Cruise control

project report

Author:

William

Introduction

The project aims to create a system of control system to maintain an desired constant speed of a car. The system involved Laplace transform and lots of types of control. We realize this function by Python program.

Contents

Chapter 1

Preliminaries

This chapter will introduce the basic method of control.

1.1 Ordinary differential equation

$$\frac{dy}{dx} = f(x)$$

That's a simple form of ODE, with the order of 1. Here comes the example,

$$\mathbf{F} = \frac{d\mathbf{p}}{dt}$$

which is the newton's second law. We could solve the equations like that for dynamics problem in our cruise control project. For example, $\frac{dv}{dt} = a$, We can get v,if t is known. That helps our simulation of the motion of the cruise car. We can solve ODE use Euler's method(which is not accurate enough), integration, and Laplace transform. We will use different methods to meet our need.

1.2 Laplace transform and system modelling

1.2.1 Introduction of Laplace transform

The Laplace transform is a powerful alternative method for solving differential equations. The method involves transforming differential equations into algebraic ones. What's more, Laplace transform could solve ODE in any orders of them. The Laplace transform method has the major advantage that it could switch the ODE in different domains.

The Laplace transform is defined as below, $f(t)$is the function of a real variable t

$$L(f) = F(s) = \int_0^\infty f(t)e^{-st}\ \mathrm{d}t$$

There is an important example which could make our calculation much easier.

Example: Find the Laplace transform of $f(t) = 1(t \geq 0)$

$$F(s) = \int_0^\infty 1e^{-st}dt$$

$$\begin{aligned} \int_0^\infty f(t)e^{-st}\ \mathrm{d}t &= \lim_{T\to\infty}\int_0^T f(t)e^{-st}\ \mathrm{d}t \\ F(s) &= \left[\frac{-1}{s}e^{-st}\right]_0^\infty = \frac{1}{s} \end{aligned}$$

That makes our calculation of Laplace transform easier: if we want to 'Laplace transform' any number, we just need to multiply $\frac{1}{s}$ to the original function.

Laplace transform of $f'(t)$:

$$L\left(\frac{\mathrm{d}f}{\mathrm{d}t}\right) = \int_0^\infty e^{-st}\frac{\mathrm{d}f}{\mathrm{d}t}\ \mathrm{d}t$$

Then we use integration by parts:

$$\begin{aligned} L\left(\frac{\mathrm{d}f}{\mathrm{d}t}\right) &= \left[e^{-st}f(t)\right]_0^\infty + s\int_0^\infty e^{-st}f(t)\mathrm{d}t \\ &= -f(0) + sF(s) \end{aligned}$$

So we got the result:

$$L\left(\frac{\mathrm{d}f}{\mathrm{d}t}\right) = sF(s) - f(0)$$

1.2.2 Using Laplace transform to solve ODE

Here is the example:

$$3\frac{\mathrm{d}x}{\mathrm{d}t} + 2x = 4, (x = 0, t = 0)$$

$$3[sX(s) - x(0)] + 2X(s) = \frac{4}{s}$$

Then we simplfy the equation and get:

$$X(s) = \frac{4}{s(3s+2)}$$

Now we get the function with x-domain which is not what we want, so we need to apply Lapalace transform again to get the result in t-domain. Firstly, we use partial fraction to seperate them in several terms so that we could apply Laplace transform to them.

$$\frac{4}{s(3s+2)} = \frac{A}{s} + \frac{B}{3s+2}$$

$$A = 2, B = -6$$

$$\begin{aligned} X(s) &= \frac{2}{s} - \frac{6}{(3s+2)} \\ &= \frac{2}{s} - \frac{2}{(s+2/3)} \end{aligned}$$

From previous example or you could also check the Laplace transform table to get it transformed from:

$$x(t) = 2 - 2e^{-2t/3}$$

As we could compare the result and the common method of solving it, we know the Laplace transform could solve the ODE equation to simple terms which added together instead of complicated calculations.

