**EE470 Understanding Your Plant**



Created by Walter Heth

Date:6/23/13

Table of Contents

[1 Introduction 3](#_Toc365536396)

[2 Apparatus 3](#_Toc365536397)

[3 Mathematics 3](#_Toc365536398)

[3.1 Circuit equation 3](#_Toc365536399)

[3.2 Wave Integration example 4](#_Toc365536400)

[3.3 Physical Integration example 5](#_Toc365536401)

[3.4 Mathamatics for example 5](#_Toc365536402)

[4 More Accurate example 5](#_Toc365536403)

[4.1 Frictional Forces 5](#_Toc365536404)

[4.2 Mission Briefing 5](#_Toc365536405)

[5 Simulator 6](#_Toc365536406)

[5.1 Understanding the Plant Simulation 6](#_Toc365536407)

[5.2 Getting Started 8](#_Toc365536408)

[6 Manual Control 9](#_Toc365536409)

[6.1 Assembling your Plant 9](#_Toc365536410)

[6.2 Expected Results 10](#_Toc365536411)

[7 Plant Tester Code 11](#_Toc365536412)

# 1 Introduction

Understanding the plant itself and how it represents an object is space is only part of what is needed to not only complete the lab but provide the knowledge and confidence necessary to send a signal into the plant and receive an expected output that will not damage the circuit. This document goes over the math behind the double integrator using a real world example, a more accurate example of the objective, a simulator controlling the plant manually, how we will incorporate this plant into our circuit, and code provided to run a physical imitation of the simulator to verify that the plant behaves as it should. This information is meant to be all inclusive, please do not skip sections.

# 2 Apparatus

1. Background of Calc 1-2, simple integration over an arbitrary range
2. Basic understanding of DSP and Fourier transform/ S transform
3. A strong imagination
4. Internet connection and latest java update to run simulation.
5. Arduino Uno, EE470 Shield, double integrator plant

# 3 Mathematics

## 3.1 Circuit equation

Observing one of the two integrators, assume that the circuit in Figure 3.1 is represented by the equation below where f(t) is the +input signal, a to b is the range we are integrating over, dt denotes the unit we are integrating over in which case will be time and F(t) is our output signal. F(s) is a direct table conversion of an integral from time domain.



*Fig 3.1 Integrating circuit. Simplified versions of this circuit may exclude R2 and maybe R3*

*Image not created by Walter, stolen from the internet*

## 3.2 Wave Integration example

The waves to be used in our circuit are shown below in Figure 3.2. This is just one arbitrary example of an input and resulting output to keep things simple and help build an understanding. In all parts there are 4 sections (two cycles). In part a to b, a positive unit step is integrated. This results in a positive linear output. In the second part we integrate an equal area of a negative unit step. After one cycle we see that this results in 0 at our output. The second cycle behaves exactly the same. For b to c we know that integrating ramps gives an exponential output, but why does this look so strange. When integrating over the portion of the ramp that is close to 0, the output is summing up small values. Therefore the sum of small values is equal to a small value. The output increases significantly only when the peeks of the triangular wave are integrated. Notice again when the triangular wave reaches 0 after one cycle that the output does not increase nor decrease. Since the triangular wave only has positive values it is correct that the integration/output will only be constant or increase over time.



*Fig 3.2 The simplified example of a square wave being double integrated*

## 3.3 Physical Integration example

Now jump in a car, not physically. Imagine a car at rest. Pressing down on the foot pedal and holding it in place is equivalent to accelerating at a constant rate. This is the positive portion of part a and results with the velocity of the vehicle slowly increasing over time. The negative portion of part a is equivalent to deceleration or hitting the break. In this case the deceleration is of equal magnitude and over the same amount of time resulting in the car coming to a complete stop after one cycle. Part c is the position of the vehicle. The velocity is always positive resulting in a positive disposition. When velocity is high, the car is covering more distance in a shorter amount of time.

## 3.4 Mathamatics for example

This section only demonstrates the integration of units to explain how the double integrator mathematically represents our example. There are no values in the equations.

# 4 More Accurate example

## 4.1 Frictional Forces

The plant itself performs pure integration and nothing else. In this sense the force of friction or air resistance which a car experiences is not represented here. The circuit behaves as an object in a frictionless, airless environment; space. For a point of interest, if the plant were to represent a car what would need to be added?

