Matlab small project

Cheng Shi

Table of content:

1. Four population growth models

1.1 Exponential growth

Eq: $\begin{cases} \dfrac{dx}{dt} = rx \\ x(0) = x_0 \end{cases}$

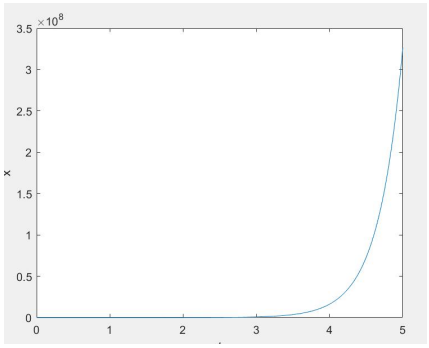
We can easily get $x(t) = x_0 e^{rt}$

Let r= 3 x_0 =100

Code:

```
function Exponential_growth

t=0:0.001:5;   % time scalex
initial_x=100;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt = 3*x;
    end
end
```

Figure:



1.2 Logistic growth

Eq: $\begin{cases} \dfrac{dx}{dt} = r(1-\dfrac{x}{K})x \\ x(0) = x_0 \end{cases}$

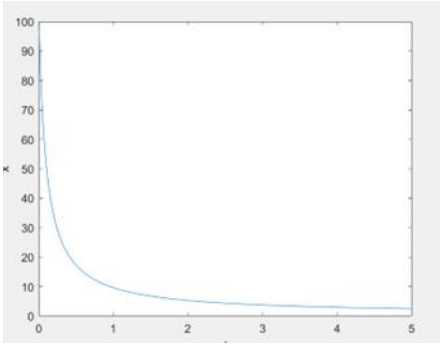
We can get $x(t) = \dfrac{K}{1+(\dfrac{K}{x_0}-1)\bullet e^{-rt}}$

Let r= 0.1 K= 1 x_0 =100

Code:

```
function Logistic_growth
t=0:0.001:5;   % time scalex
initial_x=100;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt = 0.1*x*(1-x);
    end
end
```

Figure:



1.3 Logistic growth with constant harvesting

Eq: $\begin{cases} \dfrac{dx}{dt} = r(1-\dfrac{x}{K})x\text{-}H \\ x(0) = x_0 \end{cases}$

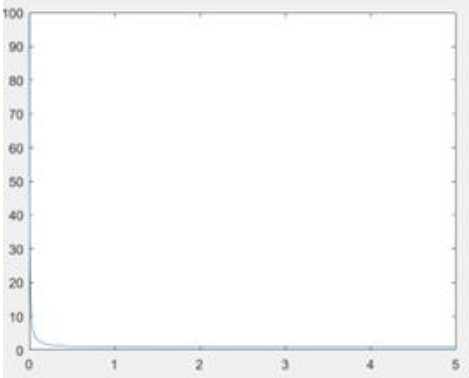
Let x_0 =100,We can get:

(1) r=4 K=1 H=0.5

Code:

```
function LGWCH1

t=0:0.001:5;   % time scalex
initial_x=100;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt = 4*x*(1-x)-0.5;
    end
end
```

Figure:

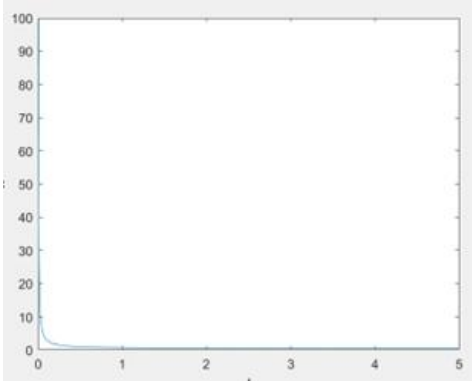


(2) r=4 K=1 H=1

Code:

```
function LGWCH2

t=0:0.001:5;   % time scalex
initial_x=100;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt = 4*x*(1-x)-1;
    end
end
```

Figure:

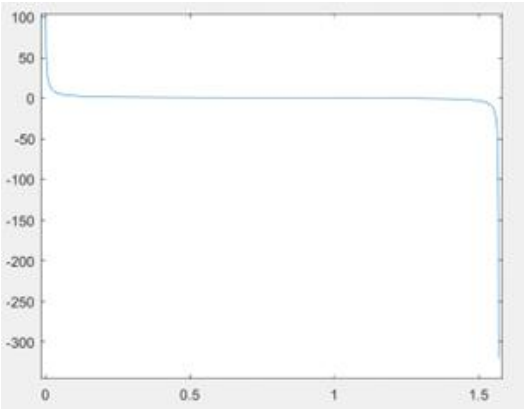


(3) r=4 K=1 H=2

Code:

```
function LGWCH3

t=0:0.001:5;   % time scalex
initial_x=100;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt = 4*x*(1-x)-2;
    end
end
```

Figure:



1.4 Logistic growth with population-dependent harvesting

Eq: $\begin{cases} \dfrac{dx}{dt} = r(1-\dfrac{x}{K})x - H\dfrac{x}{A+x} \\ x(0) = x_0 \end{cases}$

Let h=H/rK , a=A/K

Then we can get:

Eq: $$\begin{cases} \dfrac{dx}{dt} = (1\text{-}x)x\text{-h}\dfrac{x}{\text{a}+x} \\ x(0) = x_0 \end{cases}$$

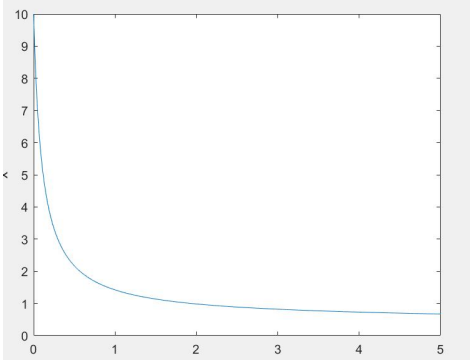
Let x_0 =10

(1) h=1 a=3

Code:

```
function LGWPDH1

t=0:0.001:5;   % time scalex
initial_x=10;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt =x*(1-x)-1/(3+x);
    end
end
```

Figure:

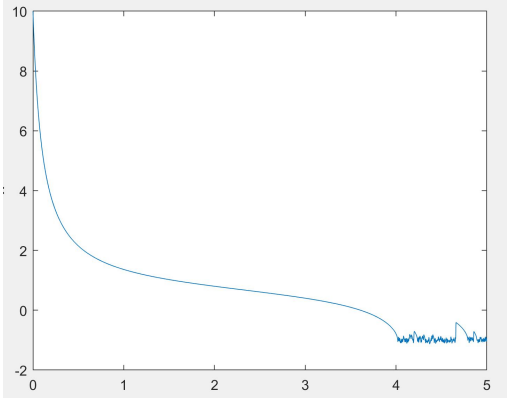


(2) h=1 a=1

Code:

```
function LGWPDH2

t=0:0.001:5;   % time scalex
initial_x=10;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt =x*(1-x)-1/(1+x);
    end
end
```

Figure:

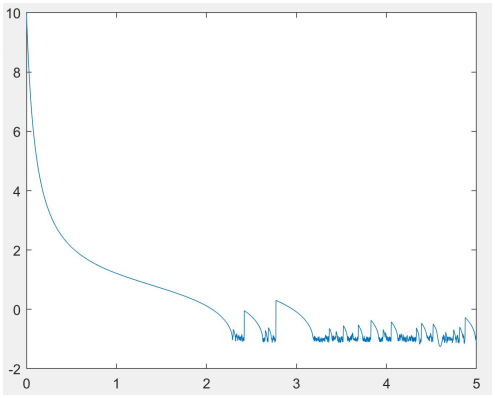


(3) h=2 a=1

Code:

```
function LGWPDH3

t=0:0.001:5;   % time scalex
initial_x=10;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt =x*(1-x)-2/(1+x);
    end
end
```

Figure:

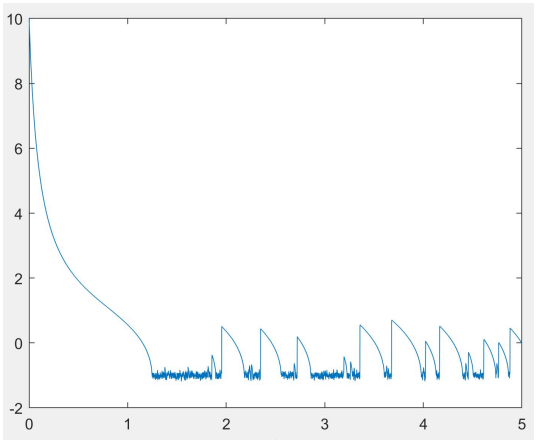


(4) h=5 a=1

Code:

```
function LGWPDH4

t=0:0.001:5;   % time scalex
initial_x=10;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt =x*(1-x)-5/(1+x);
    end
end
```

Figure:

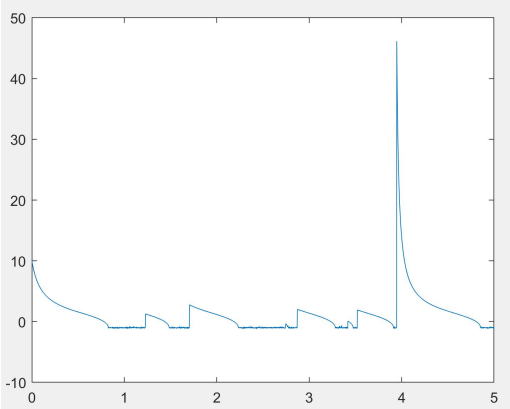


(5)h=10 a=1

Code:

```
function LGWPDH5

t=0:0.001:5;   % time scalex
initial_x=10;

[t,x]=ode45( @rhs, t, initial_x);

plot(t,x);
xlabel('t'); ylabel('x');

    function dxdt=rhs(t,x)
        dxdt =x*(1-x)-10/(1+x);
    end
end
```

Figure:



2. Fourier Series

Eq: f(t) = $\begin{cases} 0 & -\frac{T}{2} \le t < 0 \\ 1 & 0 \le t \le \frac{T}{2} \end{cases}$

We get $a_0 = \frac{1}{2}$ $a_{k=0}$ $b_k = \frac{1}{k\pi}(1 - \cos k\pi)$

the Fourier Series of the function $f(t) = \frac{1}{2} + \sum_{k=1}^{\infty} \frac{1}{k\pi}(1 - \cos k\pi) \sin k\frac{2\pi}{T} t$

let T=1 w0=2pi

Code:

```
% Fourier series of periodic square wave signals
t = -1:0.001:1;
w0 = 2 * pi;
y =max(square(pi*2*t,50),0);
%Periodic square wave signal
plot(t,y), grid on;
axis([-1 1 -1.5 1.5]);
n_max = [1 3 5 7 31];
N = length(n_max);
for k = 1:N
    n = 1:2:n_max(k);
    b = (1-cos(n*pi))./(pi*n);
    x =0.5+b *sin(w0*n'*t);
    figure;
    plot(t,y);
    hold on;
    plot(t,x);
    hold off;
    axis([-1 1 -1.5 1.5]),grid on;
    title(['Maximum harmonic number=',num2str(n_max(k))]);
end