1.2.3 Why we need Laplace transform in our project

Laplace transform is very crucial for the control system. In our cruise control project of a car, Laplace transform could help us transfer input signal, for example, the position of gas pedal, to the output signal, like the signals for engine and other part of the car to obtain our desired speed. That figure shows that

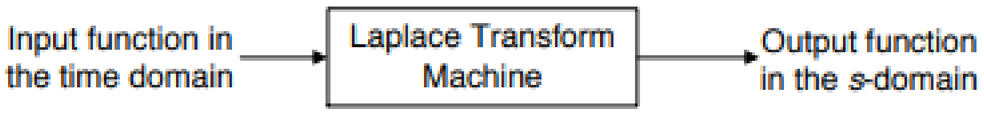

Input function in the time domain
Laplace Transform Machine
Output function in the s-domain

Figure 1.1: Laplace transform

the Laplace transform could convert an equation in the original domain(in this case, time domain) to another domain(s-domain)

1.2.4 Final value theorem

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s)$$

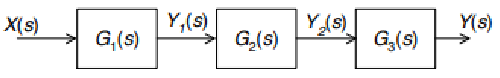
In mathematical analysis, the final value theorem (FVT) is one of several similar theorems used to relate frequency domain expressions to the time domain behavior as time approaches infinity. It's important to our later work.

1.3 Transfer function

Transfer function is a function that describe how system behaves. It's a fraction of Laplace transform of output signal over Laplace transfom of input signal. So that, we just need to input an input signal and take Laplace transform of it, then multiply its transfer function, then we get the Laplace transfom of the output signal. The last thing we do is taking inversly Laplace transform of output signal and get the desired result. We usually represent transfer functino like this:

$$\frac{Y(s)}{X(s)} = G(s)$$

A linear system which has numbers of subsystems could be represent like this:



$$G(s) = \frac{Y(s)}{X(s)} = \frac{Y_1(s)}{X(s)} \times \frac{Y_2(s)}{Y_1(s)} \times \frac{Y(s)}{Y_2(s)}$$

$$= G_1(s) \times G_2(s) \times G_3(s)$$

Figure 1.2: Linear system in series

The overall transfer function is the product of sub-transfer function. The overall transfer function describe the behaviour of the system.
As we could see the equation in Figure1.2, the overall transfer function is the ratio of two polynomials. It could be represented in form like this:

$$\begin{aligned} G(s) &= K\frac{(s-z_1)(s-z_2)\ldots(s-z_m)}{(s-p_1)(s-p_2)\ldots(s-p_n)} \\ &= K\frac{\Pi_{i=1}^{m}(s-z_i)}{\Pi_{j=1}^{n}(s-p_j)} \end{aligned}$$

Poles and zeros

The denominator of the transfer function is called poles, and the numerator of the transfer function is called zeros. We will mainly focus on poles. In page3, the order of the exponential function of the final equation is depend on the root of the poles of the transfer function. The root is negative, so we will get an convergent graph of $x(t)$. To keep our system stable, we should make sure that we have negative root in transfer function, otherwise, the system will go out of control.

1.4 Factors that effect the behaviour of the system

As we talked about before, if you got an $s - p_j$ term in the demonimator of the transfer function, when you take inverse Laplace transfer to it, it will behave like an exponential functino with the order of p_j. If p_j's value is positve, we will get our output a positive exponential functino which y value increase without limits.

1.5 Simplfing block diagrams

Here's an example of a block diagram loop.

Here's the equations of describing the system.

$$\begin{aligned} M\frac{dv}{dt} &= f_{\text{net}} \\ f_{\text{net}} &= F - f_{\text{damper}} \\ f_{\text{damper}} &= bv \\ \frac{dx}{dt} &= v \end{aligned}$$

so we could easily have block diagram like this:

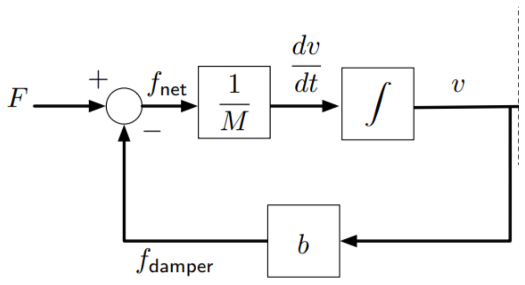


Figure 1.3: model of the equations

we know the integral in Laplace is $1/s$, so we rewrite them in domain of s.

