In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import csv
import os
import scipy.optimize as opt
%matplotlib widget
```

In [2]:
```
#pull data from FieldFox csv files
def get_data(file):
    with open(file, 'r') as file:
        reader = csv.reader(file, delimiter=',')
        freqs = []
        data = []
        for row in reader:
            try:
                freq = int(row[0])
                dat = float(row[1])
                freqs.append(freq)
                data.append(dat)
            except ValueError:
                continue
        return np.array(freqs), np.array(data)
```

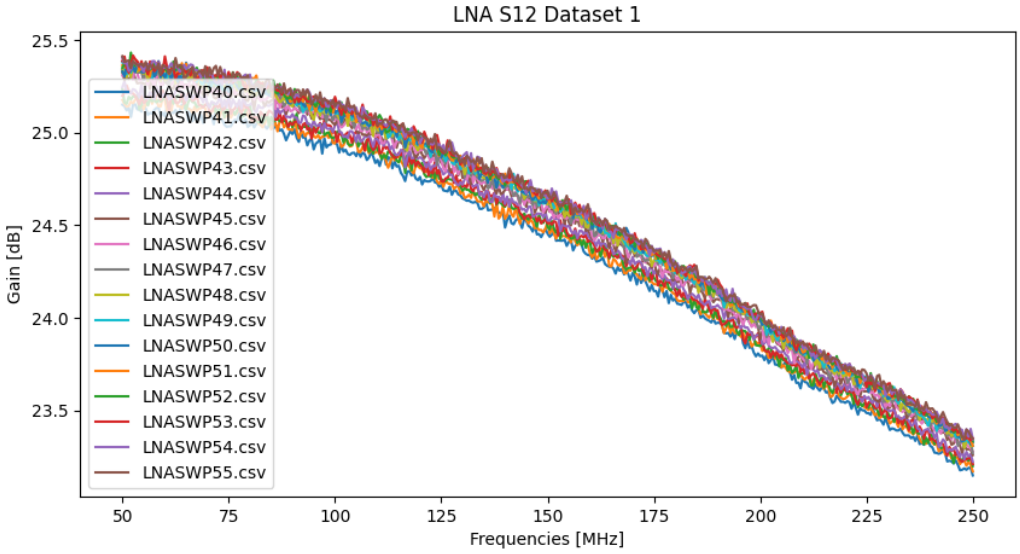
LNA data

I present two sets of S12 data for the LNA, and take the average of the difference across the frequency axis. This shows some dependency of the LNA gain on the voltage provided.

The fitted lines of the two datasets are inconsistent, which is likely a result of the temperature of the LNA. The front-end RF chain should thus be characterized in future as a function of voltage at the temperature it will be operating at.

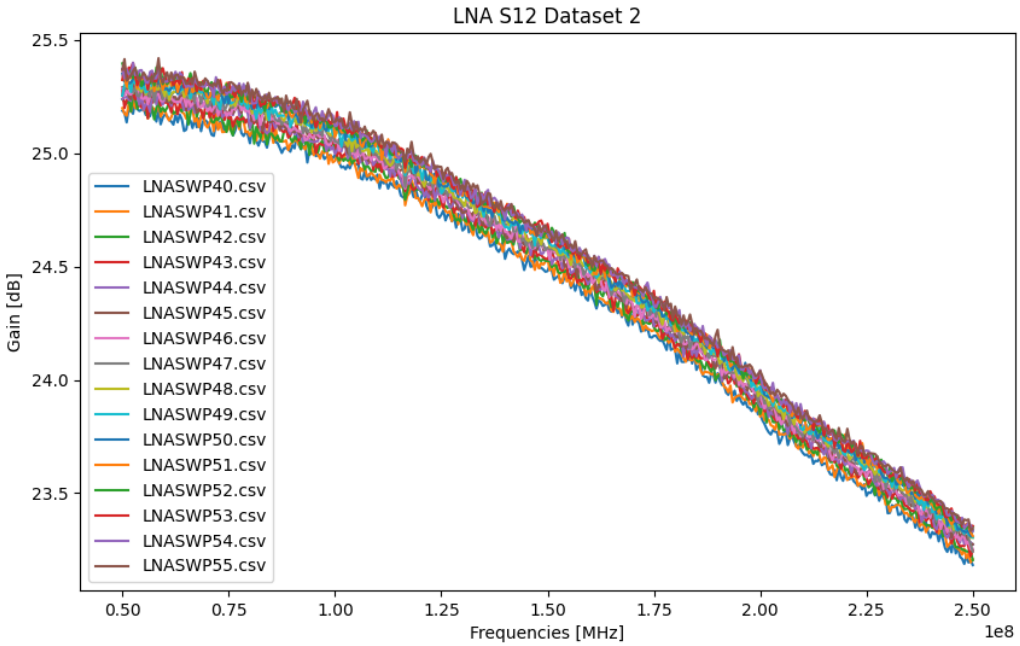
In [3]:
```
