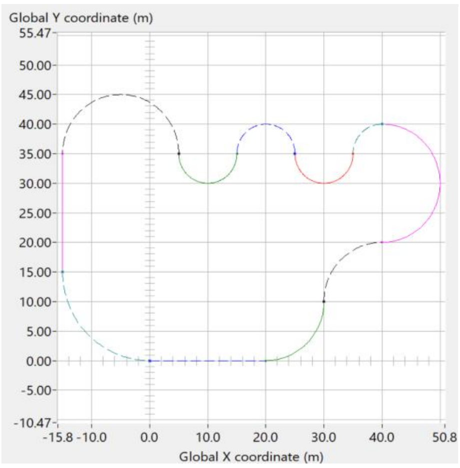
Lateral control of driverless vehicle (Carsim+Simulink)

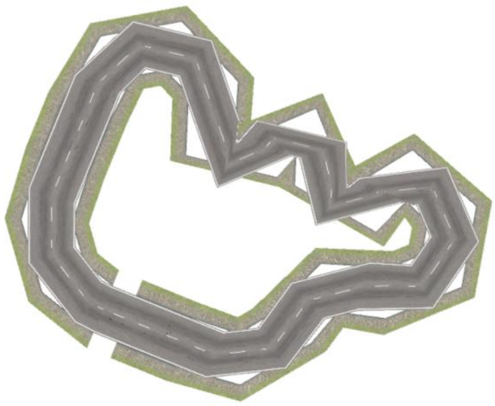
Shi Cheng

Table of contents:

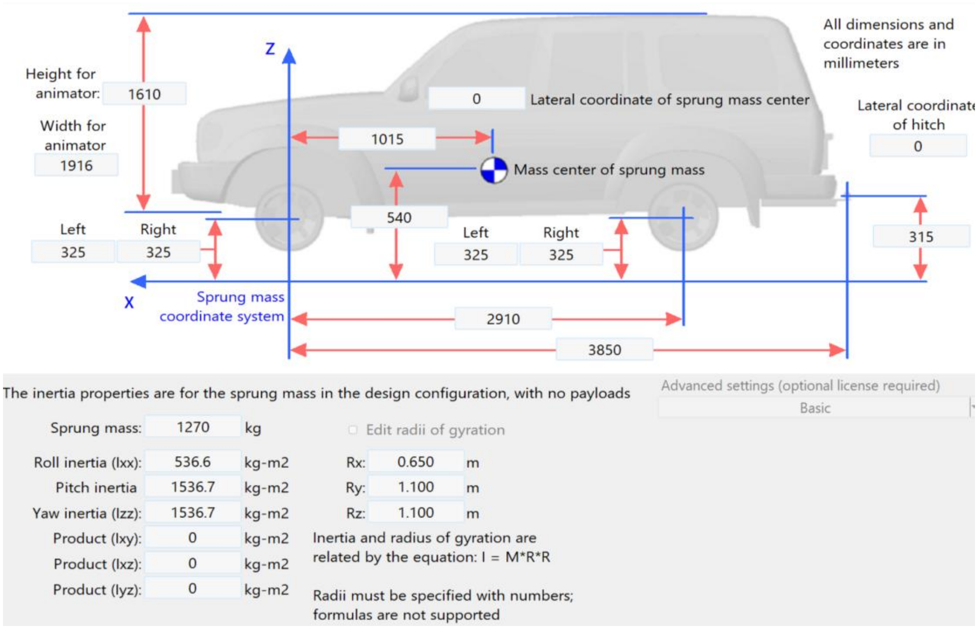
1. Simulation settings

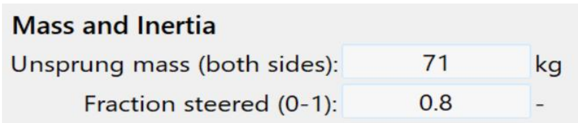
1.1 CarSim road setting and simulation



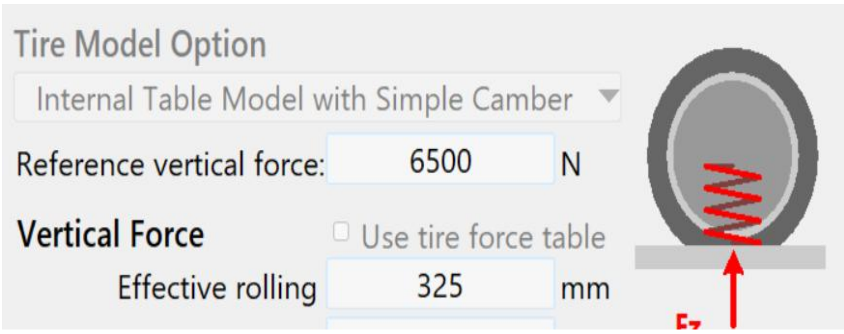


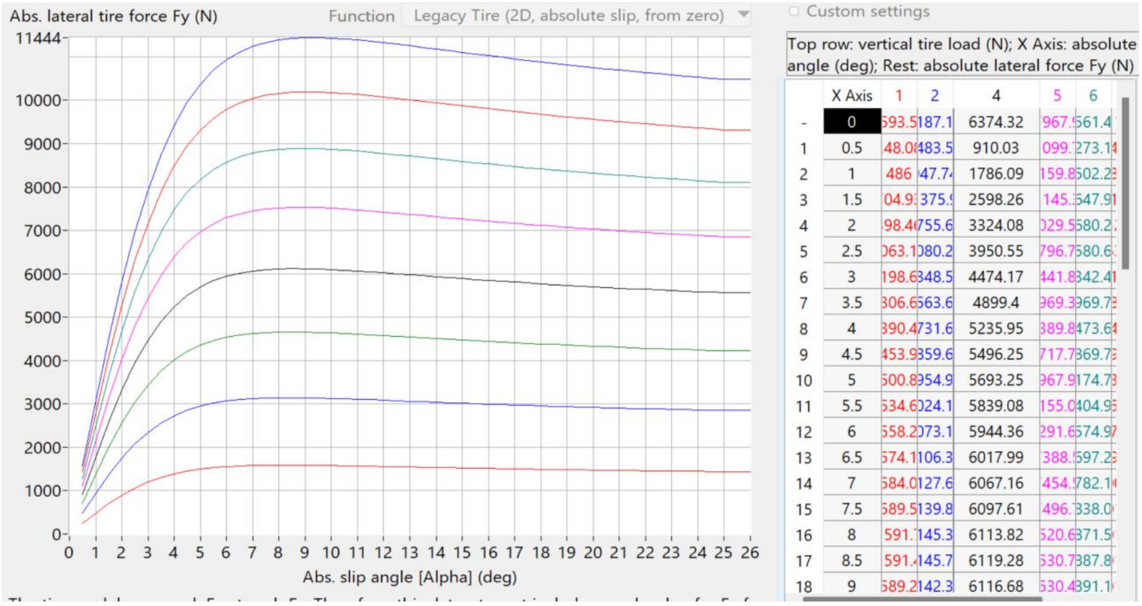
1.2 Select the correlation coefficient of the simulation vehicle - the distance from the mass point to the front wheel, the distance from the mass point to the rear wheel, and the weight of the whole vehicle (sprung mass + unsprung mass), moment of inertia.



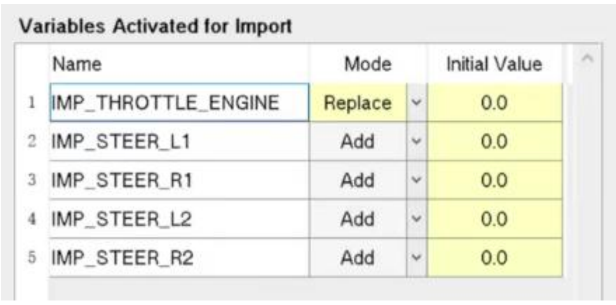


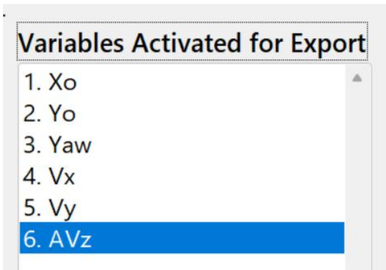
1.3 Idealized selection and estimation of tire cornering stiffness for simulation vehicle

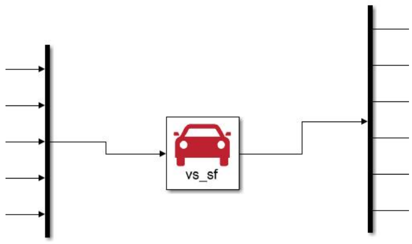




1.4 Input and output settings of vehicle model in CarSim in Simulink

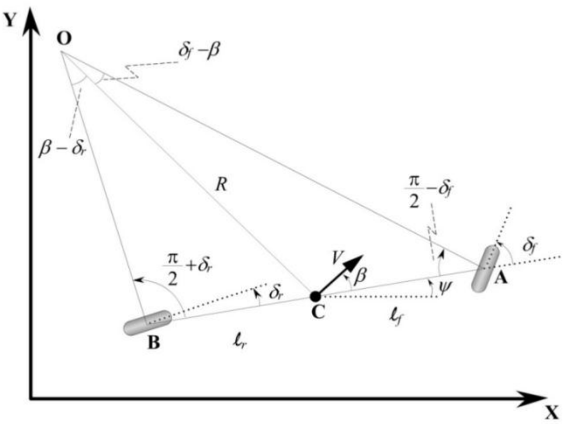






2. Algorithm

2.1 Tire cornering stiffness and vehicle dynamics equation



Idealized bicycle model of automobile

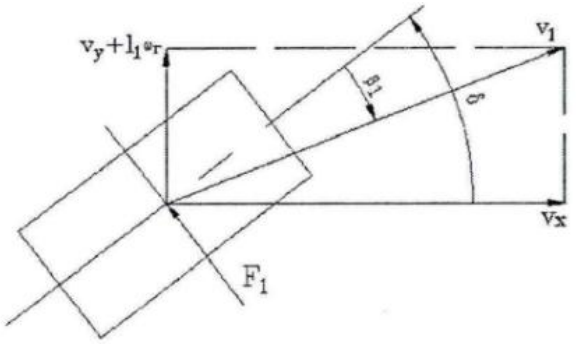
Symbol	Nomenclature	Equation
X	Global X axis coordinate	$\dot{X}=V\cos(\psi+\beta)$
Y	Global Y axis coordinate	$\dot{Y}=V\sin(\psi+\beta)$
ψ	Yaw angle; orientation angle of vehicle with respect to global X axis	$\dot{\psi}=\frac{V\cos(\beta)}{\ell_f+\ell_r}\left(\tan(\delta_f)-\tan(\delta_r)\right)$

$$\dot{X}=V\cos\psi$$

$$\dot{Y}=V\sin\psi$$

$$\dot{\psi}=\frac{V\tan\delta_f}{L}\left(L=l_f+l_r\right)$$

When driving at low speed, it will not slide by default. If vy = 0 by default β= 0 and the rear wheels do not turn δ Simplified mathematical model of R = 0.



