analysis

September 26, 2023

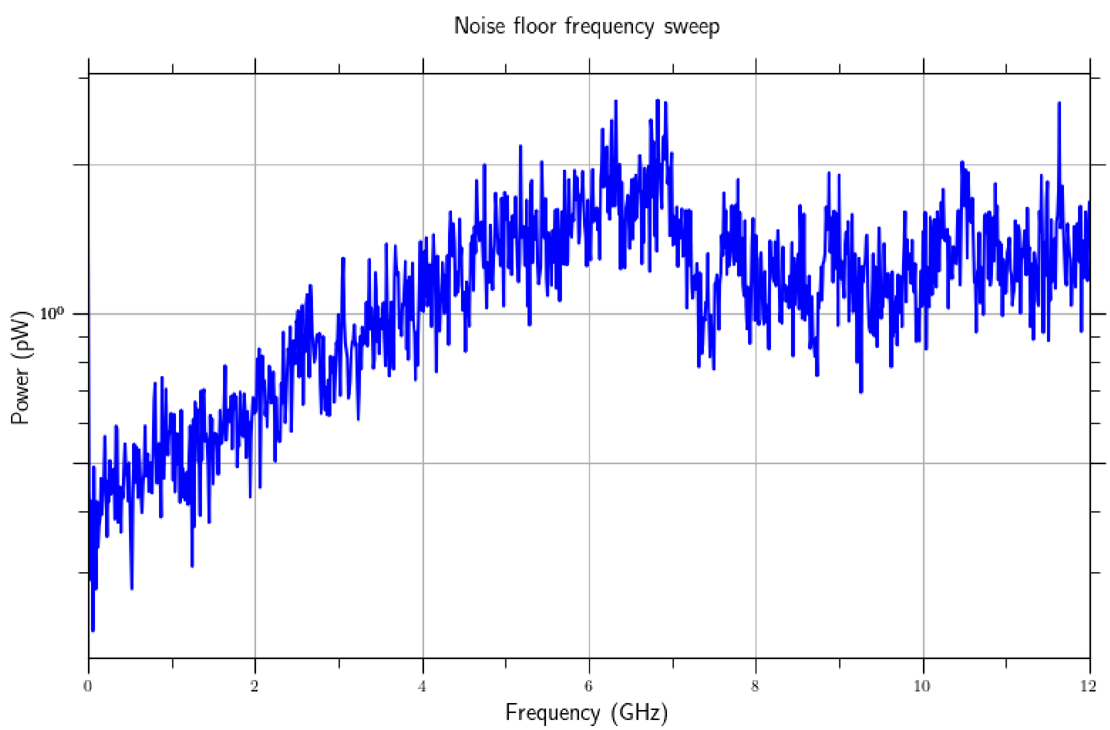
[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from matplotlib.ticker import AutoMinorLocator
import warnings

warnings.filterwarnings('ignore')
```

[2]:
```python
def sweep_plot(file_name: str, title: str = None, show: bool = False) -> None:
    data = pd.read_table(file_name, sep=';')
    data = data[["Frequency", "Power"]]

    freq = data['Frequency'] * 10**-9
    power = data['Power'] * 10**12

    # Create the plot
    plt.rcParams['text.usetex'] = True
    fig, ax1 = plt.subplots(figsize=(8, 5))
    ax1.set_xlabel("Frequency (GHz)", family="serif", fontsize=12)
    ax1.set_ylabel("Power (pW)", family="serif", fontsize=12)

    plt.semilogy(freq, power, color='blue')
    ax1.set_xlim(min(freq), max(freq))
    #ax1.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax1.yaxis.set_major_locator(ticker.LogLocator(base=10.0, subs=[1.0, 2.0, 5.
 ↪0, 10.0], numticks=20))
    # Set the minor tick marks
    ax1.xaxis.set_minor_locator(AutoMinorLocator(2))
    ax1.tick_params(axis='both', which='major', direction="out", top="on",␣
 ↪right="on", bottom="on", length=8, labelsize=8)
    ax1.tick_params(axis='both', which='minor', direction="out", top="on",␣
 ↪right="on", bottom="on", length=5, labelsize=8)