#dataset 1
data_array = []
plt.figure(figsize=(10,5))
for file in sorted(os.listdir('LNA_gain/LNAgain_618')):
    if 'SWP' in file and 'csv' in file:
        freqs, data = get_data(os.path.join('LNA_gain/LNAgain_618', file))
        data_array.append(data)
        plt.plot(freqs/1e6, data, label=file)
data_array = np.array(data_array)
plt.title('LNA S12 Dataset 1')
plt.xlabel('Frequencies [MHz]')
plt.ylabel('Gain [dB]')
plt.legend()
plt.show()
```

Figure



In [4]:
```
#dataset 2
data_array2 = []
plt.figure(figsize=(10,6))
for file in sorted(os.listdir('LNA_gain/LNAgain2_618')):
    if 'SWP' in file and 'csv' in file:
        freqs, data = get_data(os.path.join('LNA_gain/LNAgain2_618', file))
        data_array2.append(data)
        plt.plot(freqs, data, label=file)
data_array2 = np.array(data_array2)
plt.legend()
plt.title('LNA S12 Dataset 2')
plt.xlabel('Frequencies [MHz]')
plt.ylabel('Gain [dB]')
plt.show()
```

Figure

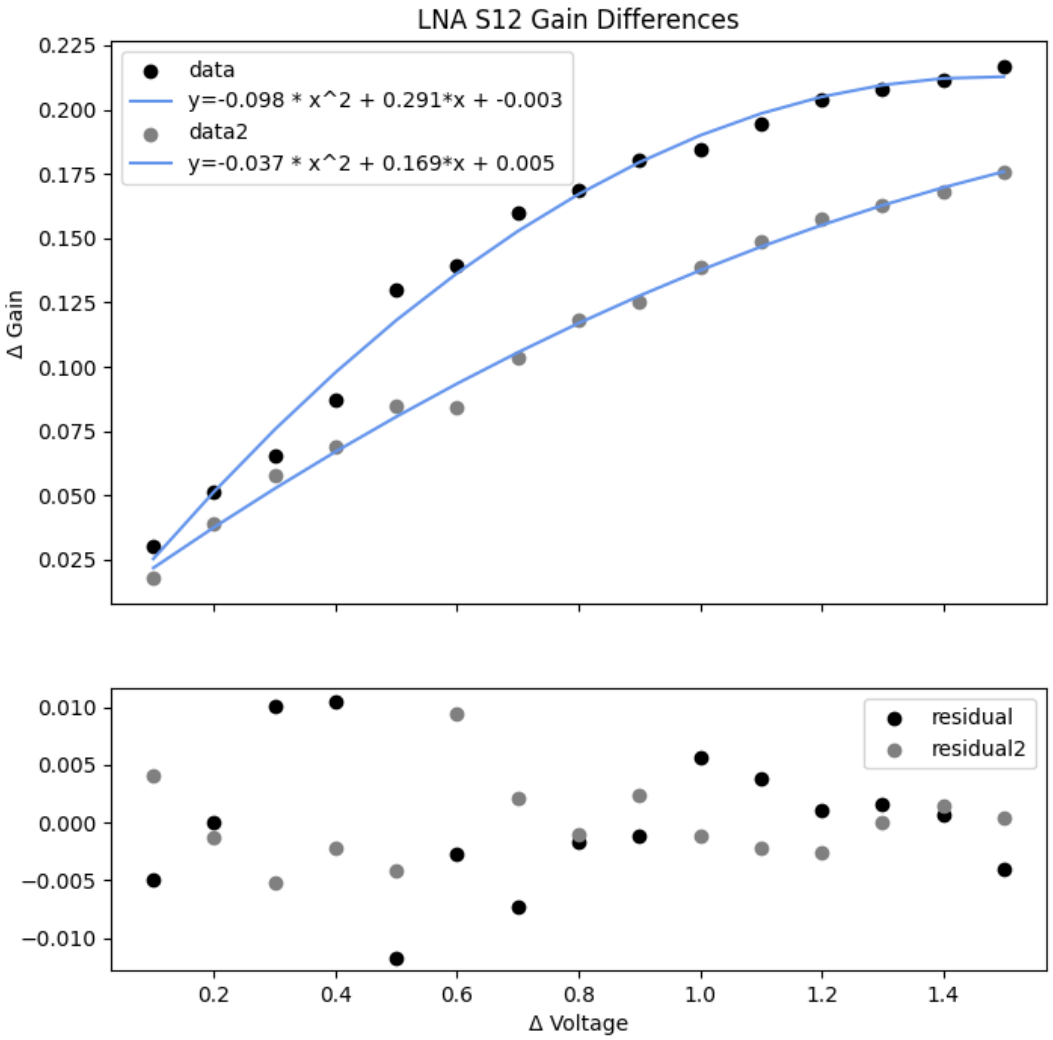


In [5]:
```
#finding the mean of the differences
voltages = np.array([4.0, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5.0, 5.1, 5.2,