Lateral force of tire F = C* α （C is lateral stiffness）

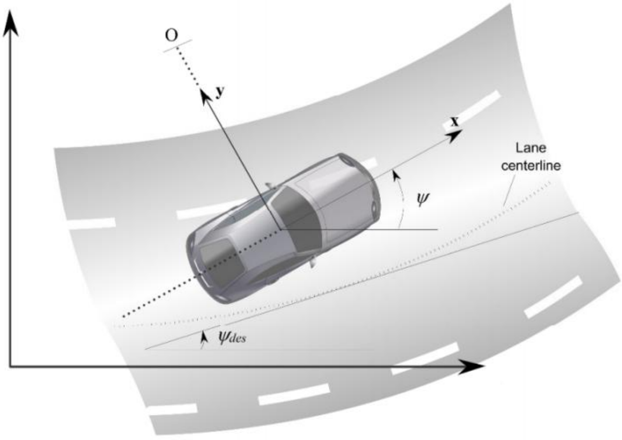
$$m(\ddot{y}+\dot{\psi}V_x)=F_{yf}+F_{yr}$$

$$I_z\ddot{\psi}=\ell_f F_{yf}-\ell_r F_{yr}$$

$$\frac{d}{dt}\begin{Bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{Bmatrix}=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & 0 & -V_x-\frac{2C_{\alpha f}\ell_f-2C_{\alpha r}\ell_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2\ell_f C_{\alpha f}-2\ell_r C_{\alpha r}}{I_z V_x} & 0 & -\frac{2\ell_f^2 C_{\alpha f}+2\ell_r^2 C_{\alpha r}}{I_z V_x} \end{bmatrix}\begin{Bmatrix} 0 \\ \dot{y} \\ 0 \\ \dot{\psi} \end{Bmatrix}+\begin{Bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2\ell_f C_{\alpha f}}{I_z} \end{Bmatrix}\delta$$

$$\dot{X}=Ax+Bu$$

2.2 Lateral error differential equation



$$\vec{a}_{inertial} = \vec{\Omega} \times (\vec{\Omega} \times \vec{r}) + \dot{\vec{\Omega}} \times \vec{r} + 2\vec{\Omega} \times \dot{\vec{r}} + \vec{a}_{body\_fixed}$$

or

$$\vec{a}_{inertial} = \dot{\psi}\hat{k} \times (\dot{\psi}\hat{k} \times -R\hat{j}) + \ddot{\psi}\hat{k} \times -R\hat{j} + 2\dot{\psi}\hat{k} \times -\dot{R}\hat{j} + \ddot{x}\hat{i} + \ddot{y}\hat{j}$$

or

$$\vec{a}_{inertial} = \dot{\psi}^2 R\,\hat{j} + (R\ddot{\psi} + 2\dot{\psi}\dot{R})\,\hat{i} + \ddot{x}\,\hat{i} + \ddot{y}\,\hat{j} \tag{2.36}$$

Hence $a_y = \dot{\psi}^2 R + \ddot{y} = V_x\dot{\psi} + \ddot{y}$.
Hence the inertial acceleration along the y axis is

$$a_y = \ddot{y} + V_x\dot{\psi} \tag{2.37}$$

Define $\ddot{e}_1$ and e_2 as follows (Guldner, et. al., 1996):

$$\ddot{e}_1 = (\ddot{y} + V_x\dot{\psi}) - \frac{V_x^{\,2}}{R} = \ddot{y} + V_x(\dot{\psi} - \dot{\psi}_{des}) \tag{2.40}$$

and

$$e_2 = \psi - \psi_{des} \tag{2.41}$$

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \dfrac{2C_{\alpha f}}{m} \\ 0 \\ \dfrac{2C_{\alpha f}\ell_f}{I_z} \end{bmatrix}\delta + \begin{bmatrix} 0 \\ -\dfrac{2C_{\alpha f}\ell_f - 2C_{\alpha r}\ell_r}{mV_x} - V_x \\ 0 \\ -\dfrac{2C_{\alpha f}\ell_f^{\,2} + 2C_{\alpha r}\ell_r^{\,2}}{I_z V_x} \end{bmatrix}\dot{\psi}_{des} + \begin{bmatrix} 0 \\ g \\ 0 \\ 0 \end{bmatrix}\sin(\phi)$$

$$+\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\dfrac{2C_{\alpha f} + 2C_{\alpha r}}{mV_x} & \dfrac{2C_{\alpha f} + 2C_{\alpha r}}{m} & \dfrac{-2C_{\alpha f}\ell_f + 2C_{\alpha r}\ell_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\dfrac{2C_{\alpha f}\ell_f - 2C_{\alpha r}\ell_r}{I_z V_x} & \dfrac{2C_{\alpha f}\ell_f - 2C_{\alpha r}\ell_r}{I_z} & -\dfrac{2C_{\alpha f}\ell_f^{\,2} + 2C_{\alpha r}\ell_r^{\,2}}{I_z V_x} \end{bmatrix}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix}$$

SUMMARY OF DYNAMIC MODEL EQUATIONS		
Symbol	Nomenclature	Equation
x	State space vector	$x = [e_1 \quad \dot{e}_1 \quad e_2 \quad \dot{e}_2]^T$
		$\dot{x} = Ax + B_1\delta + B_2\dot{\psi}_{des} + B_3\sin(\phi)$
		Matrices A, B_1, B_2 and B_3 are defined in equation (2.46)
e_1	Lateral position error with respect to road	$\ddot{e}_1 = \ddot{y} + V_x(\dot{\psi} - \dot{\psi}_{des})$
e_2	Yaw angle error with respect to road	$e_2 = (\psi - \psi_{des})$
δ	Front wheel steering angle	
$\dot{\psi}_{des}$	Desired yaw rate determined from road radius R	$\dot{\psi}_{des} = \frac{V_x}{R}$
ϕ	Bank angle with sign convention as defined by Fig. 2.8	

Idealized model without considering road slope $\Phi = 0$, $\dot{\psi}_{des}$ can also be ignored.

