DSP For Scientists and Engineers Notes

Philip Tracton

April 21, 2017

Contents

Listings

List of Tables

%#+INCLUDE: "../Chapter01/chapter01.org"

1 Chapter 2 – Statistics, Probability and Noise

1.1 Signals and Graph Terminology

1.1.1 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

1.1.2 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)

1.2 Mean and Standard Deviation

1.2.1 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.

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

```
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
```

1.2.2 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.

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

```
1   def StandardDeviation(self):
2       """
3       Calculate the standard deviation of a list of values.
4       The self.samples list should be set when instantiating
5       this instance.
6       """
7       mean = self.Mean()
8       std = 0.0
9       for x in self.samples:
10          std = std + math.pow((x - mean), 2)
11      std = std / (len(self.samples) - 1)
12      std = math.sqrt(std)
13      return std
```

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

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

```
1   def Variance(self):
2       """
3       Calculate the variance of a list of values.
4       The self.samples list should be set when instantiating
5       this instance.
6       """
7       return math.pow(self.StandardDeviation(), 2)
```

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

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

1.2.3 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

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

```
1
2     def RunningStatistics(self):
3         """
4         Calculate the mean, variance and std while running through a list of
5         values. The self.samples list should be set when instantiating
6         this instance.
7         """
8         mean = 0
9         variance = 0
10        std = 0
11        temp_sum = 0
12        sum_squares = 0
13        N = len(self.samples)
14        for x in self.samples:
15            temp_sum = temp_sum + x
16            sum_squares = sum_squares + math.pow(x, 2)
17            mean = temp_sum/N
18            variance = (sum_squares - (math.pow(temp_sum, 2)/N)) / (N - 1)
19            std = math.sqrt(variance)
20            print("RunningStatistics: Mean {} Variance {} STD {}".format(
21                mean, variance, std))
22        return mean, variance, std
```

1.3 Signal vs. Underlying Process

1.4 The Histogram, PMF and PDF

1.5 The Normal Distribution

1.6 Digital Noise Generation

1.7 Precision and Accuracy

%#+INCLUDE: "../Chapter03/chapter03.org"

2 Appendix A – Code Listing

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

import math


class DSP():
    """
    DSP Demonstration class that implements the code
    from DSP For Scientists and Engineers, Second Edition
    """

    def __init__(self, samples=None):
        """
        """
        self.samples = samples
        return

    def Mean(self):
        """
        Calculate the mean of a list of values.
        The self.samples list should be set when instantiating
        this instance.
        """
        mean = 0
        for x in self.samples:
            mean = mean + x
        mean = mean/len(self.samples)
        return mean

    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

    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)

    def RunningStatistics(self):
        """
        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
```