```

Original figure:

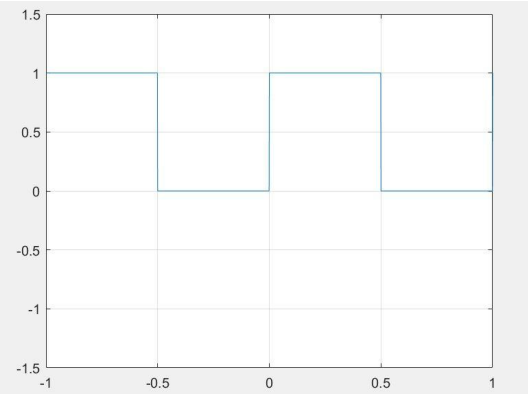
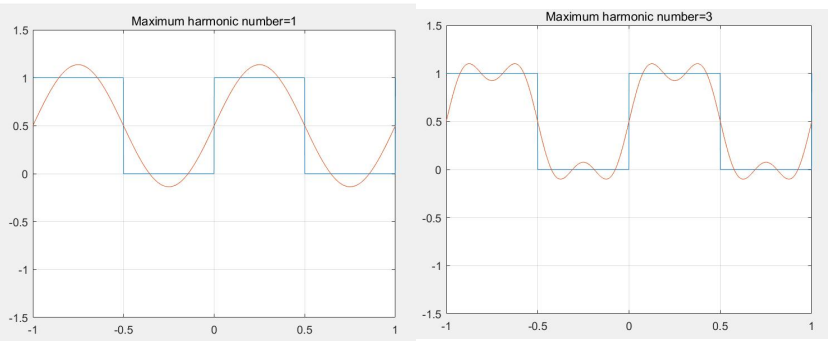
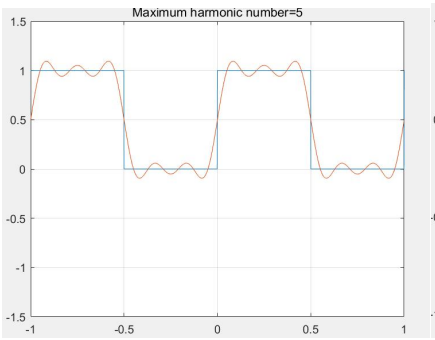
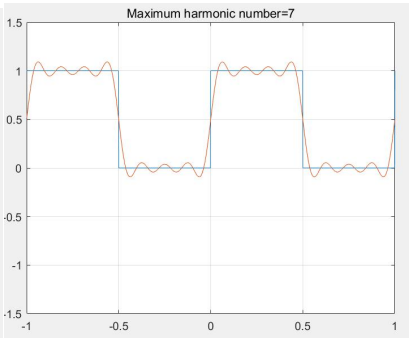
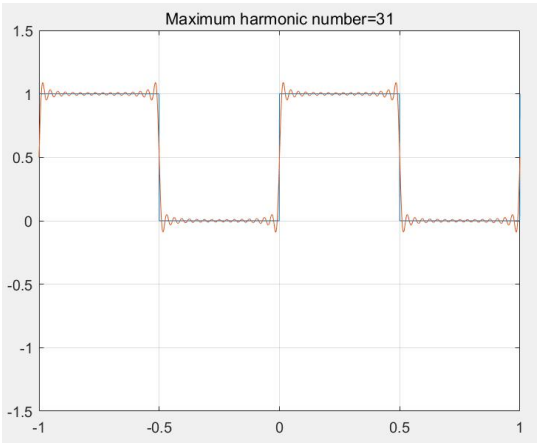


Figure with Fourier number n=1,3,5,7,31 :



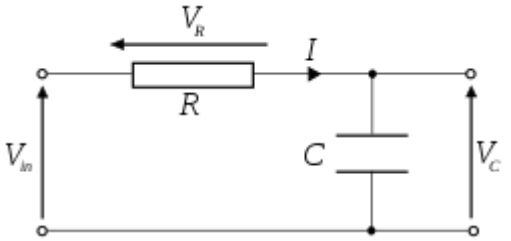






3. First order systems

RC circuits:



Eq:

$$\frac{dv_c}{dt} + \frac{1}{RC}v_c = \frac{1}{RC}v(t)$$

Solution:

$$v_c(t) = v_c(0)e^{-\frac{t}{RC}} + \frac{1}{RC}\int_0^t v(\tau)e^{-\frac{1}{RC}(t-\tau)}d\tau$$

Let R=1 C=1 Vi=1

Code:

```
%% impulse response, step response, ramp response, and frequency response:
% Impulse V(t) = dirac(t)*Vi
% Step:  V(t) = 1*Vi
% Ramp: V(t) = t*Vi
% Frequency: V(t) = exp(jwt)*Vi
%V(t)/(R*C)-Vc/(R*C)=dVc/dt
t = -1:0.001:1;
Vi = 1;
w  = 1;
R  = 1;
C  = 1;
% w  = logspace(1,4,20);
figure();
```

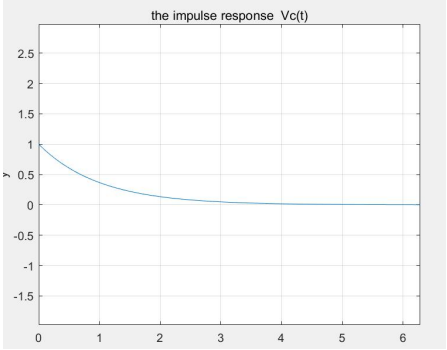
(1) impulse response

$$V(t)=\begin{cases}0 & t<0\\ \delta(t) & 0\le t\end{cases}$$

Code:

```
%% impulse response:
syms  t s
V =dirac(t)*Vi;
Vir = Vi;
Vcir = Vir/(s+1);
V1 = ilaplace(Vcir);
ezplot(t,V1);
title('the impulse response  Vc(t)')
grid on;

figure();
```

Figure:



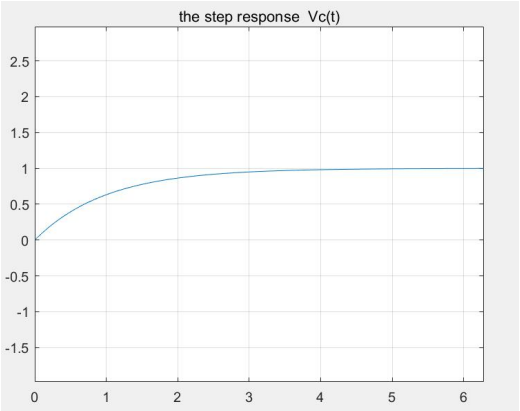
(2) step response

$$V(t)=\begin{cases}0 & t<0\\ 1 & 0\le t\end{cases}$$

Code:

```
%% step response:
syms  t s
V     = 1*Vi;
Vsr   = Vi/s;
Vcsr = Vsr/(s+1);
V2 = ilaplace(Vcsr);
ezplot(t,V2);
title('the step response  Vc(t)')
grid on;

figure();
```

Figure:



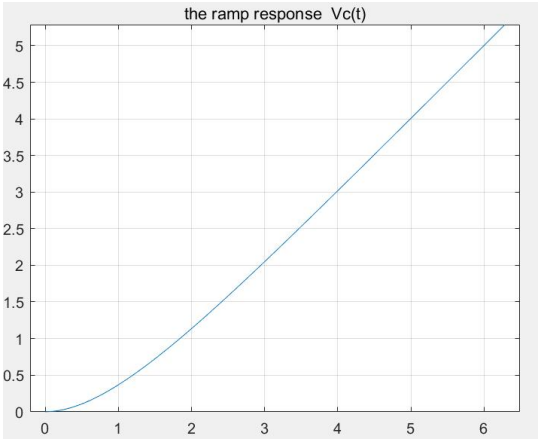
(3) ramp response,

$$V(t)=\begin{cases}0 & t<0\\ t & 0\le t\end{cases}$$

Code:

```
%% ramp response:
syms  t s
V     = Vi*t;
Vrr   = Vi/s^2;
Vcrr = Vrr/(s+1);
V3 = ilaplace(Vcrr);
ezplot(t,V3);
title('the ramp response  Vc(t)')
grid on;

figure();
```

Figure:



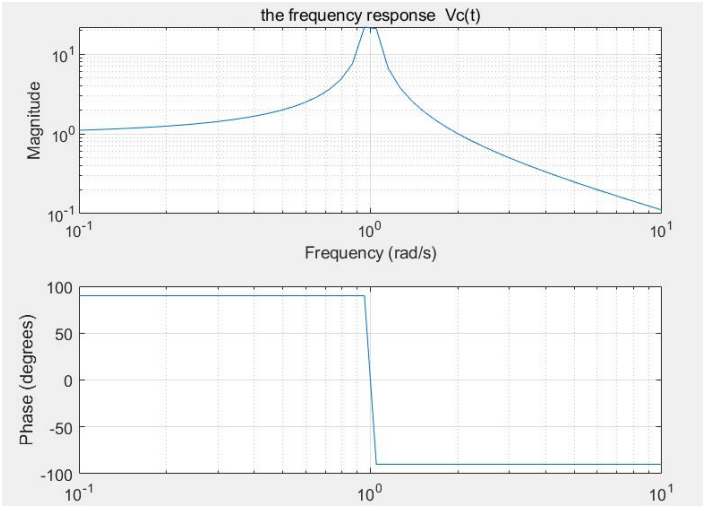
(4) frequency response

$$V(t)=\begin{cases}0 & t<0\\ e^{jw} & 0\le t\end{cases}$$

Code:

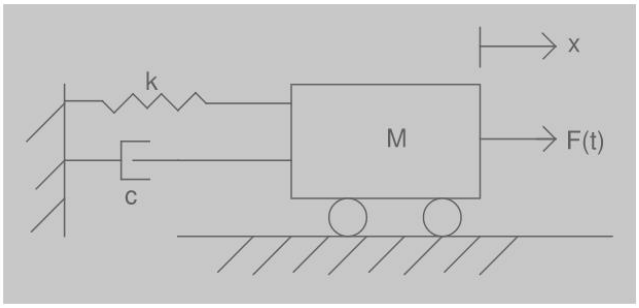
```
47      %% frequency response:
48 —    syms t s
49 —    V     = Vi.*exp(1i*w*t);
50 —    Vfr   = Vi*(s+1i)/(s^2+1);
51 —    Vcfr = Vfr/(s+1);
52 —    a = [1 0 1];
53 —    b = [0 1 1i];
54 —    w = logspace(-1,1);
55 —    freqs(b,a,w)
56 —    title('the frequency response  Vc(t)')
57 —    grid on;
```

Let w=1, then figure:



4. Second order systems

translational spring-mass-damper system:



We can get $m\ddot{x} + c\dot{x} + kx = F(t)$

$\ddot{x}+2\varepsilon\omega_n\dot{x}+\omega_n^2 x=f(t)$

where $2\varepsilon\omega_n=\frac{c}{m}$, $\omega_n^2 = \frac{k}{m}$,$f(t)=\frac{F(t)}{m}$

(1) $f(t)=\begin{cases}0 & t<0\\ F_0 & t\geq 0\end{cases}$

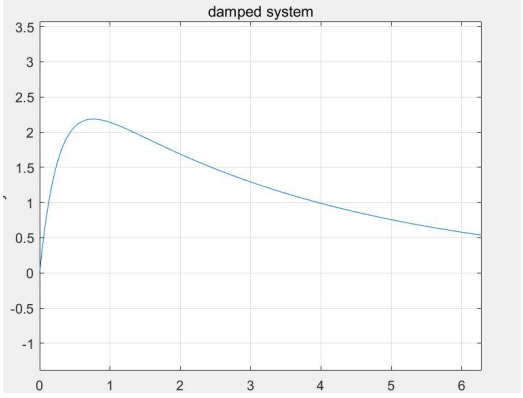
We choose F_0=10 $\omega_n = 1$

ε>1 damped system ε=2

Code:

```
%F=10 W=1f(t)=1*F a= ε
%% damped system:
syms  t s
a1=2;
X=10/(s^2+2*a1*s+1);
x = ilaplace(X);
ezplot(t,x);
title('damped system')
grid on;
figure();
```

Figure:

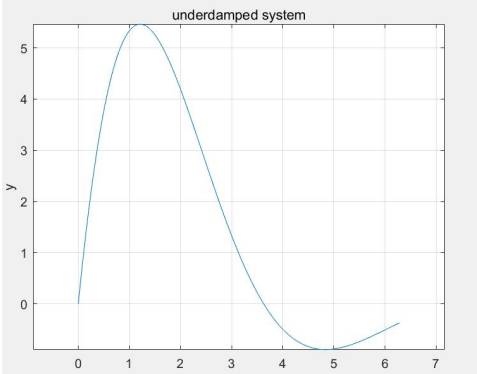


ε<1 underdamped system ε=0.5

Code:

```
%% underdamped system:
syms  t s
a2=0.5;
X=10/(s^2+2*a2*s+1);
x = ilaplace(X);
ezplot(t,x);
title('underdamped system')
grid on;
figure();
```

Figure:

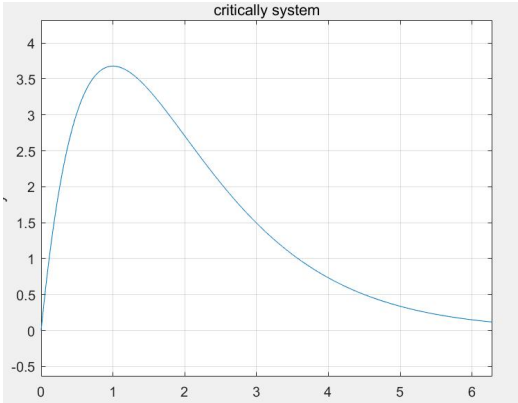


ε=1 critically damped system

Code:

```
%% critically system:
syms  t s
a3=1;
X=10/(s^2+2*a3*s+1);
x = ilaplace(X);
ezplot(t,x);
title('critically system')
grid on;
```

Figure:



(2) f(t)=$\begin{cases} 0 & t<0 \\ F_0 e^{jw} & t \geq 0 \end{cases}$

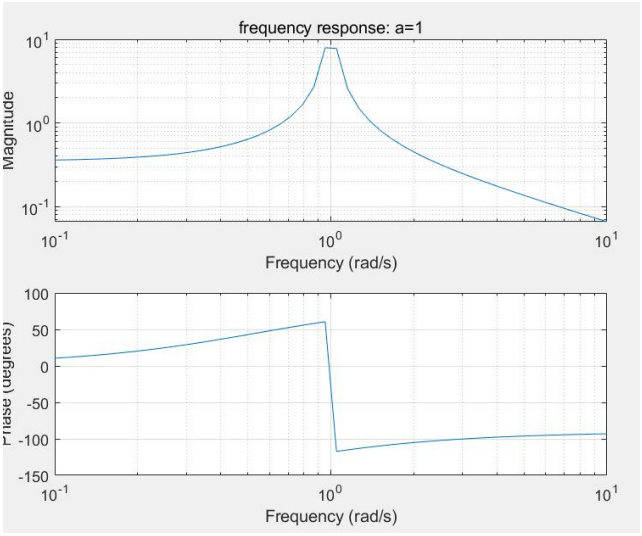
We choose F_0=1 $\omega_n = 1$ w=1

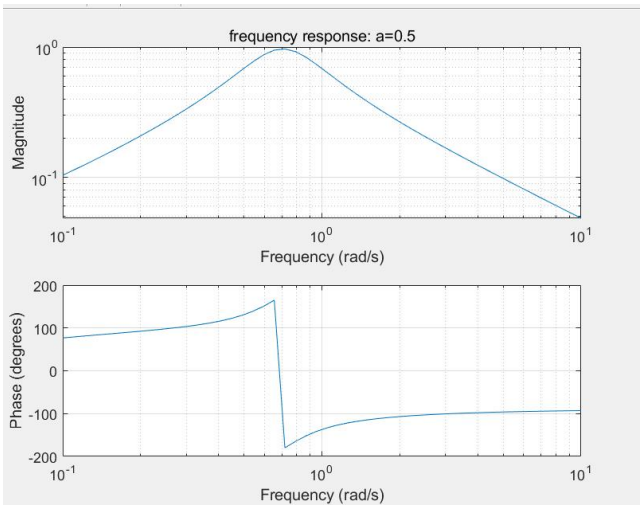
ε=0.5,1,2

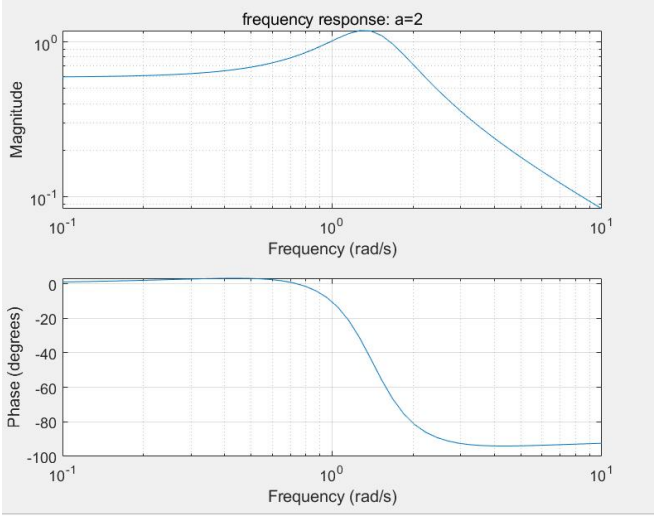
Code:

```
count  = 0;
StrName = {'a=2';'a=1';'a=0.5'};
for i = 1:3
    Value = [0.5,1,2];
    M = Value(i);
    count = count + 1
    figure()
    a0 = [1 (1/M-a) a*1/M];
    TMP1 = C+1*1/(a*M+1);
    TMP2 = 1*1/(a*M+1)*1/M-a*C;
    b = [0 TMP1 TMP2];
    w = logspace(-1,1);
    freqs(b,a0,w)
    title(['frequency response: ',StrName{count}])
    grid on;

end
```

Figure:






frequency response: a=2
Magnitude
10^0
10^{-1}
10^{-1}
10^0
10^1
Frequency (rad/s)
Phase (degrees)
0
-20
-40
-60
-80
-100
10^{-1}
10^0
10^1
Frequency (rad/s)

2-DOF and 7-DOF Dynamic Modeling and Simulation of Automobiles

Cheng Shi

Table of contents：

1. Building a road model

When analyzing the active suspension control process, the road input is an important factor that cannot be ignored. In this paper, the white noise signal is used to stimulate the road input,

$$\dot{x}_g(t) = -2\pi f_0 x_g(t) + 2\pi\sqrt{G_0 U_0}\, w(t)$$

in the formula, f_0 is the lower cutoff frequency, Hz; G_0 is the road roughness coefficient, m^3/cycle; U_0 is the forward speed, m/sec; w is random input unit white noise with mean zero. The above formula shows that the road surface displacement can be expressed as a randomly filtered white noise signal. This representation is derived from the shape of the power spectral density (PSD) curve of road surface roughness measured experimentally. We can add the road surface input to the model in the form of an equation of state:

$$\begin{cases} \dot{X}_{road} = A_{road} X + F_{road} W \\ Y_{road} = C_{road} X \end{cases}$$

$X_{road} = x_g, A_{road} = -2\pi f_0, B_{road} = 2\pi\sqrt{G_0 U_0}, C_{road} = 1$; D=0;

Considering that the road surface is an ordinary road, the road surface roughness coefficient G_0=5e-6m^3/cycle; speed U_0=20m/s; In modeling, road random white noise can be generated by random number generation or limited bandwidth white noise. This paper uses bandwidth white noise generation, and uses MATLAB/simulink to establish a simulation model as follows:

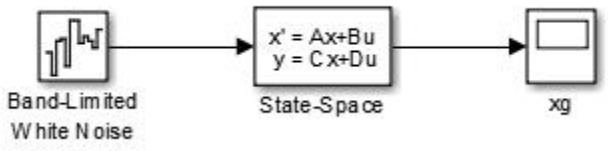


Fig. 1 Pavement model

2. Modeling of a 2-DOF system of a car

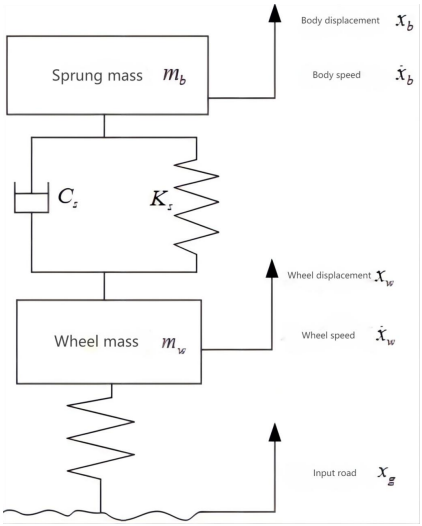


Fig. 2 Model of car 2-DOF system

According to the 2-DOF system model of the car shown in Figure 2, first establish the differential equation of motion:

$$\begin{cases} m_b\ddot{x}_b = -K_s(x_b - x_w) - C_s(\dot{x}_b - \dot{x}_w) \\ m_w\ddot{x}_w = -K_t(x_w - x_g) + K_s(x_b - x_w) + C_s(\dot{x}_b - \dot{x}_w) \end{cases}$$

Simplified：

$$\begin{cases} \ddot{x}_b = \dfrac{-C_s}{m_b}xb + \dfrac{-C_s}{m_b}x_w + \dfrac{-K_s}{m_b}x_b + \dfrac{-K_s}{m_b}x_b \\ \ddot{x}_w = \dfrac{-C_s}{m_w}x_b + \dfrac{-C_s}{m_w}x_b + \dfrac{-K_s}{m_w}x_b + \dfrac{-K_s - K_t}{m_w}x_b + \dfrac{K_t}{m_w}x_g \end{cases}$$

In the formula, C_s is the suspension damping, K_s A is the suspension stiffness, K_t A is tire stiffness, m_b is the body mass, m_w is the wheel mass, x_b、$\dot{x}_b$、$\ddot{x}_b$ are the body displacement, velocity, and acceleration, respectively. x_w、$\dot{x}_w$、$\ddot{x}_w$ are the wheel displacement, velocity, and acceleration, respectively, x_g is the road input.