voltage_diff = voltages[1:] - voltages[0]

gain_diff = data_array[1:] - data_array[0]
gain_diff_average = np.mean(gain_diff, axis=1)

gain_diff2 = data_array2[1:] - data_array2[0]
gain_diff_average2 = np.mean(gain_diff2, axis=1)
```

In [6]:
```
#curve fit with a second degree polynomial
def model(x, a, b, c):
    return a * x**2 + b * x + c

popt, pcov = opt.curve_fit(f=model, xdata=voltage_diff, ydata=gain_diff_average)
popt2, pcov2 = opt.curve_fit(f=model, xdata=voltage_diff, ydata=gain_diff_average2)
```

In [7]:
```
#plotting curve fit and residuals
m = model(voltage_diff, popt[0], popt[1], popt[2])
m2 = model(voltage_diff, popt2[0], popt2[1], popt2[2])

fig,ax = plt.subplots(2,1,sharex=True, height_ratios=(2,1), figsize=(8,8))
ax[0].scatter(voltage_diff, gain_diff_average, color='black', label='data')
ax[0].plot(voltage_diff, m, color='cornflowerblue', label=f'y={popt[0]:.3f} * x^2 + {p
ax[0].set_ylabel('Δ Gain')

ax[0].scatter(voltage_diff, gain_diff_average2, color='gray', label='data2')
ax[0].plot(voltage_diff, m2, color='cornflowerblue', label=f'y={popt2[0]:.3f} * x^2 +
ax[0].legend()
ax[0].set_title('LNA S12 Gain Differences')
