INSTRUMENTATION ASSESMENT

Assessment 2 – Chapters 4&6

Abstract

Assessing knowledge of Chapters 4 and 6 in Instrumentation and Experimental Methods.

Justin Dyer

J00655685

Honor Pledge

I, Justin Dyer, have read the description above and acknowledge that I did not contact another human being whether digitally or face to face to receive or give help on this quiz. I acknowledge that the answers below are in my own words and reflect my understanding of the material.

DIGITAL SIGNATURE Justin Dyer

DATE 4/27/22

Problem 1:

1. Assume you have 4 ADCs each with varying bits. They have 4, 8, 16 and 32 bits respectively. The input voltage is 1.28 V and the input range is 0-5V.

a. First, compute the Digital output of each Analog to digital converter
b. Convert the digital output back to voltage using the voltage range / 2^N where N is the number of bits. Do this for all 4 ADCs.
c. Compute the error between the measured voltage and the actual input voltage for all 4 ADCs
d. Plot the error as a function of N using Python.

Solution:

To begin this problem, I want to lay out my givens in an organized way. I decided to set this up in columns to better follow the steps of this process. Below is a representation of my setup:

V_{in} = 1.28 V V_{upper} = 5 V V_{lower} = 0 V

N_4 = 4-bit N_8 = 8-bit N_{16} = 16-bit N_{32} = 32-bit

This is the setup I will use to solve the problem in an organized manner.

The relationship that an analog digital converter uses to relate an input voltage to an output digital signal can be expressed as follows:

$$\tilde{V}_{in} = D_o\left(\frac{V_{upper}-V_{lower}}{2^N}\right) + V_{lower} \quad \text{(Eqn 1.1)}$$

where V_{in} is the input voltage, D_o is the digital output as a truncated integer, V_{upper} is the upper input voltage range, V_{lower} is the lower input voltage range, and N is the number of bits. Solving for the digital output one gets the following equation:

$$D_o = \frac{\tilde{V}+V_{lower}}{\left(\frac{V_{upper}-V_{lower}}{2^N}\right)} \quad \text{(Eqn 1.2)}$$

Let's apply this equation to each column of our setup:

$$D_{o,4} = \frac{\tilde{V}+V_{lower}}{\left(\frac{V_{upper}-V_{lower}}{2^N}\right)} \quad D_{o,8} = \frac{\tilde{V}+V_{lower}}{\left(\frac{V_{upper}-V_{lower}}{2^N}\right)} \quad D_{o,16} = \frac{\tilde{V}+V_{lower}}{\left(\frac{V_{upper}-V_{lower}}{2^N}\right)} \quad D_{o,32} = \frac{\tilde{V}+V_{lower}}{\left(\frac{V_{upper}-V_{lower}}{2^N}\right)}$$

$$= \frac{1.28}{\left(\frac{5}{2^4}\right)} \qquad = \frac{1.28}{\left(\frac{5}{2^8}\right)} \qquad = \frac{1.28}{\left(\frac{5}{2^{16}}\right)} \qquad = \frac{1.28}{\left(\frac{5}{2^{32}}\right)}$$

$D_{o,4} = 4$ $D_{o,8} = 65$ $D_{o,16} = 16777$ $D_{o,32} = 1099511628$

To convert these digital outputs back to voltage we will use Eqn 1.1 as follows:

$$\tilde{V}_{in,4} = 4\left(\frac{5}{2^4}\right) \quad \tilde{V}_{in,8} = 65\left(\frac{5}{2^8}\right) \quad \tilde{V}_{in,16} = 16777\left(\frac{5}{2^{16}}\right) \quad \tilde{V}_{in,32} = D_{o,32}\left(\frac{5}{2^4}\right)$$

$V_{in,4} = 1.25$ V $V_{in,8} = 1.27$ V $V_{in,16} = 1.27998$ V $V_{in,32} = 1.28$ V