The state variables and input vectors are selected as:

$$X = \begin{bmatrix} \dot{x}_b & \dot{x}_w & x_b & x_w \end{bmatrix} \qquad U = x_g$$

Then the system motion equation and road excitation can be written in the form of state space matrix:

$$\dot{X} = AX + BU$$

In the formula, A is the state matrix, B is the input matrix, and its values are as follows:

$$A = \begin{bmatrix} -\frac{C_s}{m_b} & \frac{C_s}{m_b} & -\frac{K_s}{m_b} & \frac{K_s}{m_b} \\ \frac{C_s}{m_w} & -\frac{C_s}{m_w} & \frac{K_s}{m_w} & \frac{-K_s - K_s}{m_w} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{K_t}{m_w} \\ 0 \\ 0 \end{bmatrix}$$

Taking body acceleration, tire dynamic deformation, and suspension dynamic travel as performance indicators:

$$Y = [x_b \quad x_w - x_g \quad x_b - x_w]^T$$

Write the performance index term as a linear combination of state variables and input signals:

$$Y = CX + DU$$

In the formula:

$$C = \begin{bmatrix} -\frac{Cs}{m_b} & \frac{Cs}{m_b} & -\frac{Ks}{m_b} & \frac{Ks}{m_b} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

The simulation model established by MATLAB/simulink is as follows:

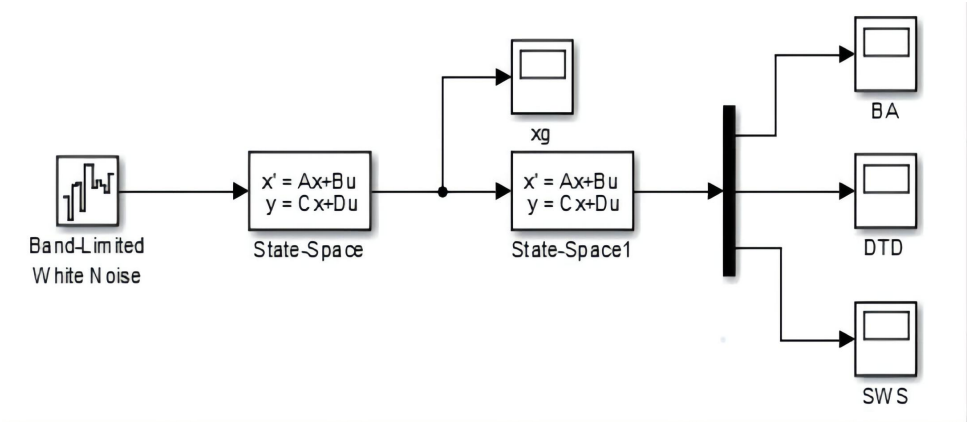


Fig. 3 Simulink model of car with 2 degrees of freedom

3. Modeling of a 7-DOF system of a car

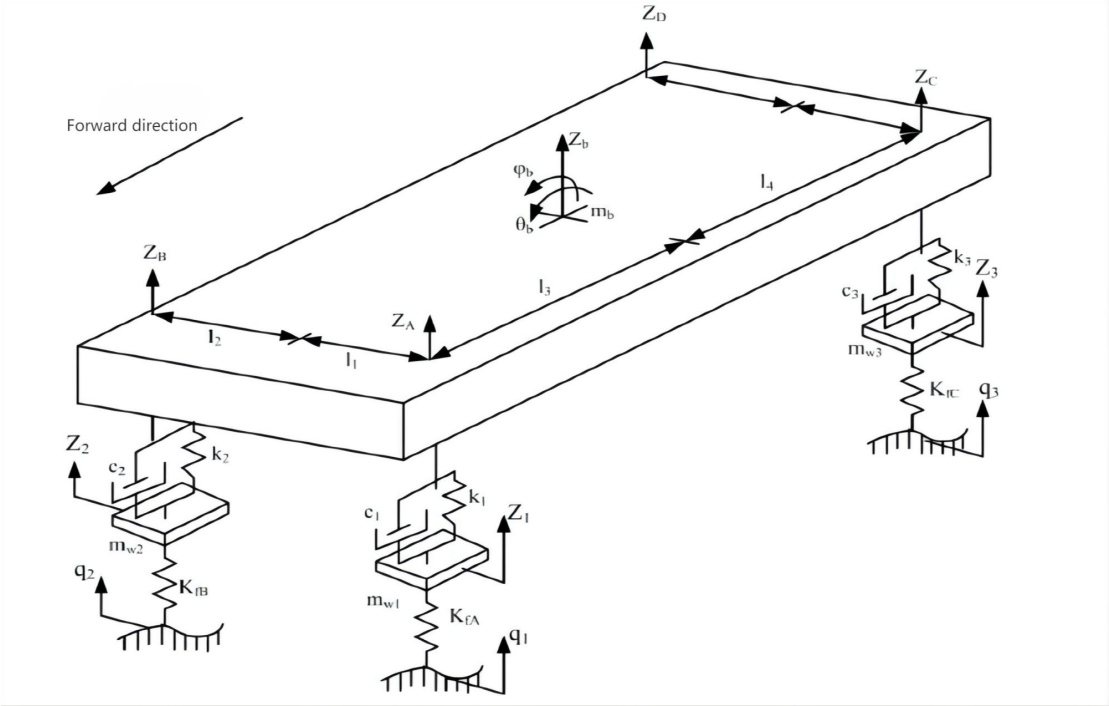


Fig. 4 Model of car 7-DOF system

According to the 2-DOF system model of the car shown in Figure 2, first establish the differential equation of motion.When the pitch angle θ_b and the roll angle ϕ are small, the vertical displacements at the four end points of the vehicle body have the following relationship:

$$z_{bA} = z_b - a\theta_b + \frac{1}{2}B_f\phi \tag{1}$$

$$z_{bB} = z_b - a\theta_b - \frac{1}{2}B_f\phi \tag{2}$$

$$z_{bC} = z_b + b\theta_b + \frac{1}{2}B_r\phi \tag{3}$$

$$z_{bD} = z_b + b\theta_b - \frac{1}{2}B_r\phi \tag{4}$$

Therefore, the equation of vertical motion at the center of mass of the body is:

$$\begin{aligned} m_b\ddot{z}_b = {} & C_{sA}(\dot{z}_{wA} - \dot{z}_{bA}) + k_{sA}(z_{wA} - z_{bA}) + C_{sB}(\dot{z}_{wB} - \dot{z}_{bB}) + k_{sB}(z_{wB} - z_{bB}) \\ & + C_{sC}(\dot{z}_{wC} - \dot{z}_{bC}) + k_{sC}(z_{wC} - z_{bC}) + C_{sD}(\dot{z}_{wD} - \dot{z}_{bD}) + k_{sD}(z_{wD} - z_{bD}) \end{aligned} \tag{5}$$

The body pitch motion equation is:

$$\begin{aligned} I_p\ddot{\theta}_b = {} & b[C_{sC}(\dot{z}_{wC} - \dot{z}_{bC}) + k_{sC}(z_{wC} - z_{bC}) + C_{sD}(\dot{z}_{wD} - \dot{z}_{bD}) + k_{sD}(z_{wD} - z_{bD})] \\ & - a[C_{sA}(\dot{z}_{wA} - \dot{z}_{bA}) + k_{sA}(z_{wA} - z_{bA}) + C_{sB}(\dot{z}_{wB} - \dot{z}_{bB}) + k_{sB}(z_{wB} - z_{bB})] \end{aligned} \tag{6}$$

The body roll motion equation is:

$$I_r\ddot{\phi} = [C_{sA}(\dot{z}_{wA} - \dot{z}_{bA}) + k_{sA}(z_{wA} - z_{bA}) - C_{sB}(\dot{z}_{wB} - \dot{z}_{bB}) - k_{sB}(z_{wB} - z_{bB})\frac{B_f}{2} + [C_{sC}(\dot{z}_{wC} - \dot{z}_{bC}) + k_{sC}(z_{wC} - z_{bC}) - C_{sD}(\dot{z}_{wD} - \dot{z}_{bD}) - k_{sD}(z_{wD} - z_{bD})\frac{B_r}{2} \quad (7)$$

The vertical motion equations of the four unsprung masses are:

$$m_{wA}\ddot{z}_{wA} = k_{tA}(z_{gA} - z_{wA}) + k_{sA}(z_{bA} - z_{wA}) + C_{sA}(\dot{z}_{bA} - \dot{z}_{wA}) \quad (8)$$

$$m_{wB}\ddot{z}_{wB} = k_{tB}(z_{gB} - z_{wB}) + k_{sB}(z_{bB} - z_{wB}) + C_{sB}(\dot{z}_{bB} - \dot{z}_{wB}) \quad (9)$$

$$m_{wC}\ddot{z}_{wC} = k_{tC}(z_{gC} - z_{wC}) + k_{sC}(z_{bC} - z_{wC}) + C_{sC}(\dot{z}_{bC} - \dot{z}_{wC}) \quad (10)$$

$$m_{wD}\ddot{z}_{wD} = k_{tD}(z_{gD} - z_{wD}) + k_{sD}(z_{bD} - z_{wD}) + C_{sD}(\dot{z}_{bD} - \dot{z}_{wD}) \quad (11)$$