Then the governing equation of lateral error can be simplified:

$$\dot{x} = Ax + B_1\delta$$

2.3 Use LQR algorithm to reduce error

Here, the principle of DLQR and Lagrange multiplier method are used to find the optimal solution. Finally, u = - Ke feedback control is used.

$$\dot{x} = Ax + Bu \xrightarrow{discrete} x_{k+1} = \bar{A}x_k + \bar{B}u_k$$

Find the minimum value of $J = \sum_{k=0}^{+\infty}(x_k^T Q x_k + u_k^T R u_k)$ under the constraint of $x_{k+1} = \bar{A}x_k + \bar{B}u_k$

Integrate $\dot{x} = Ax + Bu$ then

Using midpoint Euler method for x $x(\xi) = \frac{x(t) + x(t+dt)}{2}$, using forward Euler method for u

$u(\xi) = u(t)$ (because we do not know $u(t+dt)$)

$$\left(I - \frac{Adt}{2}\right)x(t+dt) = \left(I + \frac{Adt}{2}\right)x(t) + Bdt\,u(t)$$

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

$$\approx \left(I - \frac{Adt}{2}\right)^{-1}\left(I + \frac{Adt}{2}\right)x(t) + Bdt\,u(t)$$

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

$$dt = 0.01 \qquad x(k+1) = \bar{A}x(k) + \bar{B}u(k)$$

$$\bar{A} = \left(I - \frac{Adt}{2}\right)^{-1}\left(I + \frac{Adt}{2}\right) \qquad \bar{B} = Bdt$$

LQR Summary:

For $\dot{e}_{rr} = Ae_{rr} + Bu$

1. Discretization $e_{rr}(k+1) = \bar{A}e_{rr}(k) + \bar{B}u(k)$

2. Solving Riccati equation $P = Q + \bar{A}^T P \bar{A} - \bar{A}^T P \bar{B}(R + \bar{B}^T P \bar{B})^{-1} \bar{B}^T P \bar{A}$

3. Solving $k = -(R + \bar{B}^T P \bar{B}^{-1})\bar{B}^T P \bar{A}$

4. Then $k = -(R + \bar{B}^T P \bar{B}^{-1})\bar{B}^T P \bar{A}$

2.4 Using feedforward control to reduce error

No matter what value K takes, the error and the derivative of the error cannot be 0 at the same time.

The feed-forward control is introduced to eliminate the steady-state error. LQR only makes the error err '= 0, and err still exists. Through the feed-forward control, the error is eliminated by making err as 0 as possible through the calculated heading angle.

We know $\dot{e}_{rr} = A e_{rr} + B(-k e_{rr} + \delta_f) + C\dot{\theta}_r$, After stabilization,

$$\dot{e}_{rr} = 0 \qquad e_{rr} = -(A - Bk)^{-1} \cdot (B\delta_f + C\dot{\theta}_r)$$

Our goal is to select the appropriate δ_f so that $e_{rr} = -(A - Bk)^{-1} \cdot (B\delta_f + C\dot{\theta}_r)$ is as 0 as possible.

$$e_{rr} = \begin{pmatrix} \frac{1}{k_1}\left\{\delta_f - \frac{\dot{\theta}_r}{v_x}\left[a + b - bk_3 - \frac{m v_x^2}{a+b}\left(\frac{b}{C_f} + \frac{a}{C_r}k_3 - \frac{a}{C_r}\right)\right]\right\} \\ 0 \\ -\frac{\dot{\theta}_r}{v_x}\left(b + \frac{a}{a+b}\frac{m v_x^2}{C_{\alpha r}}\right) \\ 0 \end{pmatrix}$$

We suppose

$$\delta_f = \frac{\dot{\theta}_r}{v_x}\left[a + b - bk_3 - \frac{m v_x^2}{a+b}\left(\frac{b}{C_f} + \frac{a}{C_r}k_3 - \frac{a}{C_r}\right)\right]$$

$$\dot{\theta}_r = k v_x$$

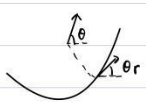
$$\delta_f = k\left[a + b - bk_3 - \frac{m v_x^2}{a+b}\left(\frac{b}{C_{\alpha f}} + \frac{a}{C_{\alpha r}}k_3 - \frac{a}{C_{\alpha r}}\right)\right]$$

Where $e_\varphi = -\frac{\dot{\theta}_r}{v_x}\left(b + \frac{a}{a+b}\frac{m v_x^2}{C_{\alpha r}}\right)$ is not affected by δ_f, k

It is found that the error of e_φ in err cannot be eliminated, and it is simplified after a series of ideal transformations.

e_φ is not heading error, $e_\varphi = \varphi - \theta_r$, The heading error is $\theta - \theta_r$ $\quad \theta = \varphi + \beta$

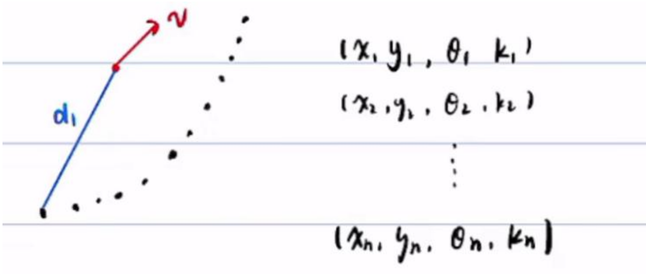
The steady-state error of e_φ is $-\beta$ $\quad e_\varphi = \varphi - \theta_r \Rightarrow -\beta = \varphi - \theta_r \quad \underline{\varphi + \beta = \theta_r}$ ✓



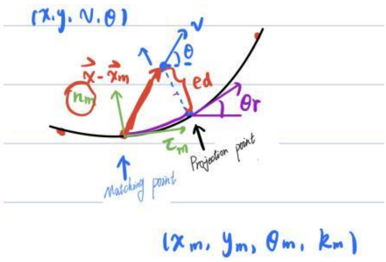
The ultimate goal is: θ-θr=0, which exactly eliminates the heading error, so this item does not need to be 0

2.5 Error of discrete programming trajectory

Error calculation of discrete trajectory points



We need to find the point closest to the real position (x, y) among the discrete planning trajectory points.