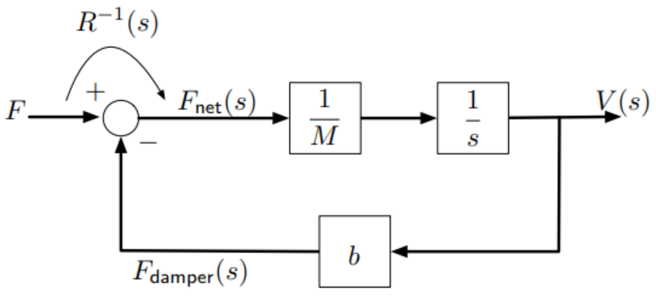


Figure 1.4: block diagram loop

The model has a single feedback loop. We call the upper straight line process 'direct path', and the whole loop 'loop path'.

According to this block diagram, we could rewrite the equations.

$$
\begin{aligned}
&F_{\text{net }}(s)=F(s)-F_{\text{damper }}(s). \\
&\mathcal{L}\left[\frac{dv}{dt}\right](s)=F_{\text{net }}(s)\cdot\frac{1}{M} \\
&V(s)=\frac{1}{s}\cdot\mathcal{L}\left[\frac{dv}{dt}\right](s) \\
&F_{\text{damper }}(s)=b\cdot V(s)
\end{aligned}
$$

Rearrange them:

$$
\begin{aligned}
&F_{\text{net }}(s)=F(s)-\frac{b}{Ms}F_{\text{net }}(s) \\
&\left[1+\frac{b}{Ms}\right]F_{\text{net }}(s)=F(s) \\
&F_{\text{net }}(s)=R^{-1}(s)F(s)
\end{aligned}
$$

which $\left[1+\frac{b}{Ms}\right] = R^{-1}(s)$ $R^{-1}(s)$ is called the return difference and the transfer function across a summing block with negative feedback is given by the inverse of the return difference.

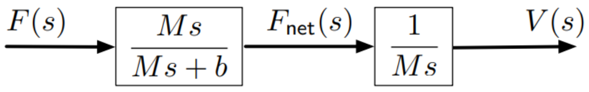


Figure 1.5: new model with blocks

As we introduced before, the blocks could be simplified by multiplying them together, so we got Here's

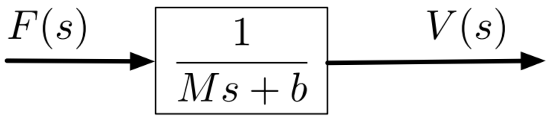


Figure 1.6: The final simplified model

two conclusions that we will get from this example, when the feed back is negative, we will have our transfer function like

$$G(s)=\frac{\text{Direct Path}}{1+\text{ Loop Path}}$$

When the feedback is positive, we will have our transfer function like

$$G(s)=\frac{\text{Direct Path}}{1-\text{ Loop Path}}$$

1.6 Euler's method

1.6.1 Introduction of Euler's method

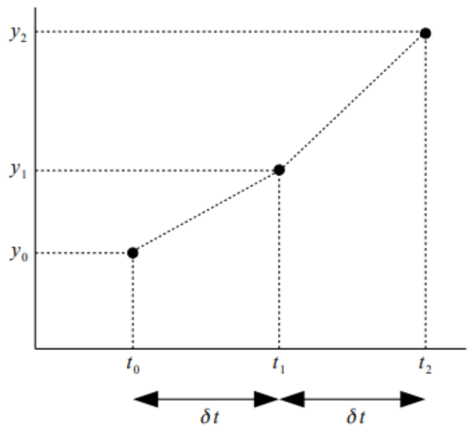
Euler's method is the method which can approximate the result of ODE in an exact point straight forward. We represent an ODE in the form below:

$$\frac{dy}{dt} = f(t, y)$$

That says $f(t, y)$ is a function of the gradient of a curve which is about t and y. The independent variable is t and dependent variable is y. Computer will plot curves by plotting huge amount of exact points and form a curve, the more points the computer plots, the more smooth the curve is. According to this scenario, we could easily got lots of points joint together and form a smooth curve if we know an initial condition like below:

$$y(t_0) = y_0$$

which t_0 and y_0 are the constant we're already known. And then, we use our initial condition to calculate the $\frac{dy_0}{dt_0}$ as the slope of the curve at t_0. Next step is to find the next point which is very close to it. The equal spacing between two adjacent points is δt. As δt is very small so we can say two adjacent points are jointed by a straight line. Therefore, we could find the t_1 and y_1.