To find the error, we will compare our computed input voltage to the actual input voltage with the following equation:

$$\%error = \frac{V_{actual}-V_{computed}}{V_{actual}} \times 100\% \qquad \text{(Eqn 1.3)}$$

$$\%error_4 = \frac{1.28-1.25}{1.28} \times 100\% = 2.34375\%$$

$$\%error_8 = \frac{1.28-1.27}{1.28} \times 100\% = 0.78125\%$$

$$\%error_{16} = \frac{1.28-1.27998}{1.28} \times 100\% = 0.0015625\%$$

$$\%error_{32} = \frac{1.28-1.28}{1.28} \times 100\% = 0\%$$

One could infer from this that by increasing the number of bits, the less error and more accurate readings can be made. To show this, I wrote the following code to plot the relationship of error versus bit size:

```
"""
Created on Tue Apr 19 12:53:30 2022

@author: Justin
"""
```

```
##EDIT
##Instrumentation Assesment 2 Problem 1

##Import needed modules
import numpy as np
import matplotlib.pyplot as plt

##Givens
Vin = 1.28
Vup = 5
Vlow = 0
bits = np.array([4,8,16,32])
#Solution calculated by hand
error_hand = np.array([2.34375,0.78125,0.0015625,0])

#Print Givens
print('Input voltage = ', Vin)
print('Voltage range = ', Vlow,'-',Vup,'V')

#Take input voltage and bits and convert to digital
digitalout = []
for i in [4,8,16,32]:
    out = np.trunc((Vin-Vlow)/((Vup-Vlow)/(2**i)))
    digitalout.append(out)
    print('Digital Output of', i, 'bit: ', out)

#Take digital signal back to voltage
count = 0
Vreversed = []
for i in [4,8,16,32]:
    if count == 0:
        count = 0
    Vrev = (digitalout[count]*((Vup-Vlow)/2**i))+Vlow
    count = count + 1
    Vreversed.append(Vrev)
    print('Digital signal converted to voltage of',i,'bit: ', Vrev, 'V')

#Compute error
error = []
count = 0
for i in Vreversed:
    e = ((Vin-i)/Vin)*100
    error.append(e)
    print('Error of ',bits[count], 'bit: ', e,'%')
    count = count+1
```

```
#Plot
plt.plot(bits,error_hand,'r-', label= 'Hand Computed')
plt.plot(bits,error,'b-',label= 'Python Computed')
plt.xlabel('N (number of bits)')
plt.ylabel('%error from actual value')
plt.title('Error as function of bits')
plt.legend()
plt.grid()

#Show plot
plt.show()
```

The output of this script yields the following:

Input voltage = 1.28

Voltage range = 0 - 5 V

Digital Output of 4 bit: 4.0

Digital Output of 8 bit: 65.0

Digital Output of 16 bit: 16777.0

Digital Output of 32 bit: 1099511627.0

Digital signal converted to voltage of 4 bit: 1.25 V

Digital signal converted to voltage of 8 bit: 1.26953125 V

Digital signal converted to voltage of 16 bit: 1.2799835205078125 V

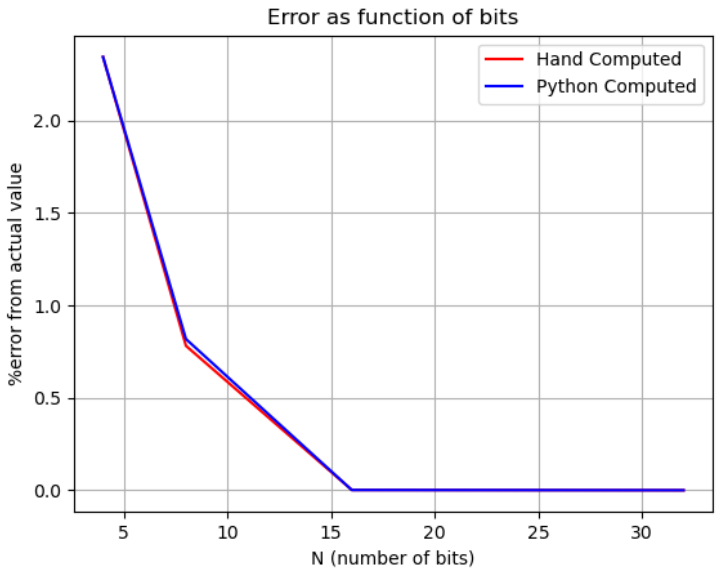
Digital signal converted to voltage of 32 bit: 1.279999999096617 V

Error of 4 bit: 2.343750000000002 %

Error of 8 bit: 0.8178710937500021 %

Error of 16 bit: 0.0012874603271505192 %

Error of 32 bit: 7.05767909392474e-08



[Click here to redirect to GitHub to use code.]