Suppose : match → The projection of k is invariant ⇒ match → The projected trajectory is approximately replace by an arc.

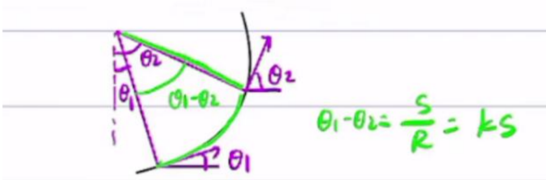
$\vec{\tau}_m = (\cos\theta_m, \sin\theta_m)$ $\vec{n}_m = (-\sin\theta_m, \cos\theta_m)$

$\vec{x} - \vec{x}_n = (x - x_m, y - y_m)$

③ $e_d \approx (\vec{x} - \vec{x}_n) \cdot \vec{n}_m$

④ $e_s \approx (x - \vec{x}_n) \cdot \vec{\tau}_m$

'es' is is the arc length between the matching point and the projection point.



$$e_d = (\vec{x} - \vec{x}_m) \cdot \vec{n}_m$$

$$e_s = (x - \vec{x}_m) \cdot \vec{\tau}_m$$

$$\theta_r = \theta_m + k_m \cdot e_s$$

$$k_r = k_m$$

$$\dot{s} = \frac{\vec{v}\cos(\theta - \theta_r)}{1 - k_r e_d}$$

$$e_\varphi = \varphi - \theta_r$$

$$\dot{e}_\varphi = \dot{\varphi} - \dot{\theta}_r = \varphi - k_r \cdot \dot{s}$$

$$\dot{e}_d = |\vec{v}| \sin(\theta - \theta_r)$$

2.6 Algorithm summary

Input in algorithm

1. Vehicle parameters : $a, b, C_{\alpha f}, C_{\alpha r}, m, I_z$

2. Vehicle position and statue : $x, y, \varphi, v_x, v_y, \dot{\varphi}$

3. Planning track points : $\begin{pmatrix} x_r \\ y_r \\ \theta_r \\ k_r \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & \cdots \\ y_1 & y_2 & \\ \theta_1 & \theta_2 & \\ k_1 & k_2 & \end{pmatrix}$

Output in algorithm

$u = -k e_{rr} + \delta_f$

Process

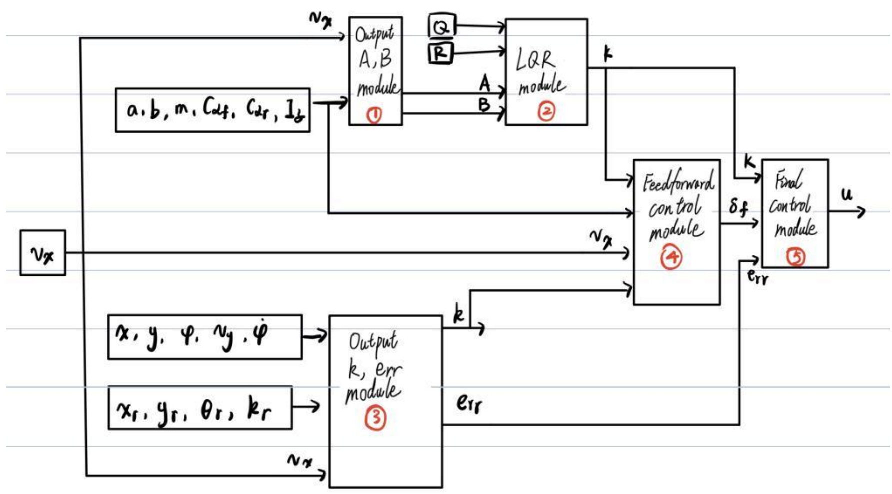
① Vehicle parameters + $v_x \Rightarrow A, B \xRightarrow[LQR]{Q,R} K$

② Vehicle position and statue + Planning track points $\Rightarrow$ k (Curvature of projection), e_{rr}

③ K in ① + k in ② + Vehicle parameters + $v_x \Rightarrow \delta_f$

④ K in ① + e_{rr} in ② + δ_f in ③ $\Rightarrow u = -Kx + \delta_f$

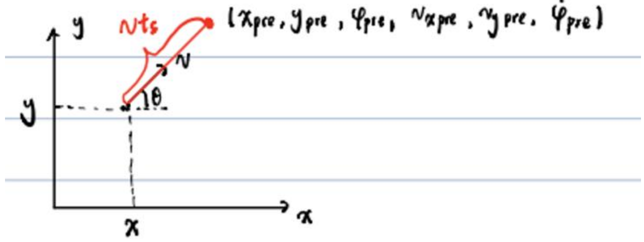
Module flow chart



3. Algorithm Optimization

3.1 Self-examination

When using the previous algorithm, it is easy to have problems. After our thinking, we finally get the optimization. We need to consider the difference between people and algorithmic control. If people drive, they know what the future path planning is, and they will turn the wheel according to the situation. If the vehicle is controlled by an algorithm, when the error in the direction is 0, the algorithm stops working, and the algorithm has hysteresis. In order to make the control more effective, we must add a prediction module.