The seven differential equations (5) to (11) above represent the vehicle dynamics model with seven degrees of freedom. Taking z_b 、 θ_b 、 ϕ 、 z_{wA} 、 z_{wB} 、 z_{wC} and z_{wD} as state variables to establish a differential matrix equation in the form of $M\ddot{X} + C\dot{X} + KX = K_t Z_g$, we get:

$$\begin{aligned} & m_b\ddot{z}_b \\ & + (C_{sA} + C_{sB} + C_{sC} + C_{sD})\dot{z}_b + (-aC_{sA} - aC_{sB} + bC_{sC} + bC_{sD})\dot{\theta}_b + \frac{1}{2}(B_fC_{sA} - B_fC_{sB} + B_rC_{sC} - B_rC_{sD})\dot{\phi} \\ & - C_{sA}\dot{z}_{wA} - C_{sB}\dot{z}_{wB} - C_{sC}\dot{z}_{wC} - C_{sD}\dot{z}_{wD} \\ & + (K_{sA} + K_{sB} + K_{sC} + K_{sD})z_b + (-aK_{sA} - aK_{sB} + bK_{sC} + bK_{sD})\theta_b + \frac{1}{2}(B_fK_{sA} - B_fK_{sB} + B_rK_{sC} - B_rK_{sD})\phi \\ & - K_{sA}z_{wA} - K_{sB}z_{wB} - K_{sC}z_{wC} - K_{sD}z_{wD} = 0 \end{aligned} \quad (12)$$

$$\begin{aligned} & I_p\ddot{\theta}_b \\ & + (-aC_{sA} - aC_{sB} + bC_{sC} + bC_{sD})\dot{z}_b + (a^2C_{sA} + a^2C_{sB} + b^2C_{sC} + b^2C_{sD})\dot{\theta}_b \\ & + \frac{1}{2}(-aB_fC_{sA} + aB_fC_{sB} + bB_rC_{sC} - bB_rC_{sD})\dot{\phi} + aC_{sA}\dot{z}_{wA} + aC_{sB}\dot{z}_{wB} - bC_{sC}\dot{z}_{wC} - bC_{sD}\dot{z}_{wD} \\ & + (-aK_{sA} - aK_{sB} + bK_{sC} + bK_{sD})z_b + (a^2K_{sA} + a^2K_{sB} + b^2K_{sC} + b^2K_{sD})\theta_b \\ & + \frac{1}{2}(-aB_fK_{sA} + aB_fK_{sB} + bB_rK_{sC} - bB_rK_{sD})\phi + aK_{sA}z_{wA} + aK_{sB}z_{wB} - bK_{sC}z_{wC} - bK_{sD}z_{wD} = 0 \end{aligned} \quad (13)$$

$$
\begin{aligned}
&I_r\ddot{\phi} \\
&+\frac{1}{2}(B_fC_{sA}-B_fC_{sB}+B_rC_{sC}-B_rC_{sD})\dot{z}_b+\frac{1}{2}(-aB_fC_{sA}+aB_fC_{sB}+bB_rC_{sC}-bB_rC_{sD})\dot{\theta}_b \\
&+\frac{1}{4}(B_f^2C_{sA}+B_f^2C_{sB}+B_r^2C_{sC}+B_r^2C_{sD})\dot{\phi}-\frac{B_fC_{sA}}{2}\dot{z}_{wA}+\frac{B_fC_{sB}}{2}\dot{z}_{wB}-\frac{B_rC_{sC}}{2}\dot{z}_{wC}+\frac{B_rC_{sD}}{2}\dot{z}_{wD} \\
&+\frac{1}{2}(B_fK_{sA}-B_fK_{sB}+B_rK_{sC}-B_rK_{sD})z_b+\frac{1}{2}(-aB_fK_{sA}+aB_fK_{sB}+bB_rK_{sC}-bB_rK_{sD})\theta_b \\
&+\frac{1}{4}(B_f^2K_{sA}+B_f^2K_{sB}+B_r^2K_{sC}+B_r^2K_{sD})\phi-\frac{B_fK_{sA}}{2}z_{wA}+\frac{B_fK_{sB}}{2}z_{wB}-\frac{B_rK_{sC}}{2}z_{wC}+\frac{B_rK_{sD}}{2}z_{wD}=0
\end{aligned}
\tag{14}
$$

$$
m_{wA}\ddot{z}_{wA}-C_{sA}\dot{z}_b+aC_{sA}\dot{\theta}_b-\frac{B_fC_{sA}}{2}\dot{\phi}+C_{sA}\dot{z}_{wA}-K_{sA}z_b+aK_{sA}\theta_b-\frac{B_fK_{sA}}{2}\phi+(K_{sA}+K_{tA})z_{wA}=K_{tA}z_{gA} \tag{15}
$$

$$
m_{wB}\ddot{z}_{wB}-C_{sB}\dot{z}_b+aC_{sB}\dot{\theta}_b+\frac{B_fC_{sB}}{2}\dot{\phi}+C_{sB}\dot{z}_{wB}-K_{sB}z_b+aK_{sB}\theta_b+\frac{B_fK_{sB}}{2}\phi+(K_{sB}+K_{tB})z_{wB}=K_{tB}z_{gB} \tag{16}
$$

$$
m_{wC}\ddot{z}_{wC}-C_{sC}\dot{z}_b-bC_{sC}\dot{\theta}_b-\frac{B_rC_{sC}}{2}\dot{\phi}+C_{sC}\dot{z}_{wC}-K_{sC}z_b-bK_{sC}\theta_b-\frac{B_rK_{sC}}{2}\phi+(K_{sC}+K_{tC})z_{wC}=K_{tC}z_{gC} \tag{17}
$$

$$
m_{wD}\ddot{z}_{wD}-C_{sD}\dot{z}_b-bC_{sD}\dot{\theta}_b+\frac{B_rC_{sD}}{2}\dot{\phi}+C_{sD}\dot{z}_{wD}-K_{sD}z_b-bK_{sD}\theta_b+\frac{B_rK_{sD}}{2}\phi+(K_{sD}+K_{tD})z_{wD}=K_{tD}z_{gD} \tag{18}
$$

Taking the coefficients of the differential equations (12) to (18), the mass matrix M , damping matrix C , stiffness matrix K and input matrix K_t :

$$
M=\begin{bmatrix}
m_b & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & I_p & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & I_r & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & m_{wA} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & m_{wB} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & m_{wC} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & m_{wD}
\end{bmatrix}
\qquad
K_t=\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
K_{tA} & 0 & 0 & 0 \\
0 & K_{tB} & 0 & 0 \\
0 & 0 & K_{tC} & 0 \\
0 & 0 & 0 & K_{tD}
\end{bmatrix}
$$

$$C=\begin{bmatrix} C_{sA}+C_{sB}+C_{sC}+C_{sD} & -aC_{sA}-aC_{sB}+bC_{sC}+bC_{sD} \\ -aC_{sA}-aC_{sB}+bC_{sC}+bC_{sD} & a^2C_{sA}+a^2C_{sB}+b^2C_{sC}+b^2C_{sD} \\ \frac{1}{2}(B_fC_{sA}-B_fC_{sB}+B_rC_{sC}-B_rC_{sD}) & \frac{1}{2}(-aB_fC_{sA}+aB_fC_{sB}+bB_rC_{sC}-bB_rC_{sD}) \\ -C_{sA} & aC_{sA} \\ -C_{sB} & aC_{sB} \\ -C_{sC} & -bC_{sC} \\ -C_{sD} & -bC_{sD} \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{2}(B_fC_{sA}-B_fC_{sB}+B_rC_{sC}-B_rC_{sD}) & -C_{sA} & -C_{sB} & -C_{sC} & -C_{sD} \\ \frac{1}{2}(-aB_fC_{sA}+aB_fC_{sB}+bB_rC_{sC}-bB_rC_{sD}) & aC_{sA} & aC_{sB} & -bC_{sC} & -bC_{sD} \\ \frac{1}{4}(B_f^2C_{sA}+B_f^2C_{sB}+B_r^2C_{sC}+B_r^2C_{sD}) & -\frac{B_fC_{sA}}{2} & \frac{B_fC_{sB}}{2} & -\frac{B_rC_{sC}}{2} & \frac{B_rC_{sD}}{2} \\ -\frac{B_fC_{sA}}{2} & C_{sA} & 0 & 0 & 0 \\ \frac{B_fC_{sB}}{2} & 0 & C_{sB} & 0 & 0 \\ -\frac{B_rC_{sC}}{2} & 0 & 0 & C_{sC} & 0 \\ \frac{B_rC_{sD}}{2} & 0 & 0 & 0 & C_{sD} \end{bmatrix}$$

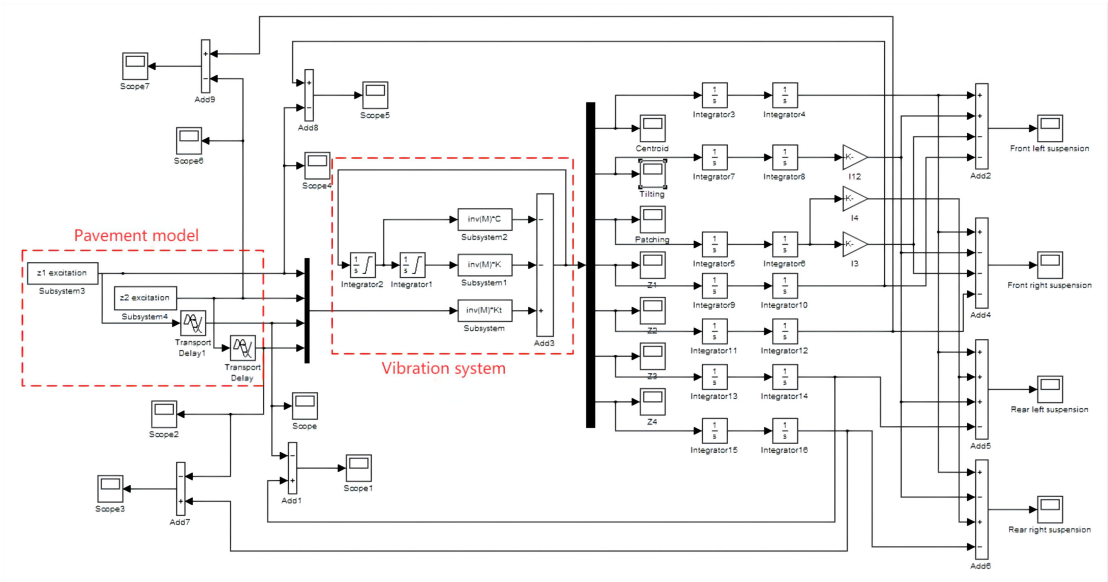
$$K=\begin{bmatrix} K_{sA}+K_{sB}+K_{sC}+K_{sD} & -aK_{sA}-aK_{sB}+bK_{sC}+bK_{sD} \\ -aK_{sA}-aK_{sB}+bK_{sC}+bK_{sD} & a^2K_{sA}+a^2K_{sB}+b^2K_{sC}+b^2K_{sD} \\ \frac{1}{2}(B_fK_{sA}-B_fK_{sB}+B_rK_{sC}-B_rK_{sD}) & \frac{1}{2}(-aB_fK_{sA}+aB_fK_{sB}+bB_rK_{sC}-bB_rK_{sD}) \\ -K_{sA} & aK_{sA} \\ -K_{sB} & aK_{sB} \\ -K_{sC} & -bK_{sC} \\ -K_{sD} & -bK_{sD} \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{2}(B_fK_{sA}-B_fK_{sB}+B_rK_{sC}-B_rK_{sD}) & -K_{sA} & -K_{sB} & -K_{sC} & -K_{sD} \\ \frac{1}{2}(-aB_fK_{sA}+aB_fK_{sB}+bB_rK_{sC}-bB_rK_{sD}) & aK_{sA} & aK_{sB} & -bK_{sC} & -bK_{sD} \\ \frac{1}{4}(B_f^2K_{sA}+B_f^2K_{sB}+B_r^2K_{sC}+B_r^2K_{sD}) & -\frac{B_fK_{sA}}{2} & \frac{B_fK_{sB}}{2} & -\frac{B_rK_{sC}}{2} & \frac{B_rK_{sD}}{2} \\ -\frac{B_fK_{sA}}{2} & K_{sA}+K_{tA} & 0 & 0 & 0 \\ \frac{B_fK_{sB}}{2} & 0 & K_{sB}+K_{tB} & 0 & 0 \\ -\frac{B_rK_{sC}}{2} & 0 & 0 & K_{sC}+K_{tC} & 0 \\ \frac{B_rK_{sD}}{2} & 0 & 0 & 0 & K_{sD}+K_{tD} \end{bmatrix}$$

