Chapter 1: Understanding Noise in Signals — A Clean Python Walkthrough

🌟 Objective

In digital signal processing (DSP) and software-defined radio (SDR), noise is unavoidable. Understanding how to **simulate noise** helps us:

- Design and test filters
- Build robust DSP systems
- Learn signal recovery techniques
  This chapter focuses on using **Python** to simulate different types of noise and lays the groundwork for applying filters effectively.

✅ Section 1: Adding White Gaussian Noise (WGN)

🔍 What is it?
White Gaussian noise is the most common type of noise in DSP. It mimics thermal noise and affects all frequencies equally. It's random, zero-mean, and follows a bell-curve (normal) distribution.
✅ Code Walkthrough
import numpy as np
import matplotlib.pyplot as plt

Generate a clean 5 Hz sine wave

t = np.linspace(0, 1, 500) # Time axis: 0 to 1 sec, 500 samples
signal = np.sin(2 * np.pi * 5 * t) # Sine wave

Add Gaussian noise

np.random.seed(0) # For reproducibility

noise = 0.5 * np.random.randn(len(t)) # Scaled noise

noisy_signal = signal + noise # Combine signal and noise

Plot

plt.plot(t, signal, label='Clean Signal')

plt.plot(t, noisy_signal, label='With Gaussian Noise', alpha=0.7)

plt.legend()

plt.grid(True)

plt.title("White Gaussian Noise")

plt.show()

✅ What It Achieves

- Simulates realistic noise in communication and sensor systems
- Provides a baseline for testing low-pass filters
- The `np.random.randn()` function generates zero-mean, unit variance noise

✅ Section 2: Adding Impulse Noise (Spikes)

🔍 What is it?

Impulse noise consists of sudden, high-amplitude disturbances. It simulates glitches, dropouts, or digital bit errors.

✅ Code Walkthrough

Copy the clean signal

impulse_signal = np.copy(signal)

Inject spikes at random positions

num_spikes = 20

spike_indices = np.random.randint(0, len(t), size=num_spikes)

spike_values = np.random.choice([2.5, -2.5], size=num_spikes)

impulse_signal[spike_indices] += spike_values

Plot

plt.plot(t, signal, label='Clean Signal')

plt.plot(t, impulse_signal, label='With Impulse Noise', alpha=0.7)

plt.legend()

plt.grid(True)

plt.title("Impulse Noise (Random Spikes)")

plt.show()

✅ What It Achieves

- Models real-world glitches in hardware
- Helps evaluate filters like **median** or **adaptive** filters
- Simulates sharp, unpredictable noise that a moving average filter may not remove well

✅ Section 3: Adding Low-Frequency Drift (Sensor Drift)

🔍 What is it?

Low-frequency drift simulates slow, gradual changes in signal level. This often happens in sensors due to temperature or calibration changes.

✅ Code Walkthrough

Add a slow sine wave as drift

drift = 0.5 * np.sin(2 * np.pi * 0.5 * t) # 0.5 Hz drift

drifted_signal = signal + drift

Plot

plt.plot(t, signal, label='Clean Signal')

plt.plot(t, drifted_signal, label='With Low-Frequency Drift', alpha=0.7)

plt.legend()

plt.grid(True)

plt.title("Low-Frequency Drift")

plt.show()

✅ What It Achieves

- Simulates slow sensor bias
- Useful for testing **high-pass** or **baseline correction** filters
- Shows how a signal can appear shifted or warped

✅ Section 4: Adding Composite Noise

🔍 What is it?

Composite noise combines multiple noise types. This is more realistic because most real-world signals contain a mix of noise sources.

✅ Code Walkthrough

Combine Gaussian noise and low-frequency drift

composite_noise = 0.4 * np.random.randn(len(t)) + 0.3 * np.sin(2 * np.pi * 0.8 * t)

noisy_composite = signal + composite_noise

Plot

plt.plot(t, signal, label='Clean Signal')

plt.plot(t, noisy_composite, label='With Composite Noise', alpha=0.7)

plt.legend()

plt.grid(True)

plt.title("Composite Noise: Gaussian + Drift")
plt.show()
✅ What It Achieves

- Simulates real-world environmental and system noise
- Prepares you to design **robust filters**
- Great for end-to-end DSP simulations

🧩 Summary Table

Noise Type	Python Method	What It Models	Best Filter Type
Gaussian Noise	`np.random.randn()`	Thermal/electronic noise	Moving Average, FIR
Impulse Noise	Random spikes using indexing	Bit errors, sensor glitches	Median Filter
Low-Frequency Drift	Slow `np.sin()` wave	Sensor bias, DC drift	High-Pass Filter
Composite Noise	Combination of Gaussian + drift	Real-world scenarios	Multi-stage filtering

🎓 Key Learning

> "You can only filter noise that you can simulate and understand."
> Now that you've simulated noise, you're ready to:

- Build and test filters (Moving Average, FIR, etc.)
- Understand how filters respond to different kinds of interference
- Design clean processing pipelines for SDR, biomedical, audio, and more

⏳ What's Next in Chapter 2:

"The Moving Average Filter — Explained from First Principles"

- Use the noisy signals from this chapter
- Apply simple filters

- Visualize noise reduction
- Learn about convolution in DSP