3.2 Prediction module

The predicted time is recorded as 'ts'.

$$x_{pre} = x + vts\cos\theta = x + vts\cos(\beta+\varphi) = x + v_x ts\cos\varphi - v_y ts\sin\varphi$$

$$y_{pre} = y + vts\sin\theta = y + v_y ts\cos\varphi + v_x ts\sin\varphi$$

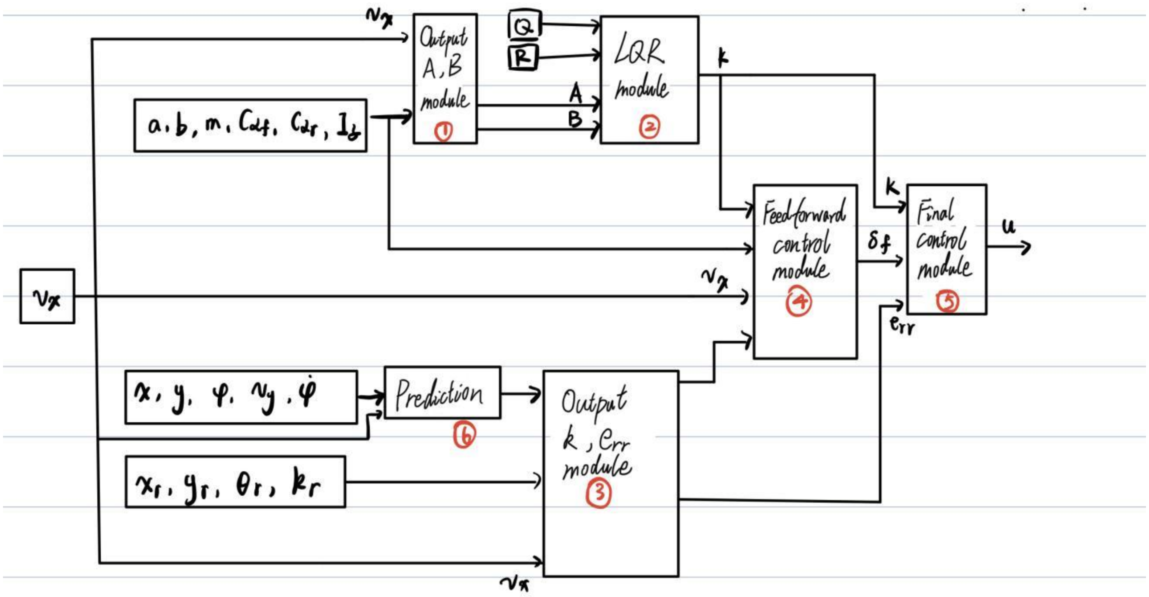
$$\varphi_{pre} = \varphi + \dot{\varphi} ts$$

$$v_{xpre} = v_x$$

$$v_{ypre} = v_y$$

$$\dot{\varphi}_{pre} = \dot{\varphi}$$

3.3 Optimized module flow chart



4. Code and Simulation

4.1 Code

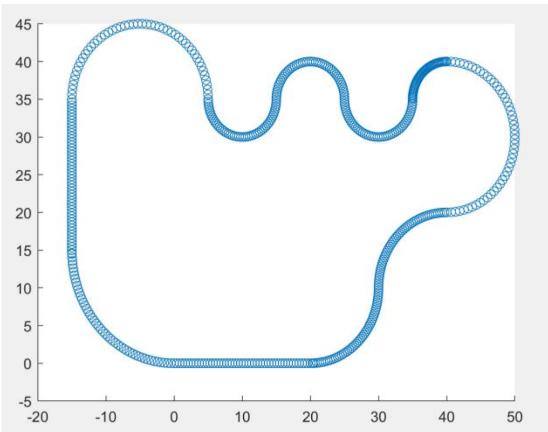
4.1.1 Routing Planning

Discrete predicted trajectory

```
count=50;
[x1,y1,theta1,kr1]=straight([0,0],[20,0],0,count);
[x2,y2,theta2,kr2]=arc([20,0],[30,10],0,pi/2,count);
[x3,y3,theta3,kr3]=arc([30,10],[40,20],pi/2,0,count);
[x4,y4,theta4,kr4]=arc([40,20],[40,40],0,pi,count);
[x5,y5,theta5,kr5]=arc([40,40],[35,35],pi,3*pi/2,count);
[x6,y6,theta6,kr6]=arc([35,35],[25,35],3*pi/2,pi/2,count);
[x7,y7,theta7,kr7]=arc([25,35],[15,35],pi/2,3*pi/2,count);
[x8,y8,theta8,kr8]=arc([15,35],[5,35],3*pi/2,pi/2,count);
[x9,y9,theta9,kr9]=arc([5,35],[-15,35],pi/2,3*pi/2,count);
[x10,y10,theta10,kr10]=straight([-15,35],[-15,15],3*pi/2,count);
[x11,y11,theta11,kr11]=arc([-15,15],[0,0],3*pi/2,2*pi,count);
xr=[x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11];
yr=[y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11];
thetar=[theta1,theta2,theta3,theta4,theta5,theta6,theta7,theta8,theta9,theta10,theta11];
kappar=[kr1,kr2,kr3,kr4,kr5,kr6,kr7,kr8,kr9,kr10,kr11];
```

```
function[xr,yr,thetar,kr]=arc(init_coord,end_coord,init_angle,end_angle,count)
    L=sqrt((init_coord(1)-end_coord(1))^2+(init_coord(2)-end_coord(2))^2);
    R=L/sqrt(2*(1-cos(end_angle-init_angle)));
    delta_angle=(end_angle-init_angle)/(count-1) ;

      for i=1:count
          if delta_angle>0
              xr(i)=init_coord(1)-R*sin(init_angle)+R*sin(init_angle+delta_angle*(i-1))
              yr(i)=init_coord(2)+R*cos(init_angle)-R*cos(init_angle+delta_angle*(i-1))
              thetar(i)=init_angle+delta_angle*i;
              kr(i)=1/R;
          else
              xr(i)=init_coord(1)+R*sin(init_angle)-R*sin(init_angle+delta_angle*(i-1))

              yr(i)=init_coord(2)-R*cos(init_angle)+R*cos(init_angle+delta_angle*(i-1))
              thetar(i)=init_angle+delta_angle*i;
              kr(i)=-1/R;
          end
      end
end
```

Then scatter(xr,yr), the following figure shows the image of discrete track points.