##  4.2 Mission Briefing

What is the point of having a controller to perform an action that nearly every human is familiar with. Well here is a very exclusive and sci-fi influenced example. Assume the following:

1. It’s the future, and the nation is in control of a mining outpost near an asteroid field.
2. The outpost and asteroids are relatively stationary to each other.

The objective is to direct the spaceship between points A(the outpost) and B(an asteroid) as quickly as possible. Under manual control it is very difficult to maneuver a vehicle in space. The upcoming simulation will make this very apparent. Programming a PID controller will make this task more manageable and leave little room for human error.

# 5 Simulator

This section is not completely necessary. If updating java or accepting the java applet to run is undesired there is a picture below that explains what is happening. It is very helpful to see how quickly the plant may respond and how it will behave before plugging it into a real world situation so if there is a more preferable then try to duplicate the simulation in that program.

## 5.1 Understanding the Plant Simulation

In the circuit, a voltage between +/-12v is injected. To simplify things only the max and min will be allowed. Green=positive, Red=Negative, Grey=0V, and the yellow dots represent current flow. The primary focus will be on the 3 scopes at the bottom. The first represents the acceleration, 2nd velocity, 3rd is position. Use the switches for +/-12v to accelerate in the positive or negative direction. The numbers in the scope represent the min and max values within the scope. Check Figure 5.2 for a quick show.



*Fig* *5.1 Falstad simulator of double integrator*

To get the result below the reset button was pressed and the position set to -1000uV which we can view as

-1000 feet/km/miles. The +12V switch was held down which increases the velocity and moves the position closer to the 0 mark. 4 things to note before getting started.

1. Velocity is inverted. Anytime one of the opamp inputs is connected to ground, it will invert the signal. Since this done twice the position is re-inverted.

2. Assume anything in the nV range is 0V

3. The simulation speed can be adjusted on the right.

4. When the acceleration drops to 0v we can see that the velocity becomes constant. From Figure 3.2, integrating a constant will give a ramp meaning that without any additional input the vehicle will reach the 0 V/feet/km/miles mark and then pass it…unless accelerate in the negative direction is applied at the right amount and time. Give it a shot.



## 5.2 Getting Started

This should take 5 minutes max to set up. First go to the website
<http://falstad.com/circuit/>

You should get a window asking to either update java or run the java update. Click run and an RLC circuit will come up. There are a lot of other example circuits you can explore in the circuits tab, but for now click File> Import and copy/past the data on the next page into the Import window and click Import. This will bring up the schematic identical to Figure 5.1



*Fig5.3 Screenshot of importing data.*

$ 1 5.0E-6 10.20027730826997 50 5.0 43

a 224 208 400 208 0 15.0 -15.0 1000000.0

a 464 224 656 224 0 15.0 -15.0 1000000.0

r 464 240 464 384 0 680000.0

r 224 224 224 384 0 680000.0

c 224 112 400 112 0 1.0E-5 0.053010998862393674

c 464 112 656 112 0 1.0E-5 8.140965085660796E-4

r 144 192 224 192 0 2000000.0

w 224 112 224 192 0

w 400 112 400 208 0

w 464 112 464 208 0

w 656 112 656 224 0

s 64 160 144 160 0 1 true

s 64 224 144 224 0 1 true

w 144 192 144 160 0

w 144 192 144 224 0

R 64 160 0 160 0 0 40.0 12.0 0.0 0.0 0.5

R 64 224 0 224 0 0 40.0 -12.0 0.0 0.0 0.5

g 224 384 224 416 0

g 464 384 464 416 0

r 400 208 464 208 0 2000000.0

o 14 64 0 291 20.0 9.765625E-5 0 -1

o 8 64 0 291 0.078125 9.765625E-5 1 -1

o 10 64 0 291 0.001220703125 9.765625E-5 2 -1

# 6 Manual Control

In this section, we will physically control our circuit in the same way the simulator is run. At this point the shield will be needed to sending in a +/-12v into the plant to accelerate the vehicle forwards and backwards. The code is provided at the end of this document.

## 6.1 Assembling your Plant

The operational amplifier should be powered by +/-2.5v, create two voltage dividers and use the +/-15v to achieve this. You will probably need to connect two resistors in parallel for Z1 as there is a high voltage drop and the 1/4Watt resistor may not be able to handle the heat. For the two RC values, I started around

10uF\*1MOhm = 10 and worked my way down. Start with any non-polar capacitor between 10-0.1uF for C1 and C2. Use potentiometers for R1 and R3. If the voltage is increasing/decreasing too slowly then reduce the resistance.