ax[1].scatter(voltage_diff, m-gain_diff_average, color='black', label='residual')
ax[1].scatter(voltage_diff, m2-gain_diff_average2, color='gray', label='residual2')
ax[1].legend()
ax[1].set_xlabel('Δ Voltage')
plt.show()
```

Figure



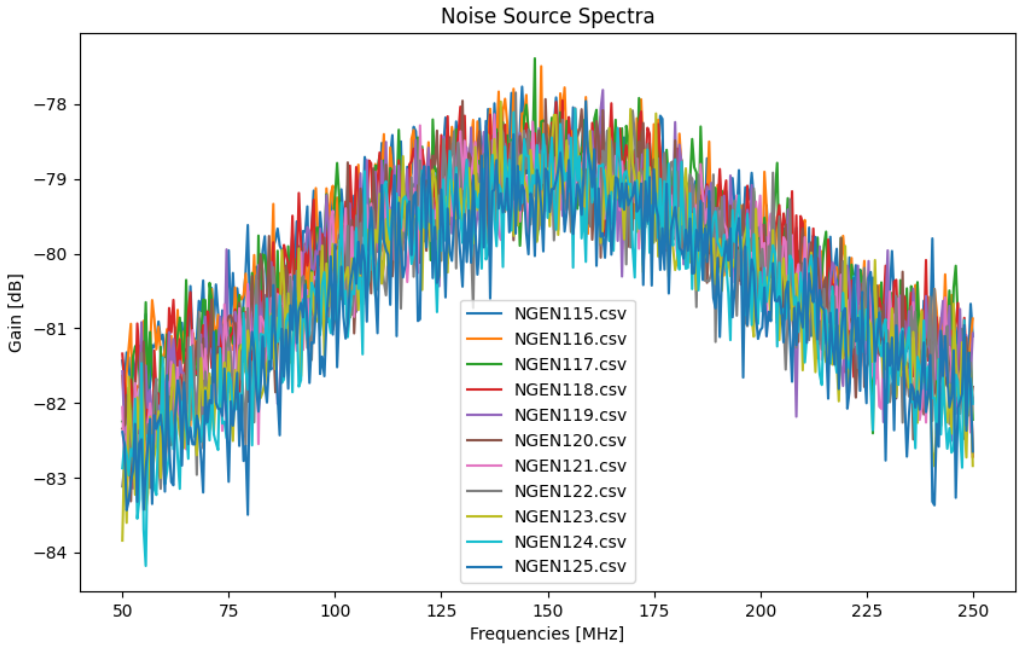
Noise Source

Following the same method, I show that there is a voltage dependence for the noise source. As we have control over the PCB that the noise source resides on, we should add a zener diode to stabilize the voltage. With the Peltier device, it should also be thermally stable.

In [8]:
```
noise_array = []
plt.figure(figsize=(10,6))
for file in sorted(os.listdir('noise_source2')):
    if 'csv' in file and 'OFF' not in file:
        freqs, noise = get_data(os.path.join('noise_source2', file))
        noise_array.append(noise)
        plt.plot(freqs/1e6, noise, label=file)
noise_array = np.array(noise_array)
plt.title('Noise Source Spectra')
plt.legend()
plt.xlabel('Frequencies [MHz]')
```

```
plt.ylabel('Gain [dB]')
plt.show()
```

Figure

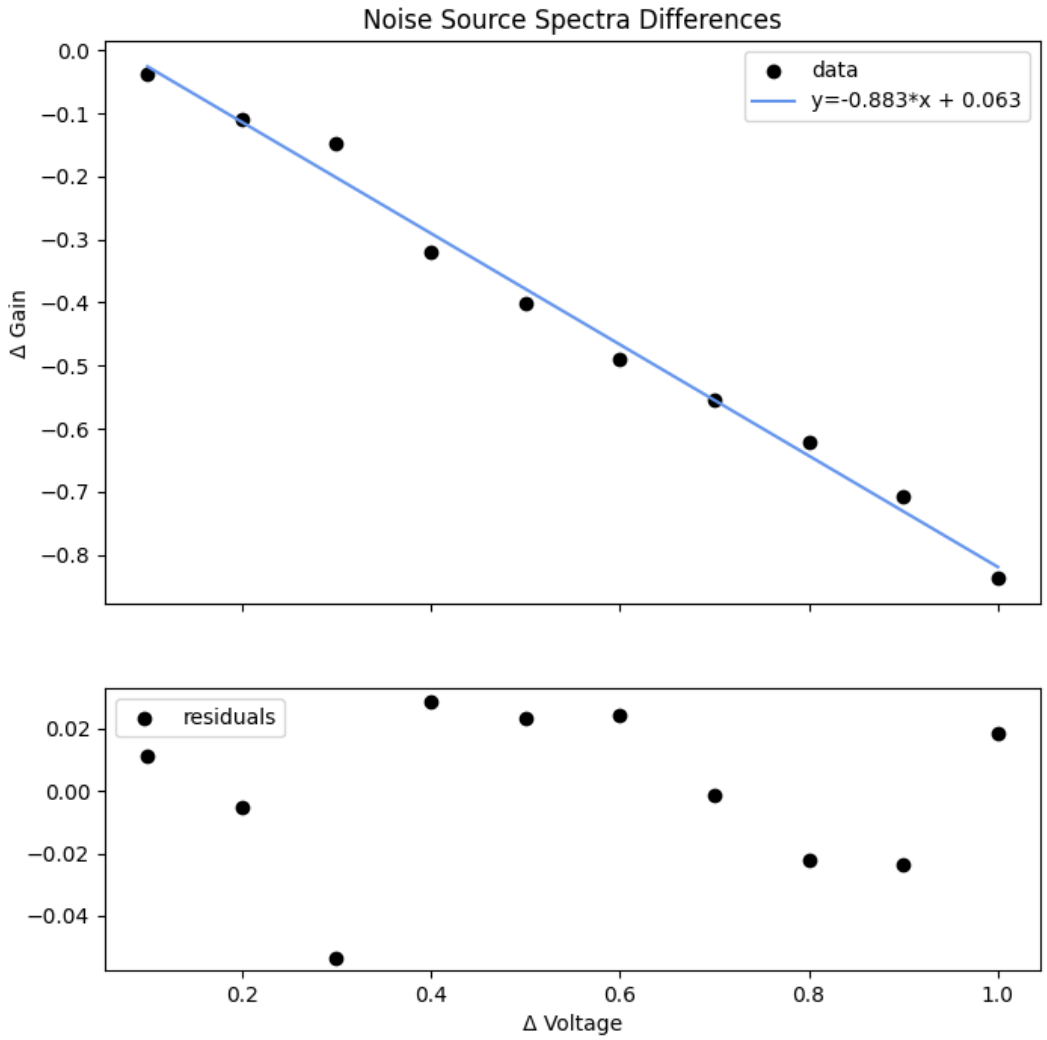


In [9]:
```
spec_diff = noise_array[1:] - noise_array[0]