4.1.2 Output A, B Module and LQR Module

DLQR Offline data sheet and A, B date

```
cf=-110000;
cr=cf;
m=1412;
Iz=1536.7;
a=1.015;
b=2.910-1.015;
k=zeros(5000,4);
for i=1:5000
    vx=0.01*i;
    A=[0,1,0,0;
        0,(cf+cr)/(m*vx),-(cf+cr)/m,(a*cf-b*cr)/(m*vx);
        0,0,0,1;
        0,(a*cf-b*cr)/(Iz*vx),-(a*cf-b*cr)/Iz,(a*a*cf+b*b*cr)/(Iz*vx)];
    B=[0;
        -cf/m;
        0;
        -a*cf/Iz];
    Q=1*eye(4);
    R=10;
    k(i,:)=lqr(A,B,Q,R);
end
k1=k(:,1)';
k2=k(:,2)';
k3=k(:,3)';
k4=k(:,4)';
```

```
function k  = fcn(k1,k2,k3,k4,vx)
    if abs(vx)<0.01
        k=[0,0,0,0];
    else
        index=round(vx/0.01);
        k=[k1(index),k2(index),k3(index),k4(index)];
    end
end
```

4.1.3 Output k, err Module

```
XAfunction [kr,err] = fcn(x,y,phi,vx,vy,phi_dot,xr,yr,thetar,kappar)
    n=length(xr);
    d_min=(x-xr(1))^2+(y-yr(1))^2;
    min=1;
    for i=1:n
        d=(x-xr(i))^2+(y-yr(i))^2;
        if d<d_min
            d_min=d;
            min=i;
        end
    end
    dmin=min;
    tor=[cos(thetar(dmin));sin(thetar(dmin))];
    nor=[-sin(thetar(dmin));cos(thetar(dmin))];
    d_err=[x-xr(dmin);y-yr(dmin)];
    ed=nor'*d_err;
    es=tor'*d_err;
    %projection_point_thetar=thetar(dmin);%apollo
    projection_point_thetar=thetar(dmin)+kappar(dmin)*es;
    ed_dot=vy*cos(phi-projection_point_thetar)+vx*sin(phi-projection_point_thetar);
    %%%%%%%%%
    ephi=sin(phi-projection_point_thetar);
    %%%%%%%%%
    s_dot=vx*cos(phi-projection_point_thetar)-vy*sin(phi-projection_point_thetar);
    s_dot=s_dot/(1-kappar(dmin)*ed);
    ephi_dot=phi_dot-kappar(dmin)*s_dot;
    kr=kappar(dmin);
    err=[ed;ed_dot;ephi;ephi_dot];
```

4.1.4 Feedforward control Module

```matlab
function forword_angle = fcn(vx,a,b,m,cf,cr,k,kr)
    forword_angle=kr*(a+b-b*k(3)-(m*vx*vx/(a+b))*((b/cf)+(a/cr)*k(3)-(a/cr)));
end

```

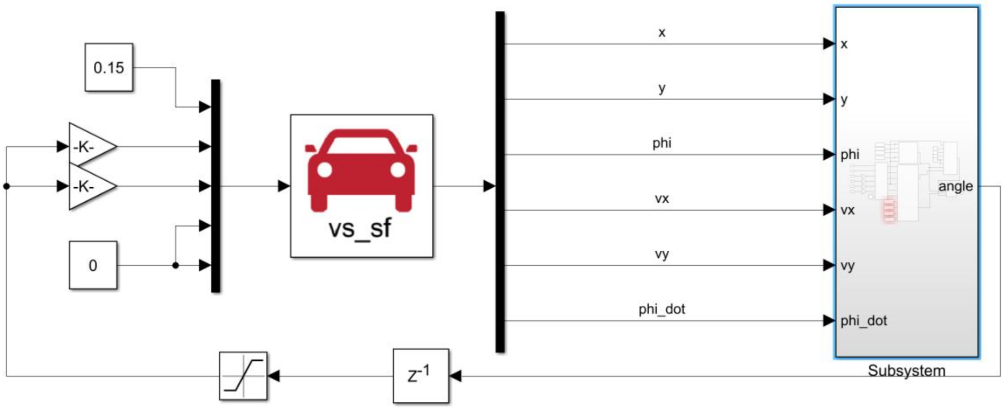
4.1.5 Final control Module

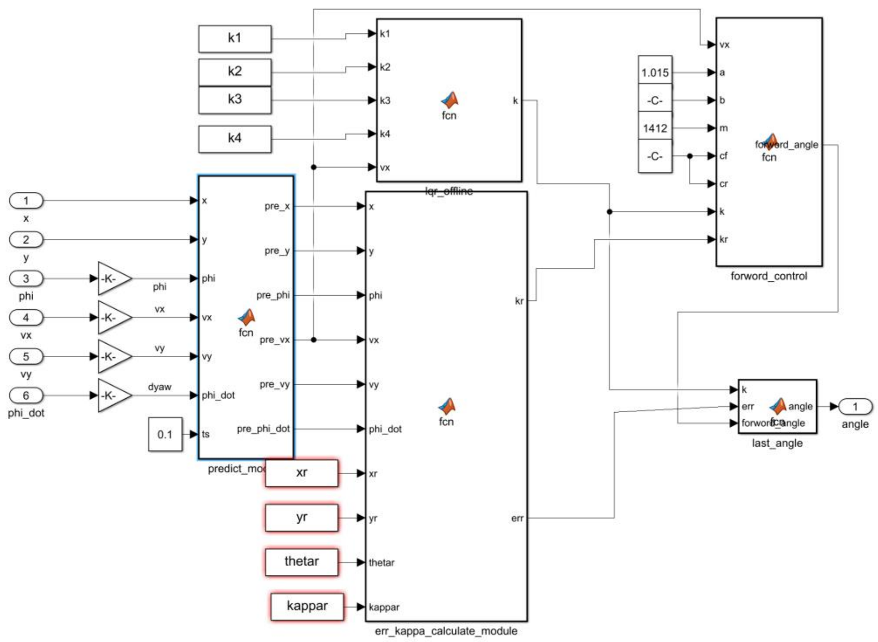
```matlab
function angle = fcn(k,err,forword_angle)

    angle=-k*err+forword_angle;
end