    # Adjust the plot layout
    fig.tight_layout()
    #plt.legend()
```

```
    plt.grid(True)

    if title is not None:
        plt.title(title, pad=20)

    if show:
        # Display the plot
        plt.show()
    else:
        # Save the plot to a file
        plt.savefig(file_name.replace('.csv', '.png').replace('Data', 'Plots'),␣
↪dpi=1000, format="png")
        plt.close()
```

1 Noise floor sweep from 0 to 12 GHz without load or amplifier

[3]:
```
sweep_plot('Data/noise_floor_sweep.DAT', title = 'Noise floor frequency sweep',␣
↪show=True)
```



[4]:
```
def time_plot(file_name: str, title: str = None, show: bool = False) -> None:
    data = pd.read_table(file_name, sep=';')
    data = data[["Time", "Power"]]
```

```
    time = data['Time'] * 10**3
    power = data['Power'] * 10**12

    # Create the plot
    plt.rcParams['text.usetex'] = True
    fig, ax1 = plt.subplots(figsize=(8, 5))
    ax1.set_xlabel("Time (ms)", family="serif", fontsize=12)
    ax1.set_ylabel("Power (pW)", family="serif", fontsize=12)

    plt.semilogy(time, power, color='blue', label = 'Raw')
    ax1.set_xlim(min(time), max(time))
    #ax1.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax1.yaxis.set_major_locator(ticker.LogLocator(base=10.0, subs=[1.0, 2.0, 5.
↪0, 10.0], numticks=20))
    # Set the minor tick marks
    ax1.xaxis.set_minor_locator(AutoMinorLocator(2))
    ax1.tick_params(axis='both', which='major', direction="out", top="on",␣
↪right="on", bottom="on", length=8, labelsize=8)
    ax1.tick_params(axis='both', which='minor', direction="out", top="on",␣
↪right="on", bottom="on", length=5, labelsize=8)

    mean_value = power.mean()

    # Plot the mean as a horizontal line
    plt.axhline(mean_value, color='red', linestyle='--', label='Mean')

    # Additional plot formatting and customization
    plt.legend()
    # Adjust the plot layout
    fig.tight_layout()
    #plt.legend()
    plt.grid(True)

    if title is not None:
        plt.title(title, pad=20)