2. One of my undergraduate students was attempting to estimate the mean and standard deviation of a magnetometer.
a. Using the spreadsheet linked below, create 3 histograms of the three columns of data which represent magnetic field strengths along the X,Y and Z axes. Include your code and your histograms
b. Report the mean and standard deviation of each axis

To solve this problem, I put the data into a text file and imported it into the code displayed below that plots a histogram and calculates the mean and standard deviation for each axis of the magnetometer:

```
"""
Created on Tue Apr 19 14:06:17 2022

@author: Justin
"""

##Instrumentation Assesment 2 Problem 2

##Import needed modules
import numpy as np
import matplotlib.pyplot as plt

#Load data
data = np.loadtxt('INSTtest2problem2data.txt')

#Assign variables to data
MAGX = data[:,0]
MAGY = data[:,1]
MAGZ = data[:,2]

#Define function to answer part 2
def get_solution(x,y):
    print('Mean of ',y, 'data = ',np.mean(x))
    print('Standard Deviation of ',y, 'data = ',np.std(x))

#Get solution to part 2
get_solution(MAGX, 'x - axis')
get_solution(MAGY, 'y - axis')
get_solution(MAGZ, 'z - axis')

#Satisfy part 1 by plotting histograms
#X-axis Histogram
plt.figure()
plt.hist(MAGX)
```

```
plt.title('Magnetometer x-axis Histogram')
plt.xlabel('Value')
plt.ylabel('Number of Occurances')
plt.grid()

#Y-axis Histogram
plt.figure()
plt.hist(MAGY)
plt.title('Magnetometer y-axis Histogram')
plt.xlabel('Value')
plt.ylabel('Number of Occurances')
plt.grid()

#Z-axis Histogram
plt.figure()
plt.hist(MAGZ)
plt.title('Magnetometer z-axis Histogram')
plt.xlabel('Value')
plt.ylabel('Number of Occurances')
plt.grid()

#Show plots
plt.show()
```

[Click here for access to Problem 2 code]

The output to the code above is as follows:

Mean of x - axis data = -26.772955082742314

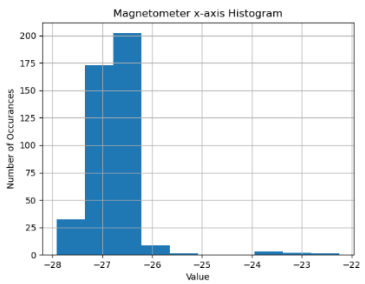
Standard Deviation of x - axis data = 0.5610640982386256

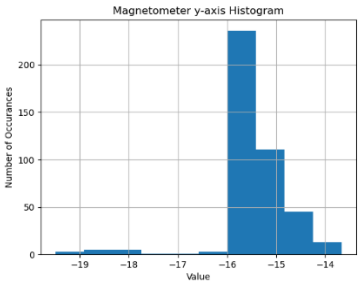
Mean of y - axis data = -15.449976359338061

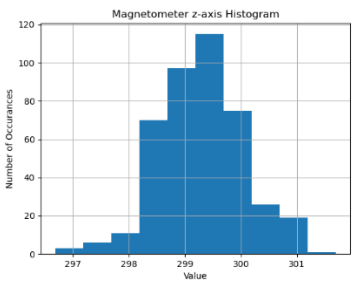
Standard Deviation of y - axis data = 0.730836982148947

Mean of z - axis data = 299.29789598108744

Standard Deviation of z - axis data = 0.7147057577418389







I have been working on making python scripts to open an application. I incorporated this into this assessment. Below is a video of me using the application and walking through the process. Also, I listed links to each file needed to run this yourself.

Tutorial Video

Problem 1 Function (INSTtest2problem1.py)

Problem 2 Function (INSTtest2problem2.py)

Problem 2 Data (INSTtest2problem2data.txt)

Application (INSTtest2App.py)

Save all these in the same location and make sure the file names match the scripts inputs. Run the application.

Sources:

Justin Dyer's Notes for Instrumentation and Experimental Methods Spring 2022

Spyder3(A python interpreter)

Justin Dyer's Central Intelligence (brain)