ECEN321 : Noise
Lab 2 Submission

Daniel Eisen : 300447549

May 10, 2020

1 Introduction

Noise, when pertaining to taking measurements, represents an uncertainty of the true value of the measured parameter. When these measurements further used, i.e functions are applied this error propagates through the process. Hence the necessity in characterising and possibly mitigating the noise and/or its effect.

2 Theory

1.

2.

3.

4.

3 Results

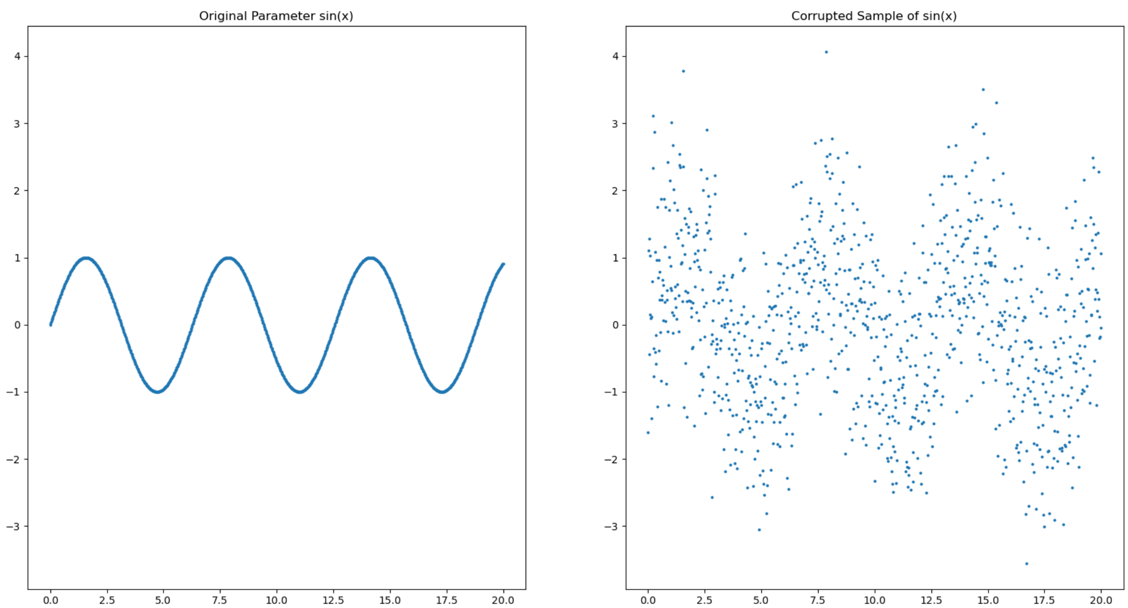
1. Amplified Noise

```
>>> q1_amp()
Signal Sample: mean=2.9693316094834987, std=1.945355127360896
Amplifier Sample: mean=29.693316094834984, std=19.45355127360896
```

2. Averaged Measurements

```
>>> q2_avg()
Signal: mean=2.8812649845429417, std=2.0473458742648933
Averaged Signal: mean=2.980104185477144, std=0.4957774846945985
```

