DSP For Scientists and Engineers Notes

Philip Tracton

April 21, 2017

Contents

Listings

List of Tables

1 The Breadth and Depth of DSP

These are my personal notes from learning DSP using The Scientists and Engineers Guide to Digital Signal Processing, Second Edition.

The first chapter goes over the history and where you can find use for DSP work. The book writes some psuedo-code in a "BASIC" like language. I will be attempting to move it over to Python. Matlab would be better but I don't have it.

2 Statistics, Probability and Noise

2.1 Signals and Graph Terminology

2.1.1 Definitions

- **Signal** is how one parameter is related to another parameter
- **Continuous Signal** is if BOTH parameters can assume a continuous range
- **Discrete Signal** is if BOTH parameters are quantized in some manner
- **Time Domain** is if the X axis (the independent variable) is time
- **Frequency Domain** is if the X axis (the independent variable) is frequency

2.1.2 Concepts

- The two parameters of a signal are not interchangeable
- The parameter on the Y axis is a function of the one on the X axis
- Mathematicians tend to do 1-N, everyone else does 0-(N-1)

2.2 Mean and Standard Deviation

2.2.1 Mean

- **Mean** μ is the average of the signal. Add all samples together and divide by N. In electronics this is the DC (direct current) value.

$$\mu = \frac{1}{N} \sum_{i=1}^{N-1} x_i$$

```python
1   def Mean(self):
2       """
3       Calculate the mean of a list of values.
4       The self.samples list should be set when instantiating
5       this instance.
6       """
7       mean = 0
8       for x in self.samples:
9           mean = mean + x
10      mean = mean/len(self.samples)
11      return mean
```

2.2.2 Standard Deviation and Variance

- **Average Deviation** is not commonly used. Sums up all the deviations, from the mean, for each sample and divided by the number of samples. Use absolute values for deviation otherwise differences could cancel out.

- **Standard Deviation** averages the power. This is the AC portion of the signal.

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N-1} (x_i - \mu)^2}$$

```python
def StandardDeviation(self):
    """
    Calculate the standard deviation of a list of values.
    The self.samples list should be set when instantiating
    this instance.
    """
    mean = self.Mean()
    std = 0.0
    for x in self.samples:
        std = std + math.pow((x - mean), 2)
    std = std / (len(self.samples) - 1)
    std = math.sqrt(std)
    return std
```

- **Variance** is commonly used in statistics. Notice variance and standard deviation both divide by N-1, not N!

$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^{N-1} (x_i - \mu)^2$$

```python
def Variance(self):
    """
    Calculate the variance of a list of values.
    The self.samples list should be set when instantiating
    this instance.
    """
    return math.pow(self.StandardDeviation(), 2)
```

- **Root Mean Square (rms)** measures both the AC and DC components.

$$x_{rms} = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (x_i)^2}$$

6

2.2.3 Running Statistics

- **Running Statistics** is often needed. In this situation we want to recompute mean and standard deviation of new signal added in without redoing all of the calculations

$$\sigma^2 = \frac{1}{N-1}\left(\sum_{i=0}^{N-1}(x_i)^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1}x_i\right)^2\right)$$

```python
def RunningStatistics(self):
    """
    Calculate the mean, variance and std while running through a list of
    values. The self.samples list should be set when instantiating
    this instance.
    """
    mean = 0
    variance = 0
    std = 0
    temp_sum = 0
    sum_squares = 0
    N = len(self.samples)
    for x in self.samples:
        temp_sum = temp_sum + x
        sum_squares = sum_squares + math.pow(x, 2)
        mean = temp_sum/N
        variance = (sum_squares - (math.pow(temp_sum, 2)/N)) / (N - 1)
        std = math.sqrt(variance)
        print("RunningStatistics: Mean {} Variance {} STD {}".format(
            mean, variance, std))
    return mean, variance, std
```

- In some situations mean decribes what is being measured and standard deviation measures noise

- **Signal to Noise Ration (SNR)** is a comparison of mean to standard deviation

$$SNR = \frac{\mu}{\sigma}$$

- **Coefficient of Variance (CV)** is the standard deviation divided by the mean and multiplied by 100%.

$$CV = \frac{\sigma}{\mu} * 100\%$$

- High SNR and Low CV is a good signal!

3 ADC and DAC

3.1 Definitions

3.2 Concepts

4 Appendix A – Code Listing

4.1 DSP Source Code

```python
1   #! /usr/bin/env python3
2
3   import math
4
5
6   class DSP():
7       """
8       DSP Demonstration class that implements the code
9       from DSP For Scientists and Engineers, Second Edition
10      """
11
12      def __init__(self, samples=None):
13          """
14          """
15          self.samples = samples
16          return
17
18      def Mean(self):
19          """
20          Calculate the mean of a list of values.
21          The self.samples list should be set when instantiating
22          this instance.
23          """
24          mean = 0
25          for x in self.samples:
26              mean = mean + x
27          mean = mean/len(self.samples)
28          return mean
29
30      def StandardDeviation(self):
31          """
32          Calculate the standard deviation of a list of values.
33          The self.samples list should be set when instantiating
34          this instance.
35          """
36          mean = self.Mean()
37          std = 0.0
38          for x in self.samples:
39              std = std + math.pow((x - mean), 2)
40          std = std / (len(self.samples) - 1)
41          std = math.sqrt(std)
42          return std
43
44      def Variance(self):
45          """
46          Calculate the variance of a list of values.
47          The self.samples list should be set when instantiating
48          this instance.
49          """
50          return math.pow(self.StandardDeviation(), 2)
51
52      def RunningStatistics(self):
```

```python
53              """
54              Calculate the mean, variance and std while running through a list of
55              values. The self.samples list should be set when instantiating
56              this instance.
57              """
58              mean = 0
59              variance = 0
60              std = 0
61              temp_sum = 0
62              sum_squares = 0
63              N = len(self.samples)
64              for x in self.samples:
65                  temp_sum = temp_sum + x
66                  sum_squares = sum_squares + math.pow(x, 2)
67                  mean = temp_sum/N
68                  variance = (sum_squares - (math.pow(temp_sum, 2)/N)) / (N - 1)
69                  std = math.sqrt(variance)
70                  print("RunningStatistics: Mean {} Variance {} STD {}".format(
71                      mean, variance, std))
72              return mean, variance, std
```

4.2 DSP Test Code

```python
#! /usr/bin/env python3

"""
Check Results:
http://www.calculator.net/standard-deviation-calculator.html?numberinputs=0%2C+1%2C+2%2C+3%2C+
"""


import DSP


if __name__ == "__main__":
    samples = [x for x in range(100)]
    dsp = DSP.DSP(samples)
    print(samples)
    print("Mean {} ".format(dsp.Mean()))
    print("StandardDeviation {} ".format(dsp.StandardDeviation()))
    print("Variance {} ".format(dsp.Variance()))
    mean, variance, std = dsp.RunningStatistics()
```