## 6.2 Expected Results

Here are the photos of my input/outputs of both integrators as I try to replicate Fig. 3.2. First I send in a -12v. The voltage is inverted so we start gaining positive voltage at the output. I then send a positive 12v and the voltage begins to decrease. When the output is near 0v I set my input to 0 and the output remained a steady 0v.



Measuring across the second integrator we can see that a positive ramp causes a negative exponential output. When the input of the 2nd integrator goes back to 0v, the output of the 2nd integrator become constant as expected. I obtained these results within 20 minutes. Do not spend this much time, just send in different voltages and make sure the circuit behaves as expected.



This verifies that the plant is behaving correctly, the code is provided blow.

# 7 Plant Tester Code

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// Name : Plant Tester //

// Author : Walter Heth //

// Date : June 28, 2013 //

// Version : 1.0 //

// Function : Sends +/-12v to plant //

// Notes 1. Is compatible with EE470 Shield Rev1 and 2 //

// 2. Controls //

/\* r = 12v hold

 f = 0v hold

 v = -12v hold

 e = 12v pulse for 500ms

 d = -12v pulse for 500ms

 w = 12v pulse for 100ms

 s = -12v pulse for 100ms

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// SPI Interface SS\_PIN(PB2), SCK\_PIN(PB5), MOSI\_PIN(PB3), MISO\_PIN

// Arduino Pin 10 13 11 12

// MCP4921 DAC SS SCK MOSI n/a

#include <SPI.h>

#include <SoftwareSerial.h>

int analogPin = 0;

const int slaveSelectPin = 10;

double Vout1 = 0;

double Vout = 0;

int timer = 0;

word sensorValue = 0;

int DACval;

byte data = 0;

char incomingByte = 0;

void setup()

{

 // set pin(s) to input and output

 pinMode(analogPin + A0, INPUT);

 // set the slaveSelectPin as an output:

 pinMode (slaveSelectPin, OUTPUT);

 // Initializes the SPI bus by setting SCK, MOSI, and SS to outputs,

 // pulling SCK and MOSI low, and SS high.

 SPI.begin();

 Serial.begin(115200);

 Serial.println("started");

}

void loop()

{

 if (Serial.available() > 0)

 {

 // read the incoming byte:

 incomingByte = (char)Serial.read();

 if (incomingByte == 'r')

 {

 //Serial.println("12v hold");

 Vout = 12;

 }

 if (incomingByte == 'f')

 {

 //Serial.println("0v hold");

 Vout = 0;

 }

 if (incomingByte == 'v')

 {

 //Serial.println("-12v hold");

 Vout = -12;

 }

 if (incomingByte == 'e')

 {

 //Serial.println("12v pulse 500ms");

 Vout = 12;

 timer = 500;

 }

 if (incomingByte == 'd')

 {

 //Serial.println("-12v pulse 500ms");

 Vout = -12;

 timer = 500;

 }

 if (incomingByte == 'w')

 {

 //Serial.println("12v pulse 100ms");

 Vout = 12;

 timer = 100;

 }

 if (incomingByte == 's')

 {

 //Serial.println("-12v pulse 100ms");

 Vout = -12;

 timer = 100;

 }

 }//end if serial available

 if (Vout != 5) //5v if there is no input and we shouldnt write to the device

 {

 SendVoltage(Vout);

 if (timer > 0)

 {

 delay(timer);

 SendVoltage(0);

 }

 timer = 0;

 Vout = 5;

 }

 timer = 0;

 Vout = 5;

 incomingByte = 0;

}// end loop

void SendVoltage(double Vout1)

{

 DACval=(((-Vout1/4.8)+2.5)\*4095)/5;

 //Serial.print("sent ");

 //Serial.println(DACval);

 digitalWrite(slaveSelectPin, LOW);

 data = highByte(DACval);

 data = 0b00001111 & data;

 data = 0b00110000 | data;

 SPI.transfer(data);

 data = lowByte(DACval);

 SPI.transfer(data);

 digitalWrite(slaveSelectPin, HIGH);

 delay(1000);

}