    if show:
        # Display the plot
        plt.show()
    else:
        # Save the plot to a file
        plt.savefig(file_name.replace('.csv', '.png').replace('Data', 'Plots'),␣
↪dpi=1000, format="png")
        plt.close()
```

[5]:
```
def noise_rms(file_name: str) -> int:
    data = pd.read_table(file_name, sep=';')
    power = data[["Power"]]
    rms = np.sqrt(np.mean(np.square(power.values)))
    return rms * 10**12, power.mean().values[0]/rms
```

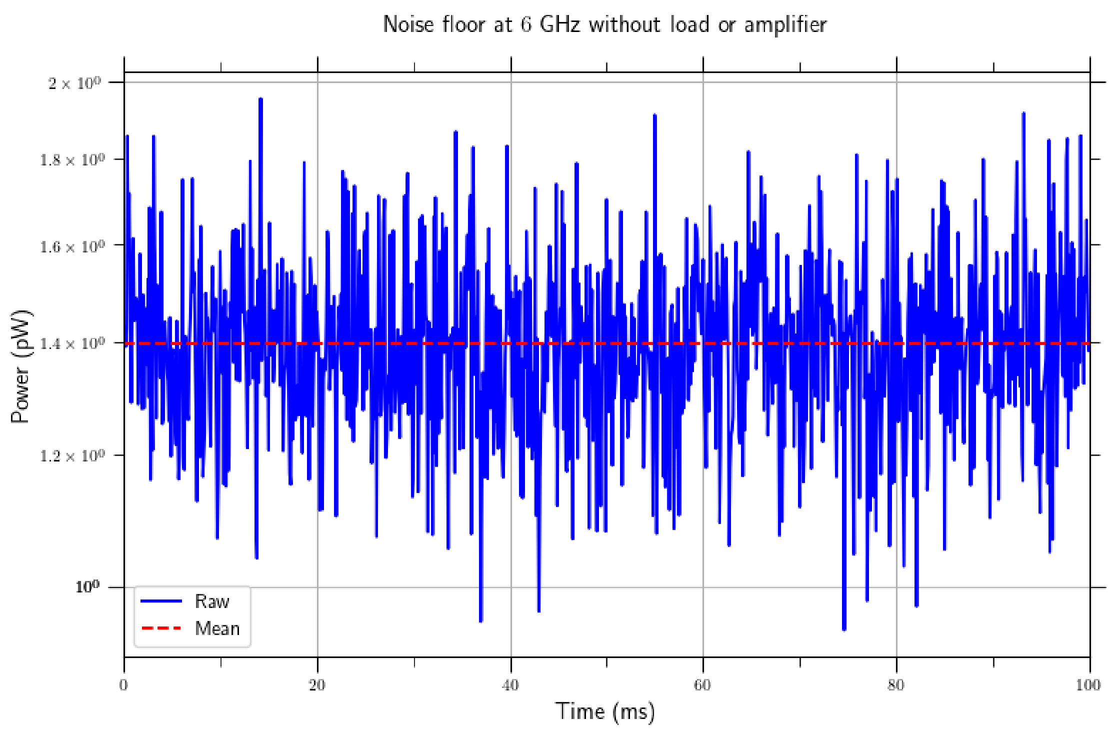
[6]:
```
def noise_ptp(file_name: str) -> int:
    data = pd.read_table(file_name, sep=';')
    power = data[["Power"]]
    mean = power.mean()
    normalised_power = power - mean
    noise_level = np.ptp(normalised_power).values[0]
    return noise_level * 10**12, mean.values[0]/noise_level
```

2 Noise floor measurements at 6 GHz without load or amplifier

---

2.0.1 Raw data with the mean highlighted

[7]:
```
time_plot('Data/noise_floor_6_GHz.DAT', title = r'Noise floor at $6$ GHz without␣
 ↪load or amplifier', show=True)
```



2.0.2 Noise analysis

[9]:
```
file_name = "Data/noise_floor_6_GHz.DAT"
avg_power = pd.read_table(file_name, sep=';')[["Power"]].mean().values[0] *␣
↪10**12
rms_noise, rms_ratio = noise_rms(file_name)
ptp_noise, ptp_ratio = noise_ptp(file_name)