In the differential equation, the subscripts A, B, C, and D represent the front left, front right, rear left, and rear right wheels, respectively, z_b is the vertical displacement at the center of mass of the body, z_w is the vertical displacement of the wheel, θ_b is the pitch angle of the vehicle, ϕ

is the roll angle of the vehicle, and z_g is the input vertical displacement of the road surface.In addition, m_b is the sprung mass of the vehicle, m_w is the unsprung mass of the vehicle, a is the distance from the center of mass of the body to the front axle, b is the distance from the center of mass of the body to the rear axle, B_f is the wheel track of the front wheel, B_r is the wheel track of the rear wheel, K_s is the spring stiffness of the suspension, K_t is the tire stiffness, C_s is the damping coefficient of the suspension, I_p is the pitch moment of inertia, and I_r is the roll moment of inertia.

The simulation model established by MATLAB/Simulink is as follows:



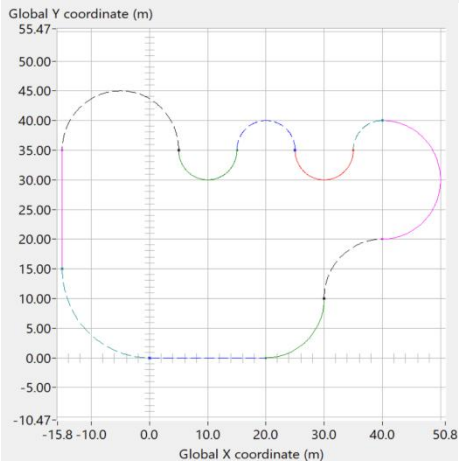
Lateral control of driverless vehicle (Carsim+Simulink)

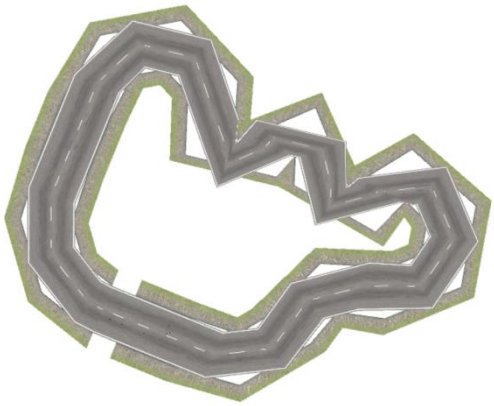
Shi Cheng

Table of contents：

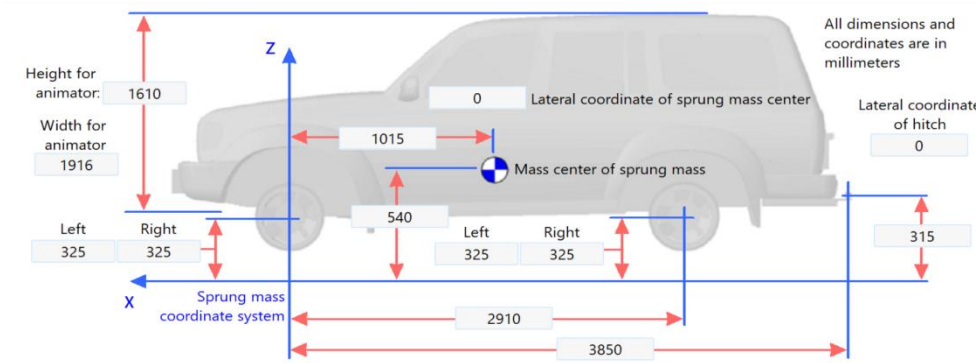
1. Simulation settings

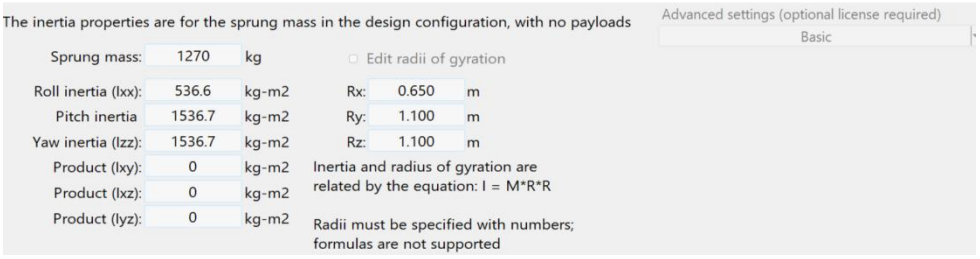
1.1 CarSim road setting and simulation

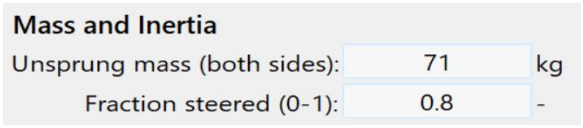




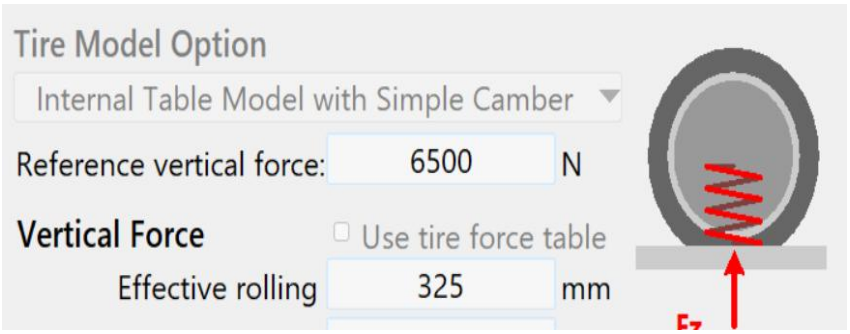
1.2 Select the correlation coefficient of the simulation vehicle - the distance from the mass point to the front wheel, the distance from the mass point to the rear wheel, and the weight of the whole vehicle (sprung mass + unsprung mass), moment of inertia.

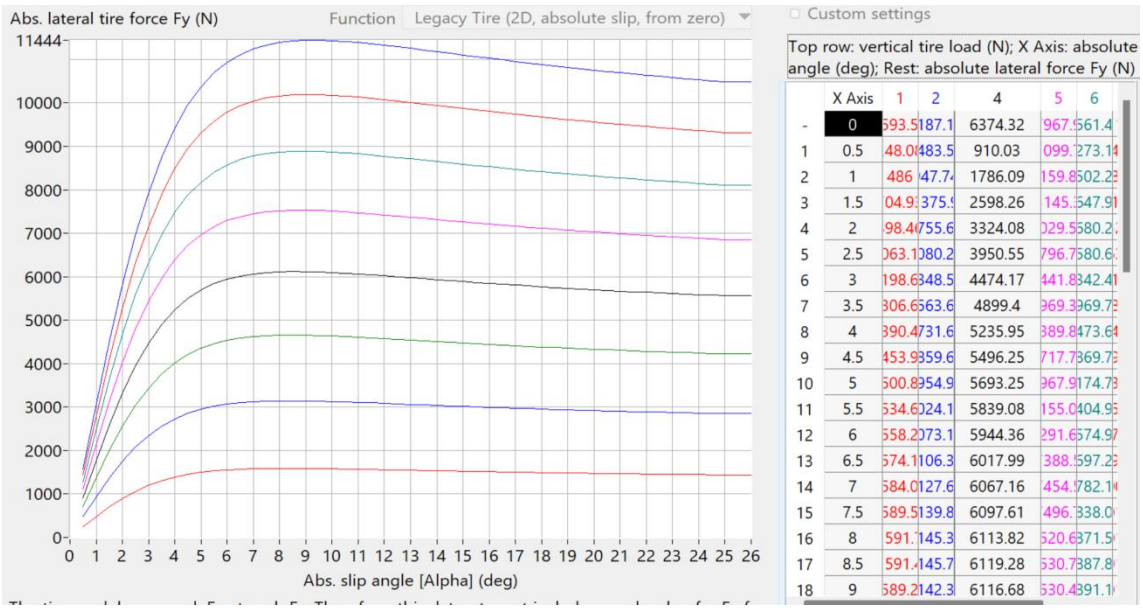




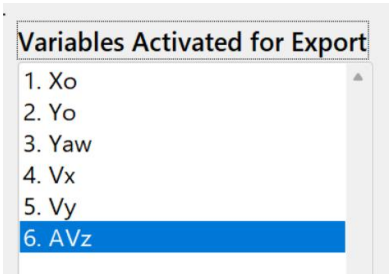


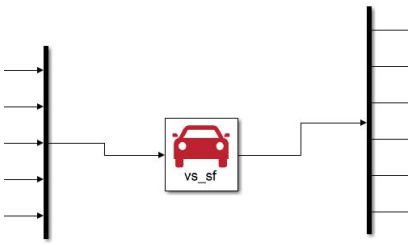
1.3 Idealized selection and estimation of tire cornering stiffness for simulation vehicle





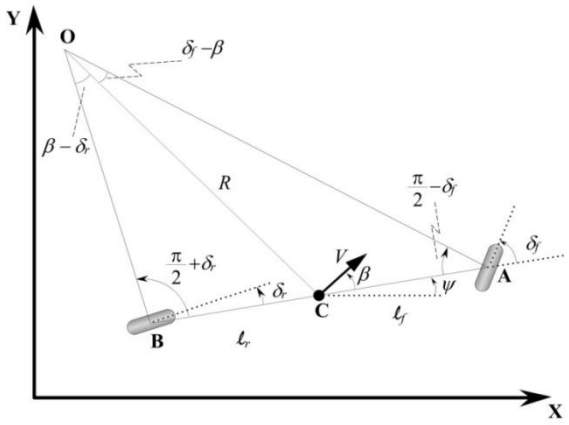
1.4 Input and output settings of vehicle model in CarSim in Simulink





