Project 1: Analyze and Plot a Noisy Signal

Let’s start by coding the noisy signal example from my previous message, but this time in PyCharm. I’ll make sure it’s self-contained and explain how to run it. Since you’re coming from MATLAB, this will feel familiar—generating a signal, analyzing it, and plotting it.

Code: Noisy Signal in PyCharm

Create a new Python file (e.g., noisy\_signal.py) and paste this code:

python

import numpy as np

import matplotlib.pyplot as plt

# Step 1: Generate time array

t = np.arange(0, 10, 0.01) # 0 to 10 seconds, step of 0.01

# Step 2: Generate a noisy sine wave

frequency = 0.5 # Hz

amplitude = 3 # Arbitrary units

signal = amplitude \* np.sin(2 \* np.pi \* frequency \* t)

noise = np.random.randn(len(t))

noisy\_signal = signal + noise

# Step 3: Compute basic stats

mean\_signal = np.mean(noisy\_signal)

std\_signal = np.std(noisy\_signal)

print(f"Mean of the noisy signal: {mean\_signal:.2f}")

print(f"Standard deviation: {std\_signal:.2f}")

# Step 4: Plot the signal and the mean

plt.plot(t, noisy\_signal, label='Noisy Signal', color='blue', alpha=0.5)

plt.axhline(y=mean\_signal, color='red', linestyle='--', label=f'Mean = {mean\_signal:.2f}')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.title('Noisy Sine Wave with Mean')

plt.legend()

plt.grid(True)

plt.show()

Running in PyCharm

1. Run the Script: Click the green “Run” button (or press Shift+F10). If everything’s set up, you’ll see a plot pop up and the stats printed in the PyCharm console.
2. Fixing Issues:
   * If you get a ModuleNotFoundError for numpy or matplotlib, double-check you installed them (see step 3 in the setup above).
   * If the plot doesn’t show, ensure plt.show() is at the end.
3. Interactive Workflow: PyCharm isn’t as interactive as Jupyter for quick plotting, but you can use the “Python Console” (bottom of the window) to run commands line-by-line if needed.

PyCharm Tips for This Code

* Autocompletion: Start typing np. and PyCharm will suggest functions like arange or sin. Hit Tab or Enter to accept.
* Code Inspection: PyCharm might underline unused imports or suggest simplifications. Hover over underlines to see suggestions.
* Debugging: If something’s off (e.g., stats look wrong), set a breakpoint by clicking next to a line number, then hit the “Debug” button (Shift+F9). You can inspect variables step-by-step.

Your Turn

Run the code and experiment:

1. Change the frequency or amplitude and rerun.
2. Add the original (non-noisy) signal to the plot in a different color (e.g., green). Hint: Add another plt.plot(t, signal, label='Original Signal', color='green') before plt.show().
3. Try zooming into a specific time range (e.g., 2 to 4 seconds) using plt.xlim(2, 4).

Let me know how it goes or if you hit any bumps!