print(f'The average power is {avg_power} pW.')
print(f'The noise level calculated via root mean square is {rms_noise} pW.')
print(f'The noise level calculated via peak-to-peak is {ptp_noise} pW.')
print(f'\nThe signal-to-noise ratio calculated via root mean square is␣
↪{rms_ratio}.')
print(f'The signal-to-noise ratio calculated via peak-to-peak is {ptp_ratio}.')
```

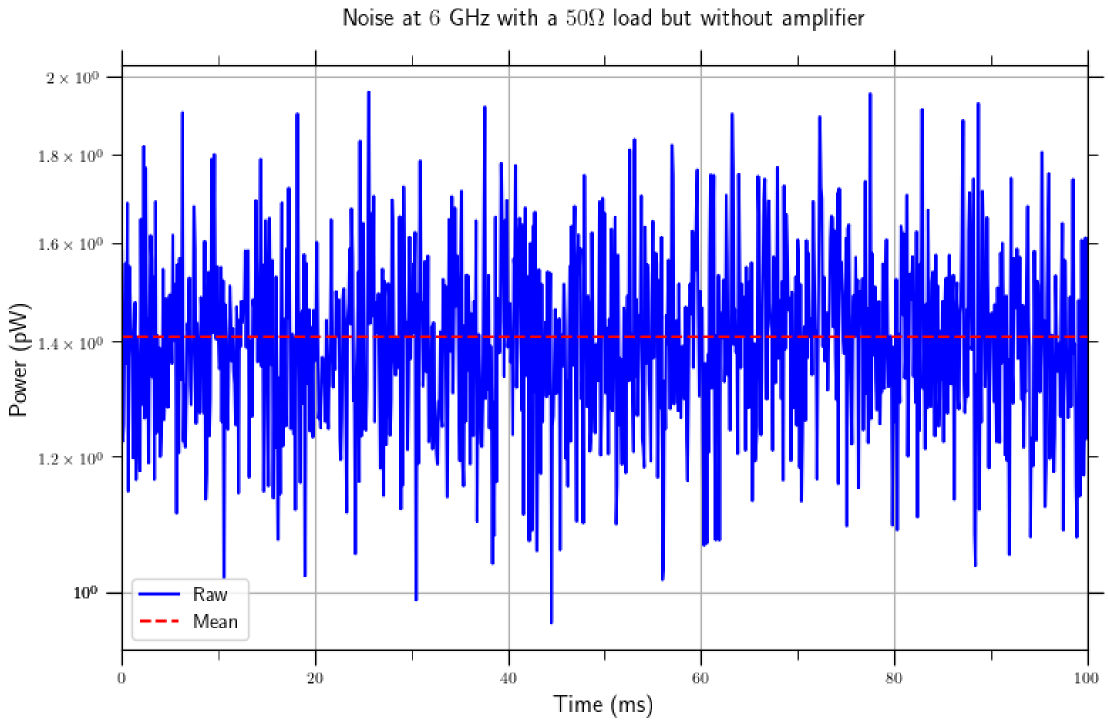
```
The average power is 1.397245621266606 pW.
The noise level calculated via root mean square is 1.407037539079332 pW.
The noise level calculated via peak-to-peak is 1.0109526063506455 pW.

The signal-to-noise ratio calculated via root mean square is 0.9930407558144233.
The signal-to-noise ratio calculated via peak-to-peak is 1.3821079370974745.
```

3 Noise measurements at 6 GHz with a 50Ω load but no amplifier

---

3.0.1 Raw data with the mean highlighted

[10]:
```
time_plot('Data/noise_floor_6_GHz_50ohm.DAT', title = r'Noise at $6$ GHz with a␣
↪$50 \Omega$ load but without amplifier', show=True)
```



3.0.2 Noise analysis

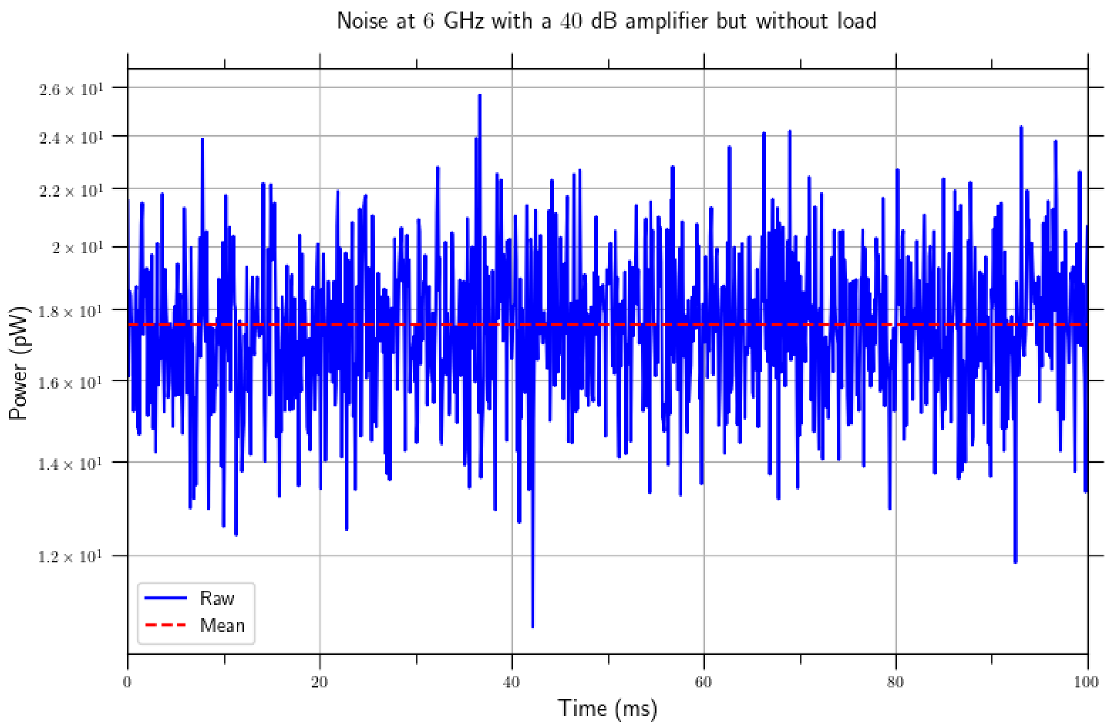
[11]:
```
file_name = "Data/noise_floor_6_GHz_50ohm.DAT"
avg_power = pd.read_table(file_name, sep=';')[["Power"]].mean().values[0] *␣
↪10**12
rms_noise, rms_ratio = noise_rms(file_name)
ptp_noise, ptp_ratio = noise_ptp(file_name)