2. Algorithm

2.1 Tire cornering stiffness and vehicle dynamics equation



Idealized bicycle model of automobile

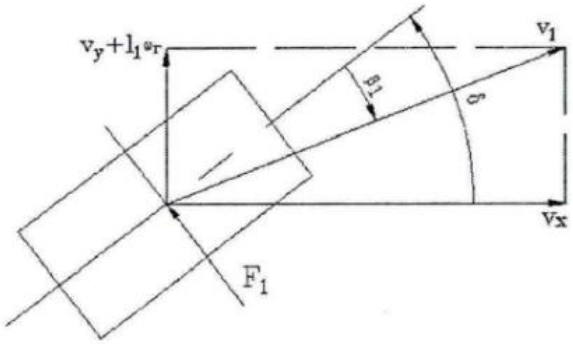
Symbol	Nomenclature	Equation
X	Global X axis coordinate	$\dot{X} = V\cos(\psi+\beta)$
Y	Global Y axis coordinate	$\dot{Y} = V\sin(\psi+\beta)$
ψ	Yaw angle; orientation angle of vehicle with respect to global X axis	$\dot{\psi} = \frac{V\cos(\beta)}{\ell_f+\ell_r}\left(\tan(\delta_f)-\tan(\delta_r)\right)$

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

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

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

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）

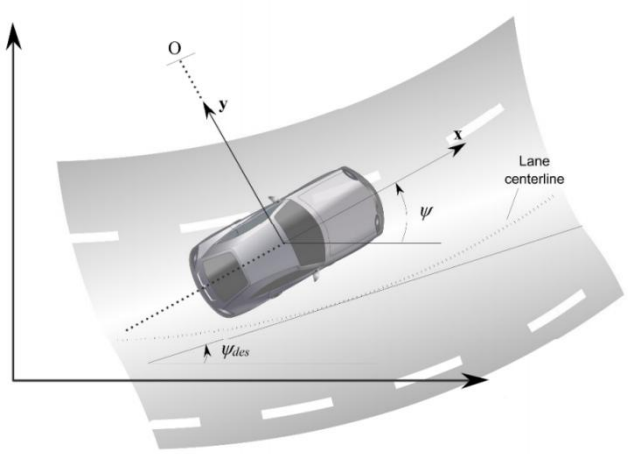
$$m(\ddot{y}+\dot{\psi}V_x)=F_{yf}+F_{yr}$$

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

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

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

2.2 Lateral error differential equation



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

or

$$\vec{a}_{inertial} = \dot{\psi}\hat{k}\,\mathrm{x}\,(\dot{\psi}\hat{k}\ \mathrm{x}\ -\mathrm{R}\hat{\mathrm{j}}) + \ddot{\psi}\,\hat{k}\ \mathrm{x}\ -\mathrm{R}\hat{\mathrm{j}} + 2\dot{\psi}\,\hat{k}\ \mathrm{x}\ -\dot{\mathrm{R}}\hat{\mathrm{j}} + \ddot{x}\,\hat{\imath} + \ddot{y}\,\hat{\jmath}$$

or

$$\vec{a}_{inertial} = \dot{\psi}^2 R\,\hat{\jmath} + (R\ddot{\psi} + 2\dot{\psi}\dot{R})\,\hat{\imath} + \ddot{x}\,\hat{\imath} + \ddot{y}\,\hat{\jmath} \qquad (2.36)$$

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

$$a_y = \ddot{y} + V_x\dot{\psi} \qquad (2.37)$$

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

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

and

$$e_2 = \psi - \psi_{des} \qquad (2.41)$$

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

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

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.

$$\dot{X} = AX + Bu \xrightarrow{\text{discrete}} X_{k+1} = \bar{A}X_k + \bar{B}u_k$$

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 $X(\zeta) = \frac{X(t) + X(t+dt)}{2}$, using forward Euler method for u

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

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

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

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

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

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

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

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,

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

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.

$$e_{rr} = \begin{pmatrix} \frac{1}{k_1}\left\{\delta_f - \frac{\dot{\theta}_r}{v_x}\left[a + b - bK_3 - \frac{mv_x^2}{a+b}\left(\frac{b}{C_f} + \frac{a}{C_r}K_3 - \frac{a}{C_r}\right)\right]\right\} \\ 0 \\ -\frac{\dot{\theta}_r}{v_x}\left(b + \frac{a}{a+b}\frac{mv_x^2}{C_{ar}}\right) \\ 0 \end{pmatrix}$$

We suppose

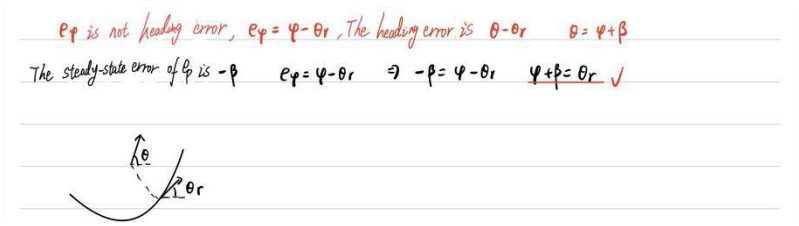
$$\delta_f = \frac{\dot{\theta}_r}{v_x}\left[a + b - bK_3 - \frac{mv_x^2}{a+b}\left(\frac{b}{C_f} + \frac{a}{C_r}K_3 - \frac{a}{C_r}\right)\right]$$

$$\theta_r = k v_x$$

$$\delta_f = k\left[a + b - bk_3 - \frac{mv_x^2}{a+b}\left(\frac{b}{C_{af}} + \frac{a}{C_{ar}}k_3 - \frac{a}{C_{ar}}\right)\right]$$

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

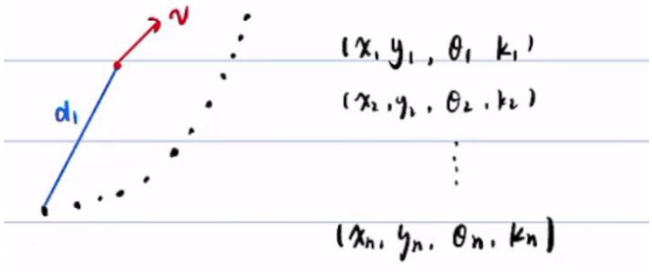
It is found that the error of e_φ in err cannot be eliminated, and it is simplified after a series of ideal transformations.



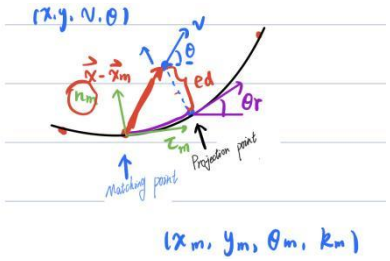
The ultimate goal is: θ-θr=0, which exactly eliminates the heading error, so this item does not need to be 0

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.

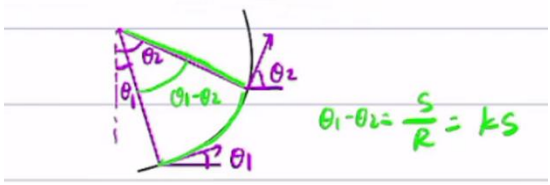
$\vec{\tau}_m = (\cos\theta_m, \sin\theta_m)$ $\vec{n}_m = (-\sin\theta_m, \cos\theta_m)$

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

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

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

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



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

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

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

$k_r = k_m$

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

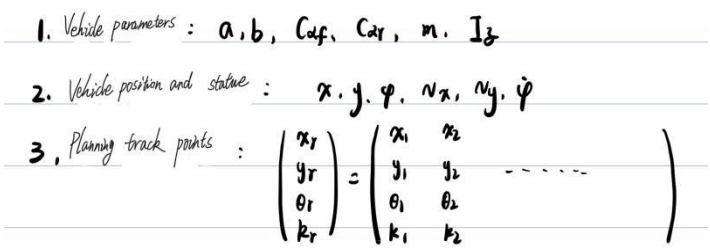
$e_\varphi = \varphi - \theta_r$

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

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

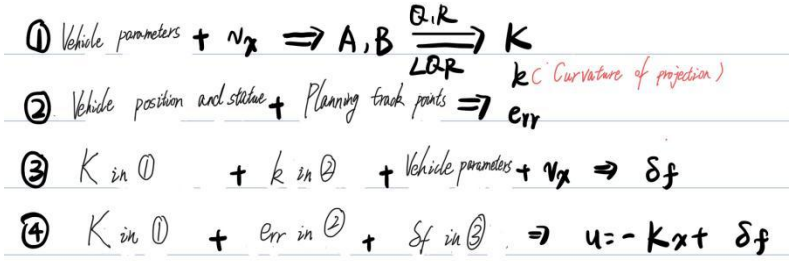
2.6 Algorithm summary

Input in algorithm

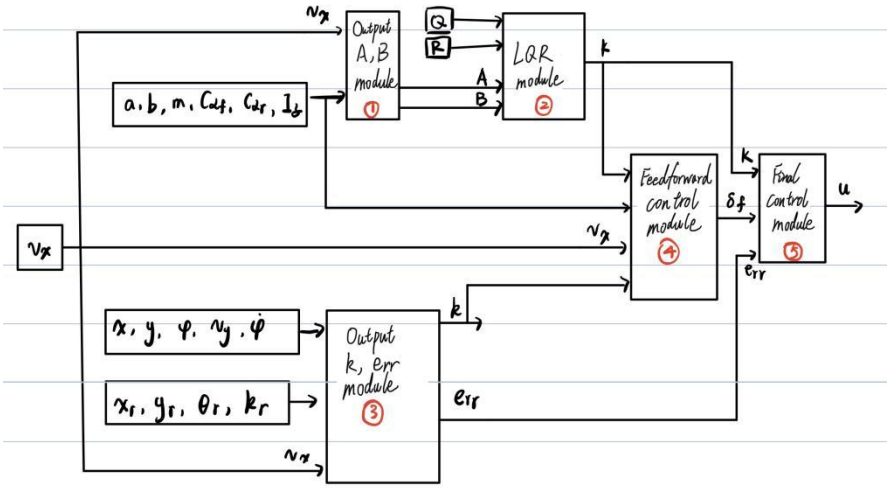


Output in algorithm

Process



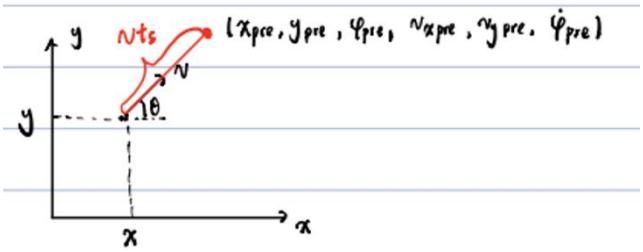
Module flow chart



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’.