The whole process is done, the only thing we need to do is let computer iterate it and plot a curve.

$$y_{k+1} = y_k + \delta t f(t_k, y_k)$$

1.6.2 Why we need Euler's method in our project

As Euler's method can form a curve with plenty of discrete points, it's compulsory for us to get simulation graph of our cruise control car by coding and showing the motion graph by plotting the points and form a curve in program. Euler's method is the way we get motion graph in program.

1.7 Control Theory

Control theory is to develop a model or algorithm governing the application of system inputs to drive the system to a desired state.

1.7.1 Open-loop control and closed-loop control

Open-loop control

Open-loop control is the most simple type of control that we could see it everywhere in our daily life. For example, irrigation of grass lands, the sprinklers use numbers of liters of water which the number is set up before, and use same amount of water every time, but if it' raining outside, so maybe we don't want the sprinklers to use same amount of water like before. But the sprinklers will work as usual. The input maybe is the moisture of the grassland that we desired, and the output is the water used every time by sprinklers. So how the sprinklers transfer the input signals to output signals? We call it 'PLANT' in our control block diagram. Basically 'PLANT' is the machine, used to transfer the signals.

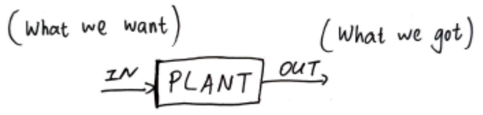


Figure 1.7: Open-loop control block diagram

That's open-loop control. Open-loop control system's input signal does not depend on the system output.

Closed-loop control

Closed-loop control control is the advanced control loop compare with the open-loop control. It involves a feedback, which doesn't appear in open-loop control. Feedback is detected by sensors. In closed-loop control, after the sensors detected the value of output, the control loop calculate the difference between the reference value, which we desired value, and form a new value to input. The advantage is that the input value could affected by the output value detected by the sensors.
So we will model our system by closed-loop control to get a better result and make our system to behave in a controlable way.

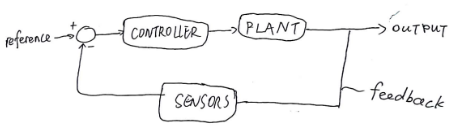


Figure 1.8: Closed-loop control block diagram

Chapter 2

System modelling with block diagrams

2.1 tranfer function design of p-control

Firstly, the diagram below shows the complete loop of control block diagram.

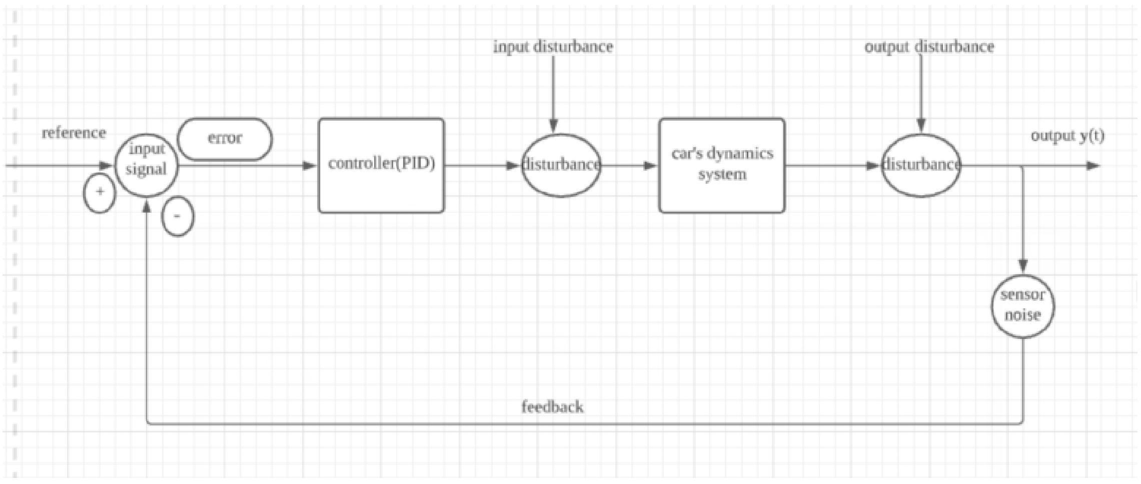


Figure 2.1: block diagram

We need to design our car's dynamics. Let's say the resistive force of air is porprotional to the speed of the car.

