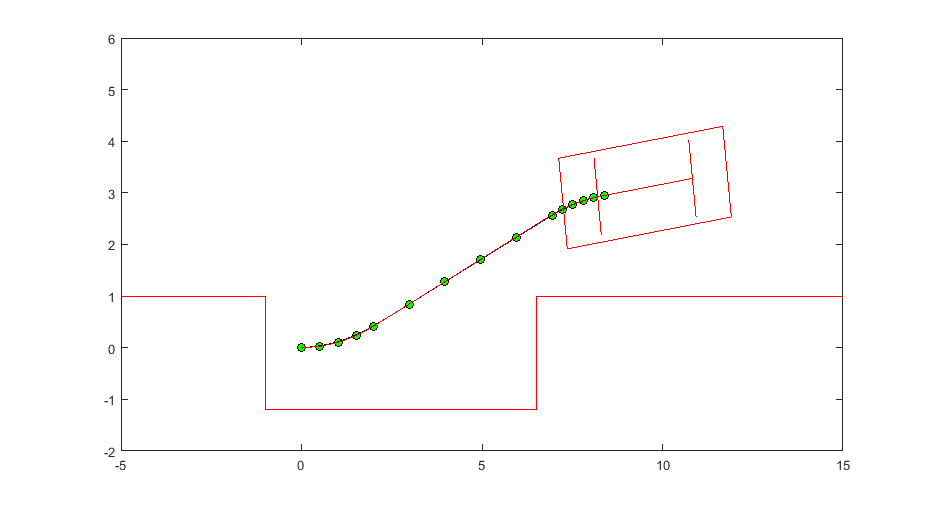
end

end

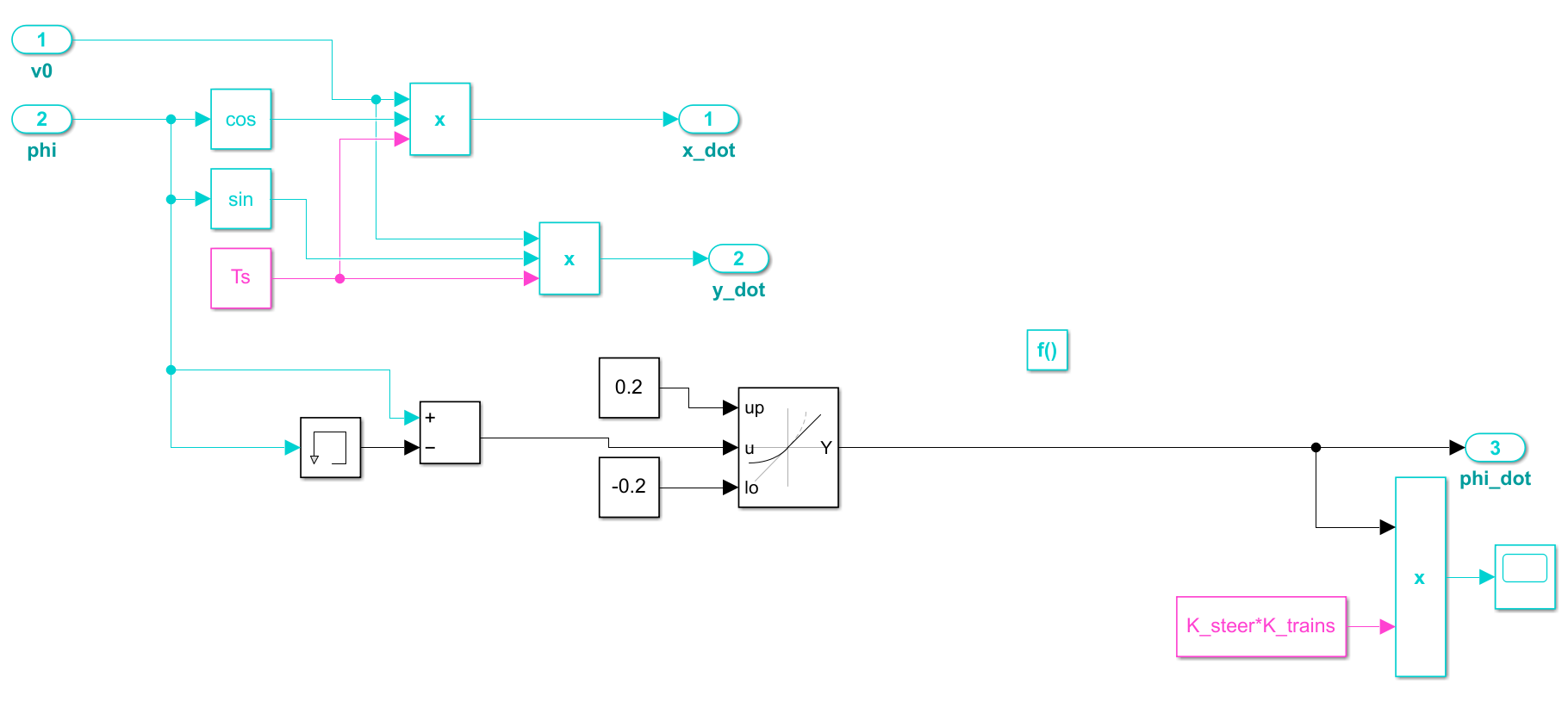
根据上述画任意轨迹点ReedsSheeps轨迹示例图，



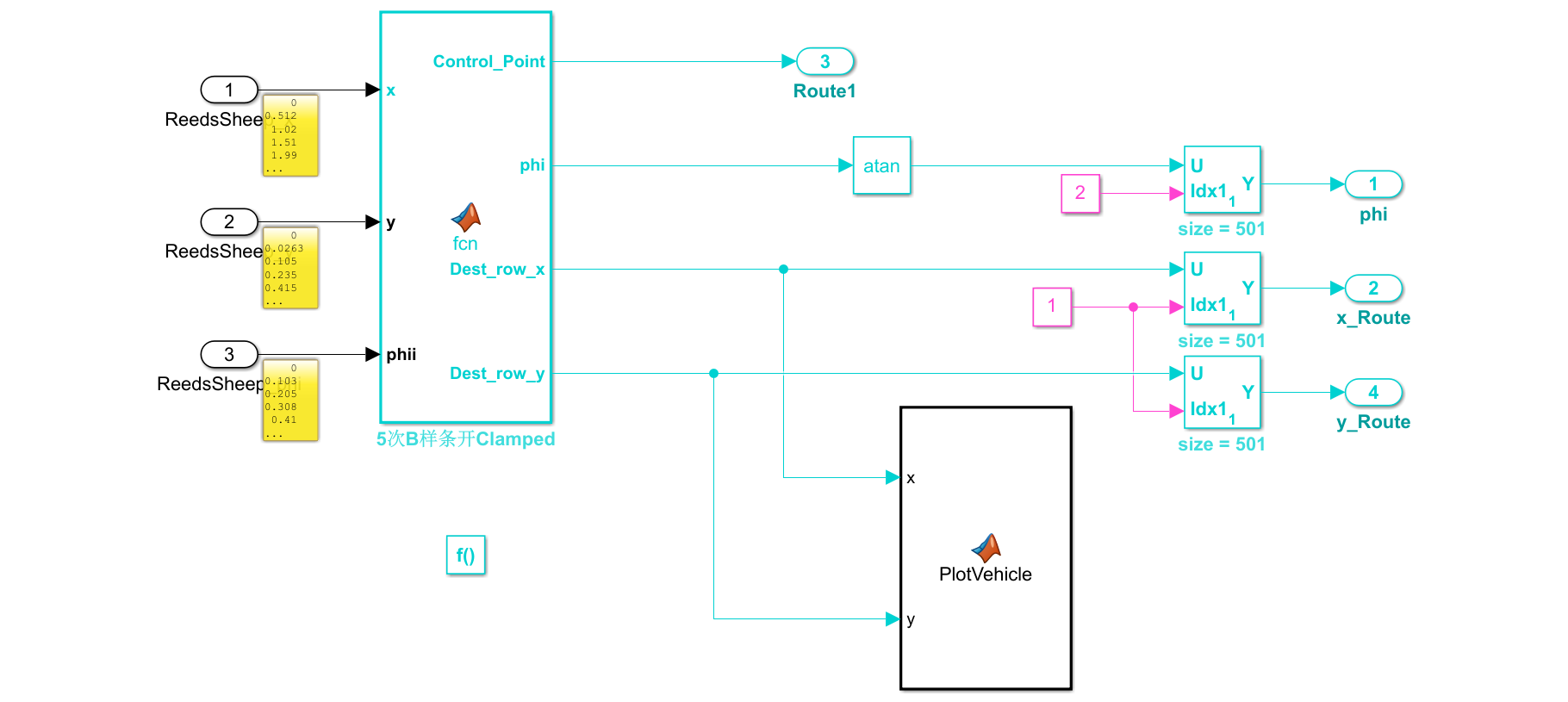
考虑到实际泊车场景，当车辆平行泊车泊入泊出时，修改ReedsSheeps中的word字段，减少计算量；在设计simulink模型中，将ReedsSheeps基函数减少，只在CSC字段中形成约束轨迹。此处简化了轨迹字段，只考虑最小转弯半径约束，没有考虑到障碍物约束；本文在前文中B样条曲线基础上，添加了ReedsSheeps算法处理，理论上有更好的适应性。在每一调度时刻，先计算ReedsSheeps轨迹，然后再输出5阶光滑B样条轨迹点。模型及结果如下所示