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

$$y_{pre} = y + v t_s \sin\theta = y + v_y t_s \cos\varphi + v_x t_s \sin\varphi$$

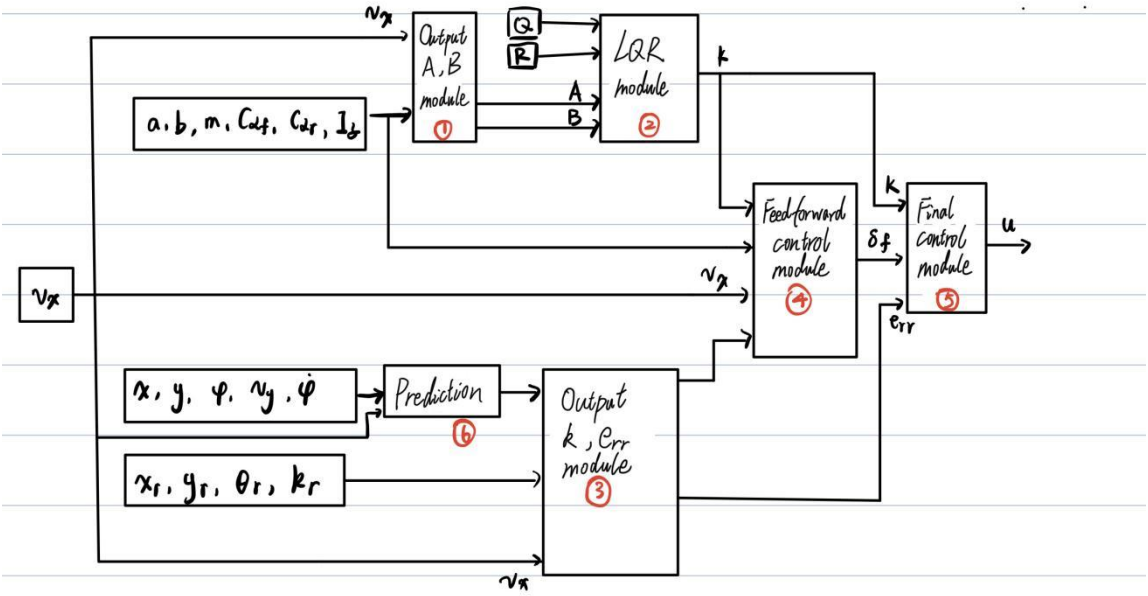
$$\varphi_{pre} = \varphi + \dot{\varphi} t_s$$

$$v_{xpre} = v_x$$

$$v_{ypre} = v_y$$

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

3.3 Optimized module flow chart



4. Code and Simulation

4.1 Code

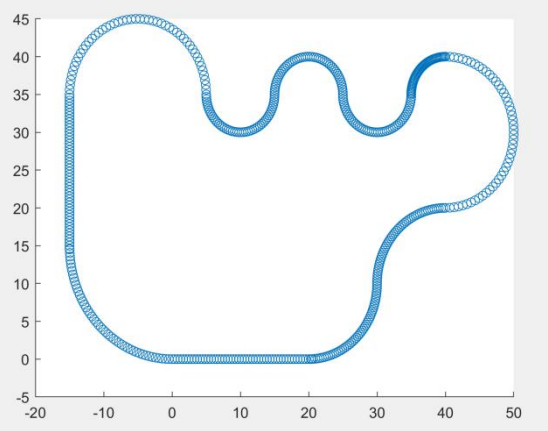
4.1.1 Routing Planning

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

```
function [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

```
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
```

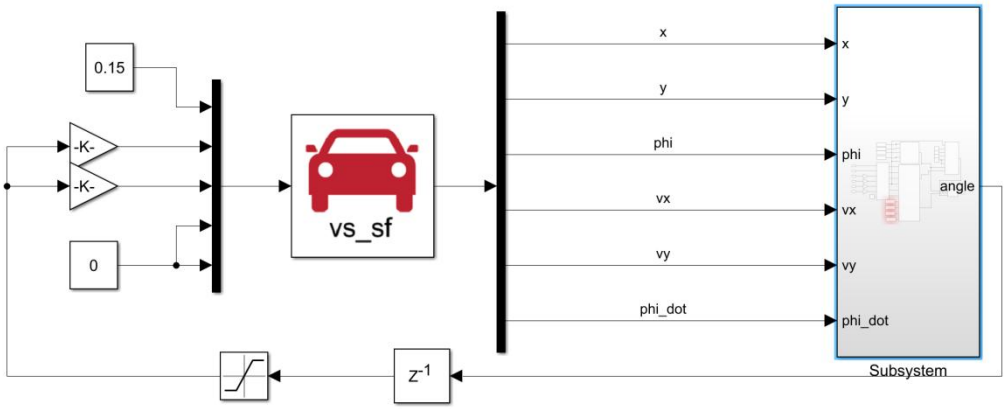
4.1.5 Final control Module

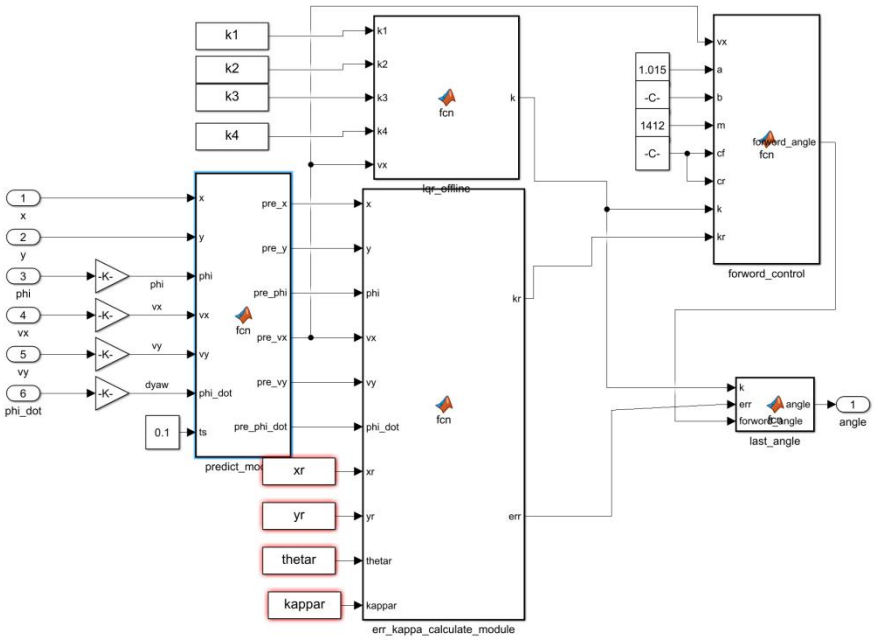
```
function angle = fcn(k,err,forword_angle)

    angle=-k*err+forword_angle;
end
```

4.1.6 Prediction Module

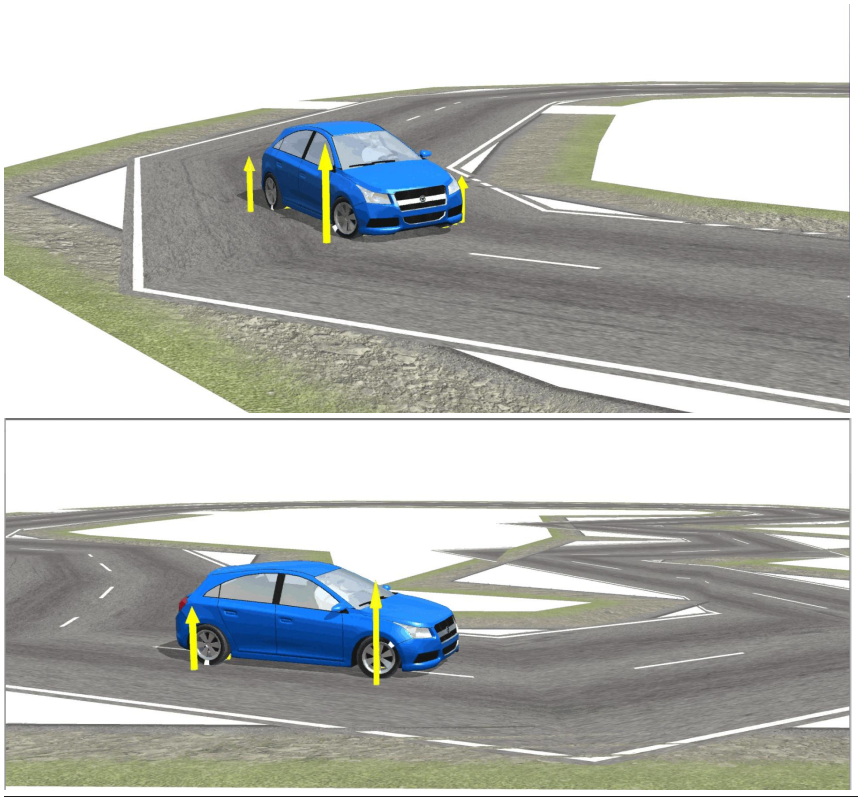
```
function [pre_x,pre_y,pre_phi,pre_vx,pre_vy,pre_phi_dot] = fcn(x,y,phi,vx,vy,phi_dot,ts)
    pre_x=x+vx*ts*cos(phi)-vy*ts*sin(phi);
    pre_y=y+vy*ts*cos(phi)+vx*ts*sin(phi);
    pre_phi=phi+phi_dot*ts;
    pre_vx=vx;
    pre_vy=vy;
    pre_phi_dot=phi_dot;
end
```

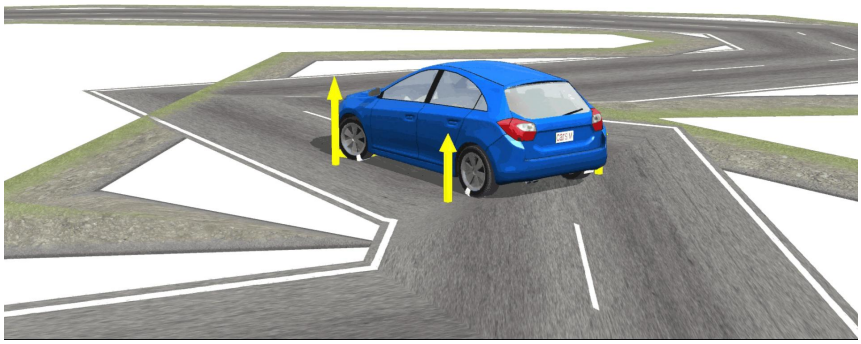
4.2 Simulink Module

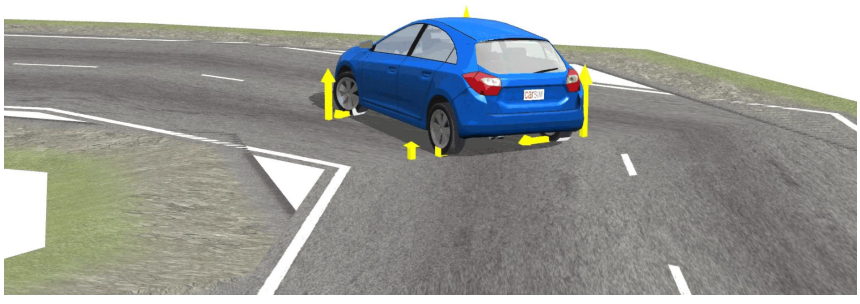




4.3 Carsim Simulation







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.