$$\frac{dv}{dt} \cdot m + kv = F$$

$$F(s) = (ns + k)V(s)$$

$$G(s) = \frac{V(s)}{F(s)} = \frac{1}{ms+k}$$

That's our transfer function for the car's dynamic system. $V(s)$ is our output, F is the input. So we could derive the transfer function. As we could see, the reference is the setpoint, the desired value. The controller is a system that decide how we deal with the error, or input in annother word. The car's dynamics block is the system that transfer our input signal, after controller, to the car's dynamics signal, and make the car's enginge to react. Then it output a value, detected by the sensor, and feed back to the start, to deal with the new error value. That's a simple introduction of this block diagram. The

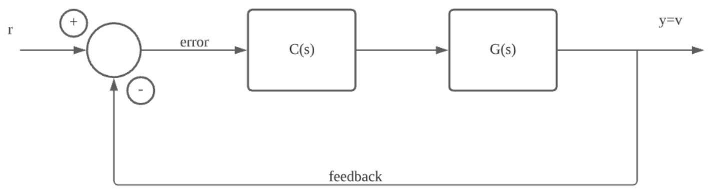


Figure 2.2: total feedback system

$G(s)$ is we calculated previously, the transfer function of the car's dynamics.

The total transfer function, for negative feed back, we could use the conclusion in previous page.

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}$$

Chapter 3

Simulation with types of controllers

3.1 Simulatino with p-control

3.1.1 What is p-control

P-control is a type of a controller that multiply a constant to our error, which get from the difference with set points and present value, and the difference will be the new input to adjust our system to try to reach the set point.

3.1.2 the p-controller

the transfer function of p-controller

The total tranfer function with p-controller will be like this:

$$\begin{aligned} C(s) &= K_p \\ T(s) &= \frac{k_p \cdot \frac{1}{ms+k}}{1 + k_p \cdot \frac{1}{ms+k}} \\ &= \frac{k_p}{m} \cdot \frac{1}{s + \frac{k+k_p}{m}} \end{aligned}$$

We let the value of the controller to be k_p. Here comes the transfer function.

$$\begin{aligned} C(s) &= K_p \\ T(s) &= \frac{k_p \cdot \frac{1}{ms+k}}{1 + k_p \cdot \frac{1}{ms+k}} \\ &= \frac{k_p}{m} \cdot \frac{1}{s + \frac{k+k_p}{m}} \end{aligned}$$

The root of the poles is $-\frac{k+k_p}{m}$, so we have a stable controller.

```
V[a+1] = v0

    error = v_set - v0

    es[a+1] = error

    u = ubias + Kc * error
```

It's a part of simple code to show that how Kc works as the constant multiplying the error.$ubias$ is the value that we want to set 0 because we want to make this system, the car, to have the exactly the same value of the setpoint value. By plotting the motion graphs with p-controller, we could easily notice that we can't reach the set point, no matter how big the Kc is. In another word, the error is steady. This error is called steady state error.

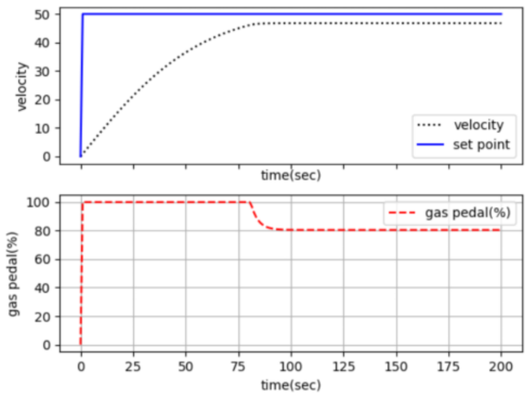


Figure 3.1: steady state error

3.1.3 the limitation of p-controller

Steady state error is the difference between desired value and present value in porprotional controller when the error gain multiply the error but make the system in equilibrium, in another word, the error will stay constant which the situation is not what we want. In our case, the result of Kc multiply the error is the force that the car need to reach the equilibrium, so the car will have a constant speed, and the error will still exist.

What is PID control?

Another difference between the two block diagrams showed above, the 'controller'. The PID control is the controller. We will go through the PID control in this subsection.
'PID' stands for proportional, integral and derivative. It's a controller to deal with the difference between the reference signal and the feedback value with these three methods. In the previous simulation, we only implement the p-control, which is the proportional controller. PID control is a more complicated

controller to solve our problem, like steady state error.