print(f'The average power is {avg_power} pW.')
print(f'The noise level calculated via root mean square is {rms_noise} pW.')
print(f'The noise level calculated via peak-to-peak is {ptp_noise} pW.')
print(f'\nThe signal-to-noise ratio calculated via root mean square is␣
↪{rms_ratio}.')
print(f'The signal-to-noise ratio calculated via peak-to-peak is {ptp_ratio}.')
```

```
The average power is 1.40951074770017 pW.
The noise level calculated via root mean square is 1.4196296463375668 pW.
The noise level calculated via peak-to-peak is 0.9987133732861092 pW.

The signal-to-noise ratio calculated via root mean square is 0.9928721560137167.
The signal-to-noise ratio calculated via peak-to-peak is 1.411326598203443.
```

4 Noise measurements at 6 GHz with an amplifier but no load

4.0.1 Raw data with the mean highlighted

[12]:
```
time_plot('Data/noise_floor_6_GHz_amplif.DAT', title = r'Noise at $6$ GHz with a␣
↪$40$ dB amplifier but without load', show=True)
```



4.0.2 Noise analysis

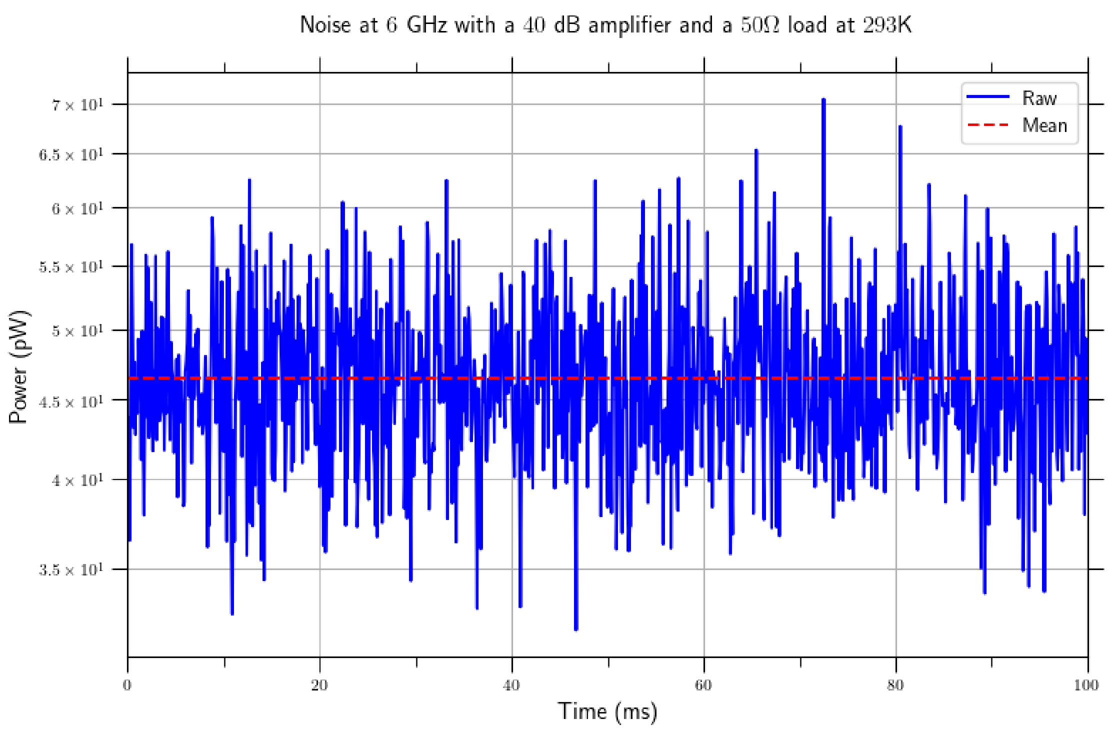
[13]:
```
file_name = "Data/noise_floor_6_GHz_amplif.DAT"
avg_power = pd.read_table(file_name, sep=';')[["Power"]].mean().values[0] *␣
↪10**12
rms_noise, rms_ratio = noise_rms(file_name)
ptp_noise, ptp_ratio = noise_ptp(file_name)