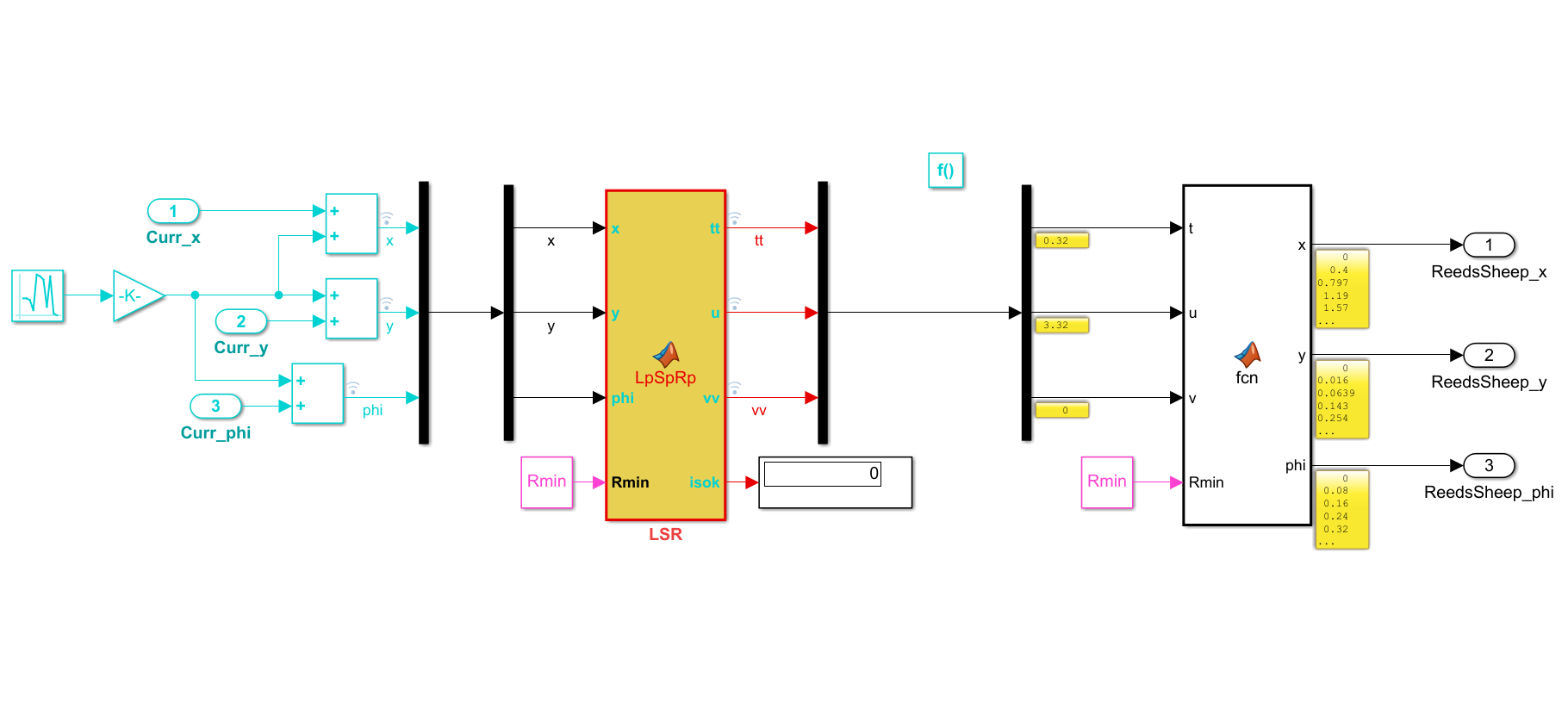
过程中车辆运动学模型没考虑实时性，参数用自定义参数代替仿真