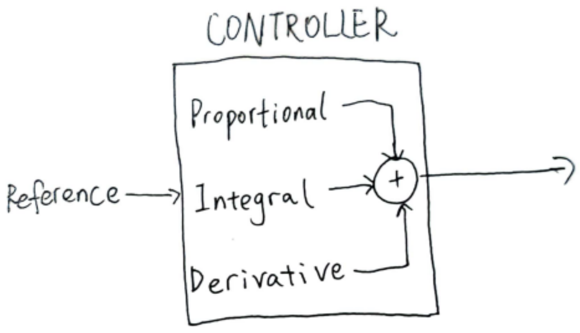


Figure 3.2: Controller

We could involve all these three types of controllers in our system. However, we try to involve p and i controller first to try to solve the steady state error.

3.2 PI control

PI control is replace the p-controller with combination of p-controller and I-controller. *I* stands for Integral. It means that we do an integration with the error value, to sum them up, so that we could eliminate the steady state error. It explained below.

$$u(t) = K \int_0^t e(\tau) d\tau$$

The K value is the key. When the time goes, the error values will be sumed up. So the system will react to increase the input signal. That integral controller makes sure that the car will adjust its speed even though the error value is small. It tries to reach the setpoint value. When the present speed reaches the setpoint, the porprotional controller will be zero, but the integral will stay constant, because of the sum of the error will not increase as the time goes. It means when the error stays constant, it will remain that speed.
According to our total feedback block diagram.

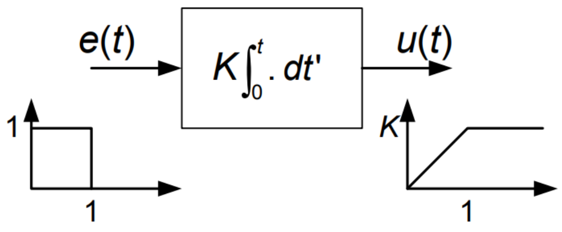


Figure 3.3: Integration of unit square signal

Taking Laplace transform and using the relationship for the Laplace transform of the integral of a signal, we obtain:

$$U(s) = K\frac{1}{s}E(s)$$

$$U(s) = C(s)E(s)$$

The transfer function will be like:

$$C(s) = \frac{K}{s}$$

we use python to simulate it.

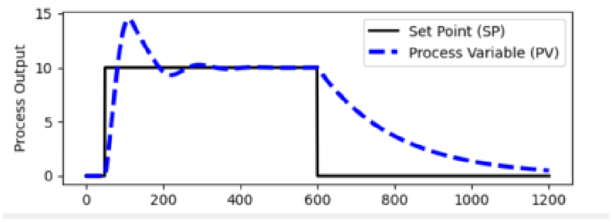


Figure 3.4: Simulation with PI control

3.2.1 overshoot

Overshoot is the situation that present value is higher than the set value, it usually happens when we run the PI control, the combination of porprotional and integral control. Overshoot happens because the property of Integral control. In integral control, the controller sum the past errors up by integration and multiple an Kc/T value, which are constant and could change then to make the process more aggresive or sluggish. Overshoot is dangerous in our cuise control car system.

3.3 PID control

3.3.1 I controller

$$u(t) = T_d \frac{de}{dt}$$

This equation means the controller differentiate the error and get a value to adjust the system behaviour. The combination of PID controller is $u(t) = Ke(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de}{dt}$ The advantage of D-controller is when the error value decresing, the differentiation of the error is negative, that gives an negative input signal to the car's dynamics, which allows the car to deccelerate to avoid overshoot, which solve the problem of the PI control. Here's the transfer function of d-controller.

$$U_{(t)} = \frac{de(t)}{dt} \cdot K_D$$
$$U(s) = K_D \cdot SE(s)$$
$$\frac{U_{(s)}}{E(s)} = K_D \cdot S$$

You could see it's similiar to integral controller, it's because integral is the quite like inverse of differentiation. We apply PID controller to our program. It could eliminate the error caused by the PI controller.

Chapter 4

Conclusion

As all these errors were solved one by one by using more complicated controller, the cruise control system could work as we expected.