print(f'The average power is {avg_power} pW.')
print(f'The noise level calculated via root mean square is {rms_noise} pW.')
print(f'The noise level calculated via peak-to-peak is {ptp_noise} pW.')
print(f'\nThe signal-to-noise ratio calculated via root mean square is␣
↪{rms_ratio}.')
print(f'The signal-to-noise ratio calculated via peak-to-peak is {ptp_ratio}.')
```

```
The average power is 17.577836150786414 pW.
The noise level calculated via root mean square is 17.706253161393406 pW.
The noise level calculated via peak-to-peak is 15.012096600541813 pW.

The signal-to-noise ratio calculated via root mean square is 0.9927473639150834.
The signal-to-noise ratio calculated via peak-to-peak is 1.1709114734947814.
```

5 Noise measurements at 6 GHz with a 40 dB amplifier and a 50Ω load at 293K

5.0.1 Raw data with the mean highlighted

[14]:
```
time_plot('Data/noise_floor_6_GHz_amplif_50ohm_rt.DAT', title = r'Noise at $6$␣
↪GHz with a $40$ dB amplifier and a $50 \Omega$ load at $293$K', show=True)
```



5.0.2 Noise analysis

[15]:
```
file_name = "Data/noise_floor_6_GHz_amplif_50ohm_rt.DAT"
avg_power = pd.read_table(file_name, sep=';')[["Power"]].mean().values[0] *␣
↪10**12
rms_noise, rms_ratio = noise_rms(file_name)
```

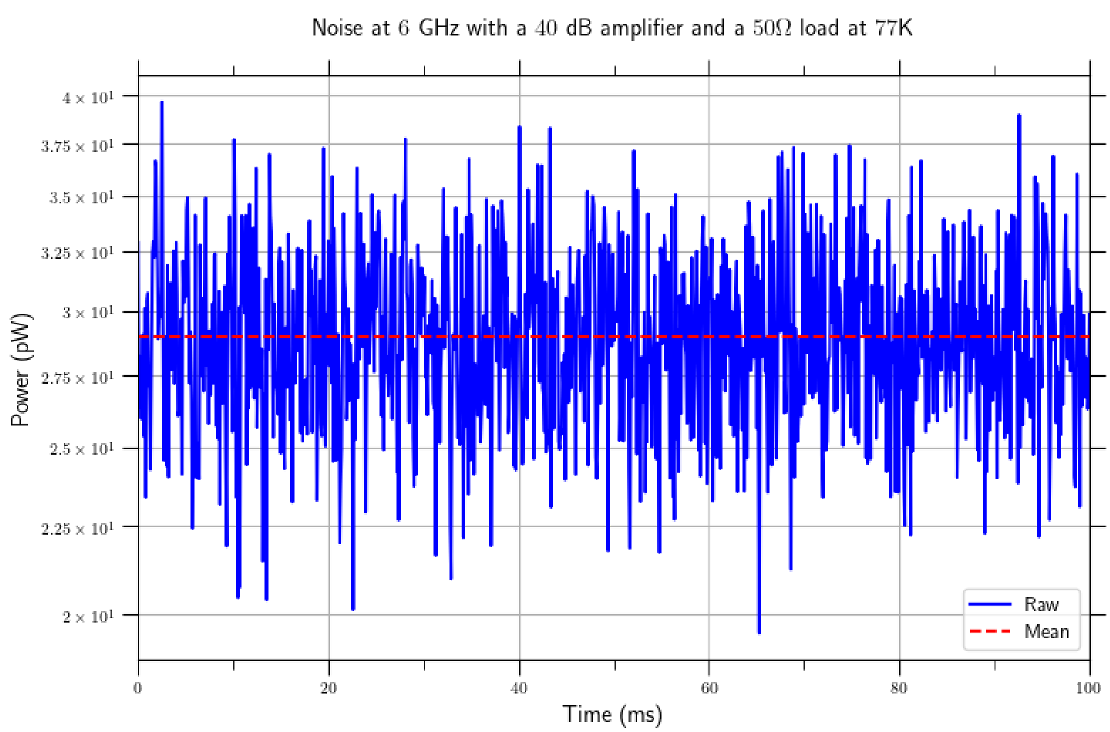
```
ptp_noise, ptp_ratio = noise_ptp(file_name)