```

4.1.6 Prediction Module

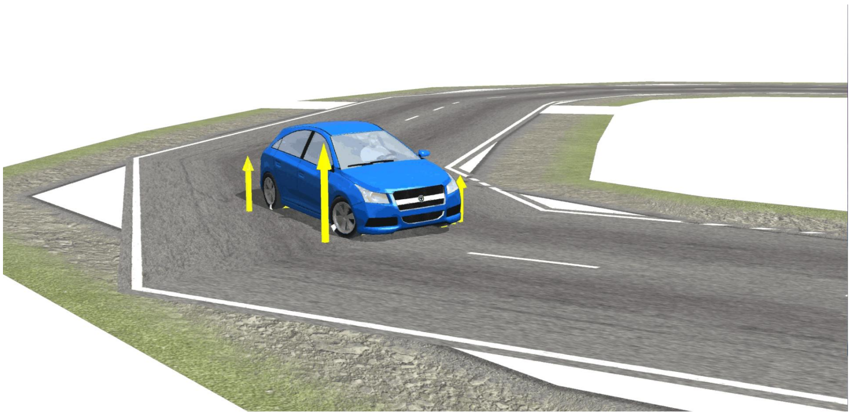
```matlab
function [pre_x,pre_y,pre_phi,pre_vx,pre_vy,pre_phi_dot] = fcn(x,y,phi,vx,vy,phi_dot,ts)
    pre_x=x+vx*ts*cos(phi)-vy*ts*sin(phi);
    pre_y=y+vy*ts*cos(phi)+vx*ts*sin(phi);
    pre_phi=phi+phi_dot*ts;
    pre_vx=vx;
    pre_vy=vy;
    pre_phi_dot=phi_dot;
end
```

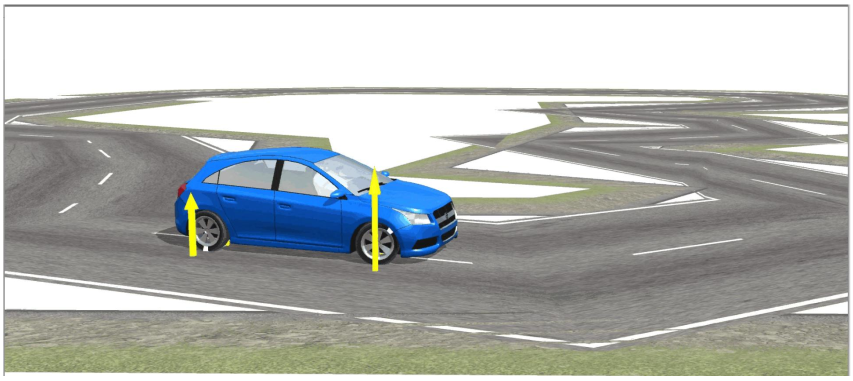
4.2 Simulink Module

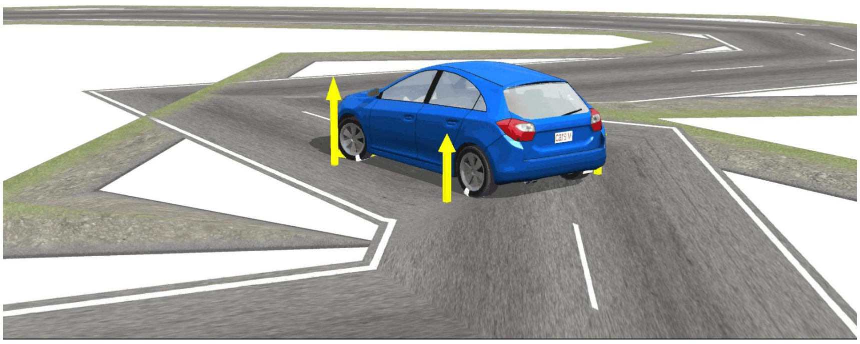


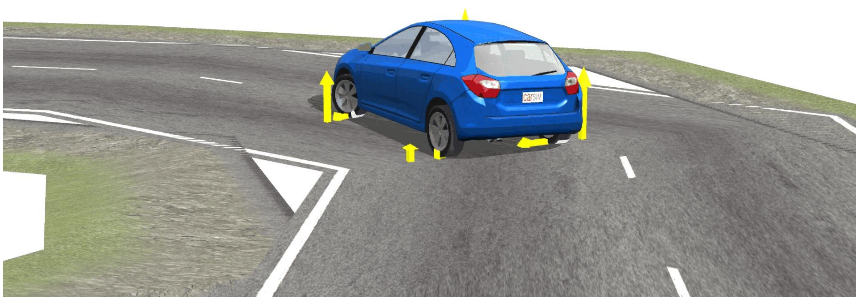


4.3 Carsim Simulation









5. Summary

After a series of efforts, it is finally realized that a specific vehicle runs on the specified road according to the preset track route. In this simulation, I learned how to use CarSim and Simulink for joint simulation, and also learned how to link modeling with code.