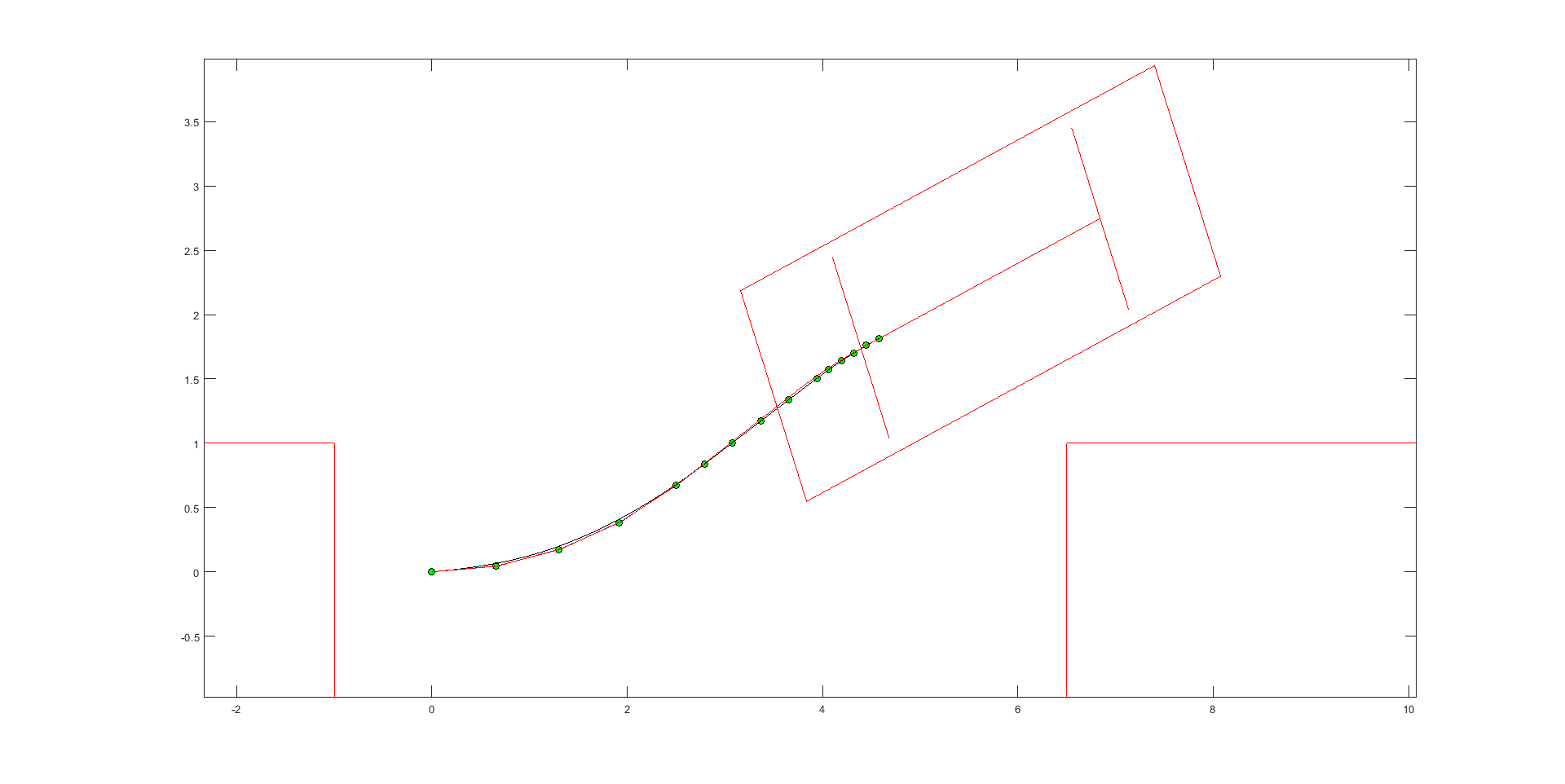
样条算法模型参考之前B样条插值算法

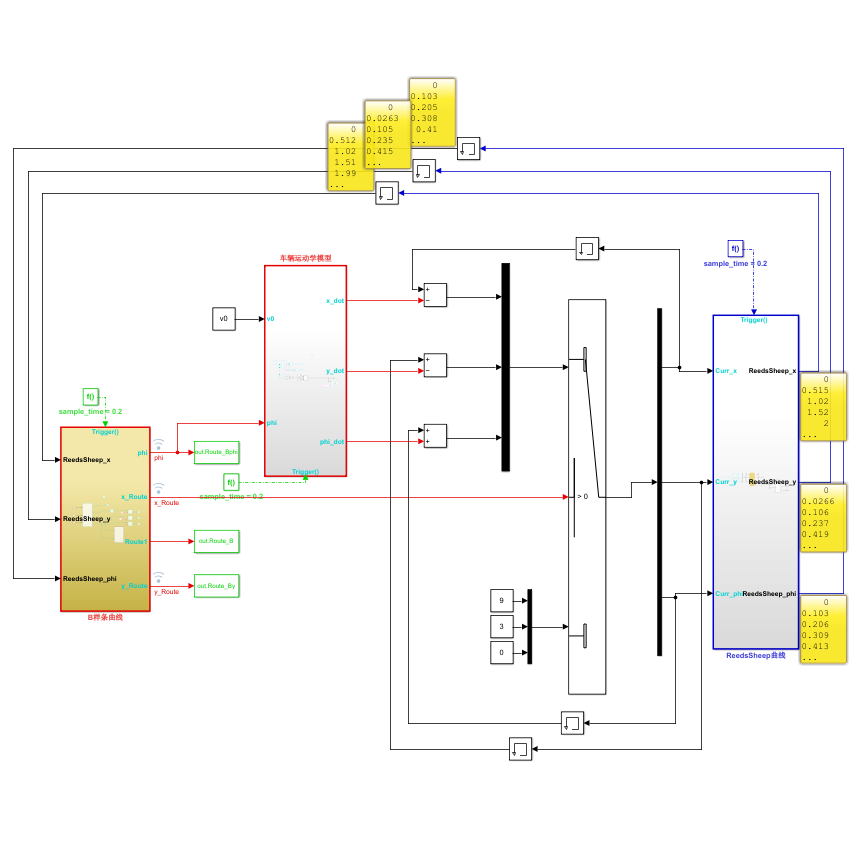


ReedsSheeps轨迹算法模型如下所示，所用字段简化为泊车场景使用；模型中，车位修正过程中，实时给车辆坐标及姿态添加小变量噪声误差。



模型及仿真结果如下，





Frenet运动规划算法