print(f'The average power is {avg_power} pW.')
print(f'The noise level calculated via root mean square is {rms_noise} pW.')
print(f'The noise level calculated via peak-to-peak is {ptp_noise} pW.')
print(f'\nThe signal-to-noise ratio calculated via root mean square is␣
↪{rms_ratio}.')
print(f'The signal-to-noise ratio calculated via peak-to-peak is {ptp_ratio}.')
```

```
The average power is 46.4540239788855 pW.
The noise level calculated via root mean square is 46.79478776896194 pW.
The noise level calculated via peak-to-peak is 38.51223853712504 pW.

The signal-to-noise ratio calculated via root mean square is 0.9927179114101579.
The signal-to-noise ratio calculated via peak-to-peak is 1.2062145890092764.
```

6 Noise measurements at 6 GHz with a 40 dB amplifier and a 50Ω load at 77K

6.0.1 Raw data with the mean highlighted

[16]:
```
time_plot('Data/noise_floor_6_GHz_amplif_50ohm_lt.DAT', title = r'Noise at $6$␣
↪GHz with a $40$ dB amplifier and a $50 \Omega$ load at $77$K', show=True)
```



6.0.2 Noise analysis

[17]:
```
file_name = "Data/noise_floor_6_GHz_amplif_50ohm_lt.DAT"
avg_power = pd.read_table(file_name, sep=';')[["Power"]].mean().values[0] *␣
↪10**12
rms_noise, rms_ratio = noise_rms(file_name)
ptp_noise, ptp_ratio = noise_ptp(file_name)

print(f'The average power is {avg_power} pW.')
print(f'The noise level calculated via root mean square is {rms_noise} pW.')
print(f'The noise level calculated via peak-to-peak is {ptp_noise} pW.')
print(f'\nThe signal-to-noise ratio calculated via root mean square is␣
↪{rms_ratio}.')
print(f'The signal-to-noise ratio calculated via peak-to-peak is {ptp_ratio}.')
```

```
The average power is 28.985810319627877 pW.
The noise level calculated via root mean square is 29.183656631489008 pW.
The noise level calculated via peak-to-peak is 20.15551614498179 pW.

The signal-to-noise ratio calculated via root mean square is 0.9932206469408753.
The signal-to-noise ratio calculated via peak-to-peak is 1.4381080648656377.
```