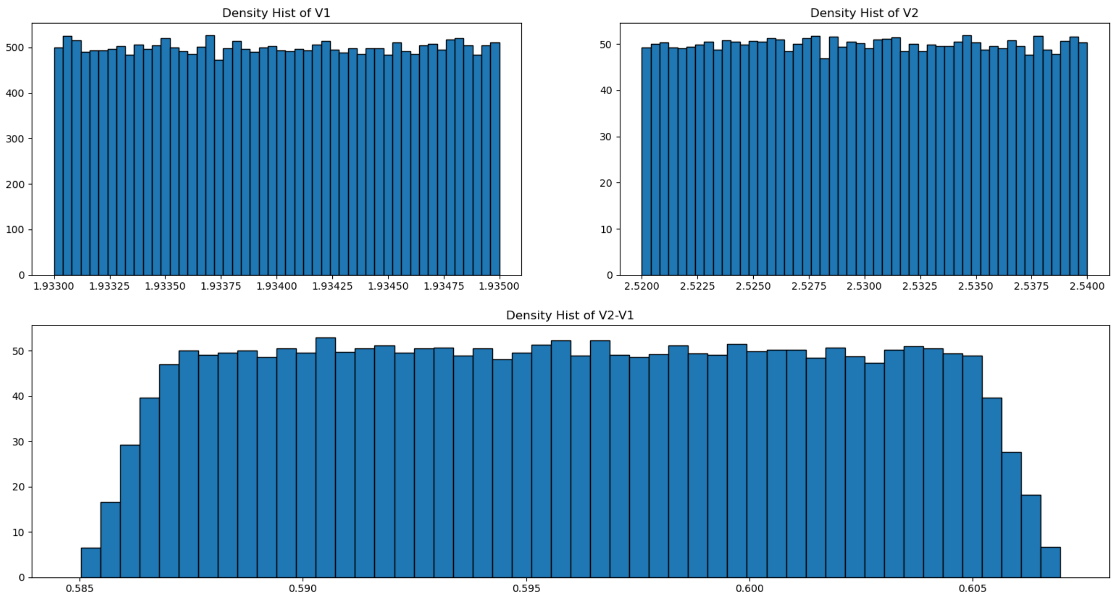
3. Covariance and Correlation




```
>>> q3_cov_corr()
Covariance of Original and Noisy signal:
0.48332539327802315

Correlation Coefficient:
0.5534612318519198
```

4. Combined Uncertainty




```
>>> q4_combined_uncertainties()
V2-V2 = 0.5959999999999999+-0.01004987562112089
```

Appendix

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats


def q1_amp():
    mean_noise = 0
    std_noise = 2
    DC = 3

    noise = mean_noise + std_noise * (np.random.randn(1000))
    sig = DC + noise
    print(f"Signal Sample: mean={np.mean(sig)}, std={np.std(sig)}")

    amped = 10 * sig
    print(f"Amplifier Sample: mean={np.mean(amped)}, std={np.std(amped)}")


def q2_avg():
    mean_noise = 0
    std_noise = 2
    DC = 3

    noise = mean_noise + std_noise * (np.random.randn(16, 1000))
    sigs = DC + noise

    print(f"Signal: mean={np.mean(sigs[0])}, std={np.std(sigs[0])}")
    normed = np.sum(sigs, axis=0) / 16
    print(f"Averaged Signal: mean={np.mean(normed)}, std={np.std(normed)}")


def q3_cov_corr():
    x = np.linspace(0, 20, 1000)
    sin = np.sin(x)
    noise = np.random.randn(1000)
    sin_noisy = sin + noise

    print(f"Covariance of Original and Noisy signal:\n"
          f"{np.cov(sin, sin_noisy)[1, 0]}")
    print(f"\nCorrelation Coefficient:\n"
          f"{np.corrcoef(sin, sin_noisy)[1, 0]}\n")

    ax1 = plt.subplot(1, 2, 1)
    plt.scatter(x, sin, linewidths=0, s=8)
    plt.title("Original Parameter sin(x)")
    plt.subplot(1, 2, 2, sharey=ax1)
    plt.scatter(x, sin_noisy, linewidths=0, s=8)
    plt.title("Corrupted Sample of sin(x)")


def q4_combined_uncertainties():
    V1_dc = 1.934
    std1 = 0.001

    V2_dc = 2.53
    std2 = 0.01
```

```
V1 = V1_dc + (np.random.uniform(low=-std1, high=std1, size=100000))
V2 = V2_dc + (np.random.uniform(low=-std2, high=std2, size=100000))
V = V2 - V1

print(f"V2-V2 = {V2_dc - V1_dc}+-{np.sqrt(std2 ** 2 + std1 ** 2)}")

plt.subplot(2, 2, 1)
plt.hist(V1, bins=50, density=True, edgecolor="black")
plt.title("Density Hist of V1")

plt.subplot(2, 2, 2)
plt.hist(V2, bins=50, density=True, edgecolor="black")
plt.title("Density Hist of V2")

plt.subplot(2, 2, (3, 4))
plt.hist(V, bins=50, density=True, edgecolor="black")
plt.title("Density Hist of V2-V1")

plt.show()
```