spec_diff_average = np.mean(spec_diff, axis=1)
spec_diff_std = np.std(spec_diff, axis=1)

voltages = np.array([11.5, 11.6, 11.7, 11.8, 11.9, 12.0, 12.1, 12.2, 12.3, 12.4, 12.5])
voltage_diff = voltages[1:] - voltages[0]
```

In [10]:
```
#curve fit with a second degree polynomial
def line(x, a, b):
    return a * x + b

popt, pcov = opt.curve_fit(xdata = voltage_diff, ydata = spec_diff_average, f=line)
```

In [11]:
```
m = line(voltage_diff, popt[0], popt[1])
fig,ax = plt.subplots(2,1,sharex=True, height_ratios=(2,1), figsize=(8,8))
ax[0].scatter(voltage_diff, spec_diff_average, color='black', label='data')
ax[0].plot(voltage_diff, m, color='cornflowerblue', label=f'y={popt[0]:.3f}*x + {popt[
ax[0].legend()
ax[0].set_ylabel('Δ Gain')
ax[0].set_title('Noise Source Spectra Differences')
ax[1].scatter(voltage_diff, m-spec_diff_average, color='black', label='residuals')
ax[1].legend()
ax[1].set_xlabel('Δ Voltage')
plt.show()
```

Figure


Noise Source Spectra Differences
data
y=-0.883*x + 0.063
Δ Gain
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
−0.8
residuals
0.02
0.00
−0.02
−0.04
0.2
0.4
0.6
0.8
1.0
Δ Voltage

In [ ]: