```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```python
# Read the CSV file
#df = pd.read_csv('/content/adv lab 1 - Sheet1.csv')
df = pd.read_csv('/content/adv lab 1 - 10_22 data.csv')

# Convert the first three columns to NumPy arrays
# Adjust column indices if you need different columns
current = df.iloc[:, 0].values
f_lo = df.iloc[:, 1].values
f_hi = df.iloc[:, 2].values

print(current)
print(f_lo)
print(f_hi)
```

```
[-2.  -1.8 -1.6 -1.4 -1.2 -1.  -0.8 -0.6 -0.4 -0.2  0.   0.2  0.4  0.6
  0.8  1.   1.2  1.4  1.6  1.8  2. ]
[152.5879 153.3203 153.8086 154.2969 154.7852 155.0293 155.5176 155.5176
 155.5176 155.5176 155.2734 155.0293 154.541  154.0527 153.3203 152.832
 152.0996 151.3672 150.6348 149.9023 149.1699]
[170.4102 169.6777 169.1895 168.457  167.9688 167.4805 167.4805 166.9922
 166.9922 166.9922 167.2363 167.4805 167.9688 168.457  168.9453 169.4336
 170.166  170.6543 171.3867 172.1191 172.6074]
```

```python
# Calculate uncertainty due to resolution
resolution_uncertainty = 97.5 / 400

# Calculate uncertainty due to accuracy (25 ppm of the frequency value)
accuracy_f_lo_unc = f_lo * (25e-6)
accuracy_f_hi_unc = f_hi * (25e-6)

# Combine uncertainties
f_lo_unc = np.sqrt(resolution_uncertainty**2 + accuracy_f_lo_unc**2)
f_hi_unc = np.sqrt(resolution_uncertainty**2 + accuracy_f_hi_unc**2)


print("Uncertainty for f_lo:", f_lo_unc)
print("Uncertainty for f_hi:", f_hi_unc)
```

```
Uncertainty for f_lo: [0.24377985 0.24378014 0.24378033 0.24378052 0.24378071 0.24378081
 0.24378101 0.24378101 0.24378101 0.24378101 0.24378091 0.24378081
 0.24378062 0.24378042 0.24378014 0.24377994 0.24377966 0.24377937
 0.24377909 0.24377881 0.24377853]
Uncertainty for f_hi: [0.24378723 0.24378691 0.2437867  0.24378638 0.24378617 0.24378596
 0.24378596 0.24378575 0.24378575 0.24378575 0.24378585 0.24378596
 0.24378617 0.24378638 0.24378659 0.2437868  0.24378712 0.24378733
 0.24378766 0.24378798 0.24378819]
```

$$\omega_- = \sqrt{\frac{\kappa}{I}} \implies \kappa = I\omega_-^2$$

$$\omega_+ = \sqrt{\frac{\kappa+2\lambda}{I}} \implies \lambda = \frac{I\omega_+^2-\kappa}{2}$$

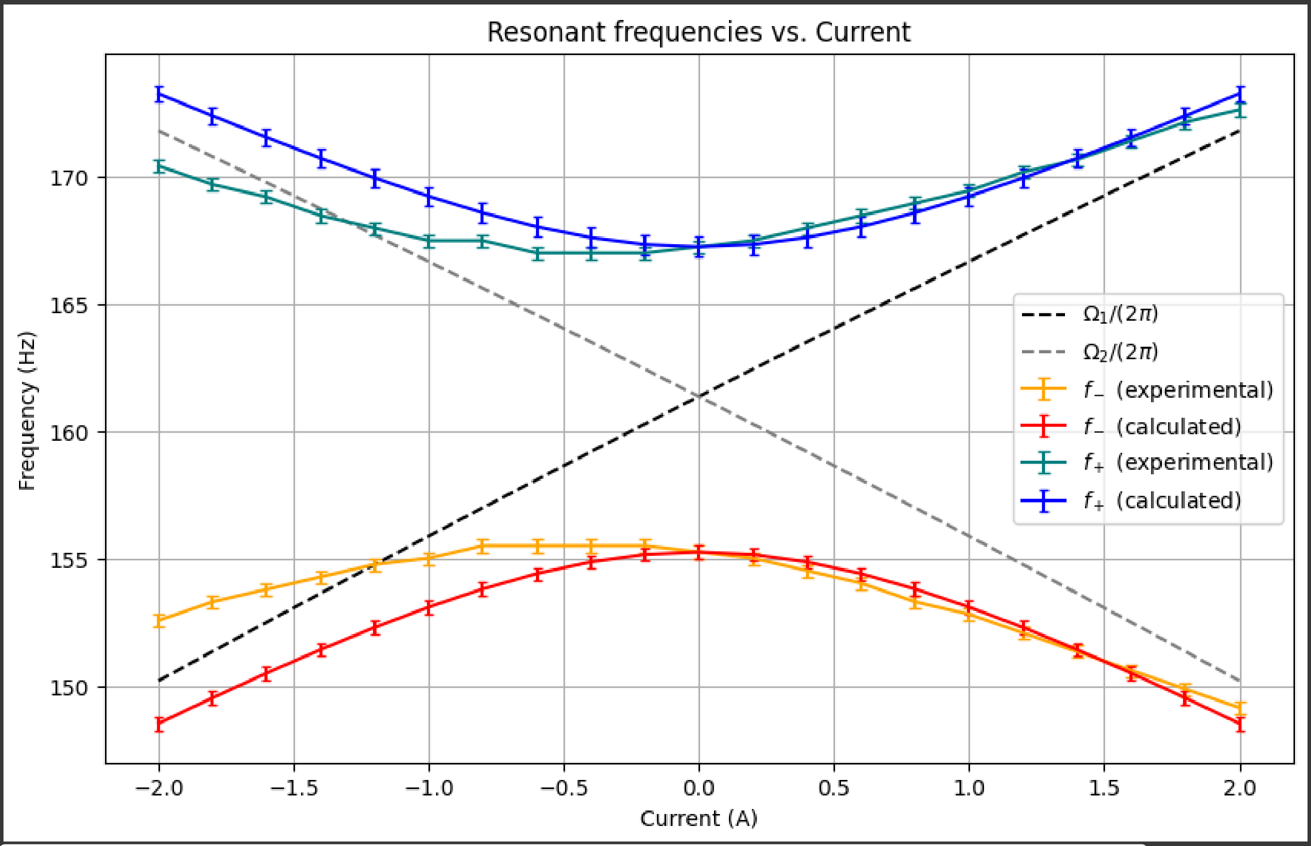
```python
I = 1.29e-8

kappa = I*(2*np.pi*f_lo[10])**2
kappa_unc = 8 * np.pi**2 * I * f_lo[10] * f_lo_unc[10]

lambda_val = (I*(2*np.pi*f_hi[10])**2 - kappa) / 2
lambda_unc = np.sqrt( (kappa_unc**2)/4 + (I*(2*np.pi*f_hi[10])*(2*np.pi*f_hi_unc[10]))**2 )

print("kappa:", kappa)
print("kappa_unc:", kappa_unc)
print("lambda_val:", lambda_val)
print("lambda_unc:", lambda_unc)

#kappa = 0.01220280
```

```python
#kappa_unc = 2.5e-8

#lambda_val = 0.001021
#lambda_unc = 0.000019

mu = 0.26
B_T = 3.4e-3*current

f_hi_calc = np.sqrt((kappa + lambda_val + np.sqrt(lambda_val**2 + (mu * B_T)**2)) / I) / (2*np.pi)
f_hi_calc_unc = np.sqrt(((lambda_unc * lambda_val / np.sqrt(B_T**2 * mu**2 + lambda_val**2) + lambda_unc)**2 +

f_lo_calc = np.sqrt((kappa + lambda_val - np.sqrt(lambda_val**2 + (mu * B_T)**2)) / I) / (2*np.pi)
f_lo_calc_unc = np.sqrt((kappa_unc**2 * (lambda_val**2 + B_T**2 * mu**2) + lambda_unc**2 * (2 * lambda_val**2 +

print("f_hi_calc:", f_hi_calc)
print("f_lo_calc:", f_lo_calc)
print("f_hi_calc_unc:", f_hi_calc_unc)
print("f_lo_calc_unc:", f_lo_calc_unc)
```

```
kappa: 0.012278450750887734
kappa_unc: 3.855459948494936e-05
lambda_val: 0.0009824234153908993
lambda_unc: 2.8332194887643867e-05
f_hi_calc: [173.2353678  172.36851929 171.52385231 170.70919486 169.93510249
 169.21575695 168.56989739 168.02133834 167.59811689 167.32892563
 167.2363     167.32892563 167.59811689 168.02133834 168.56989739
 169.21575695 169.93510249 170.70919486 171.52385231 172.36851929
 173.2353678 ]
f_lo_calc: [148.55071904 149.55568309 150.52367546 151.44695301 152.31503448
 153.11380207 153.82457046 154.42356896 154.88279439 155.1735784
 155.2734     155.1735784  154.88279439 154.42356896 153.82457046
 153.11380207 152.31503448 151.44695301 150.52367546 149.55568309
 148.55071904]
f_hi_calc_unc: [0.32350564 0.32987676 0.33699752 0.34496179 0.35382637 0.36354935
 0.37388154 0.38421033 0.3934215  0.39996686 0.40235984 0.39996686
 0.3934215  0.38421033 0.37388154 0.36354935 0.35382637 0.34496179
 0.33699752 0.32987676 0.32350564]
f_lo_calc_unc: [0.27240318 0.26805575 0.26370297 0.25941676 0.25531616 0.25157735
 0.2484244  0.24607457 0.24462601 0.24395419 0.24378091 0.24395419
 0.24462601 0.24607457 0.2484244  0.25157735 0.25531616 0.25941676
 0.26370297 0.26805575 0.27240318]
```

```python
Omega1 = np.sqrt((kappa + mu*B_T + lambda_val)/I)
Omega2 = np.sqrt((kappa - mu*B_T + lambda_val)/I)

```

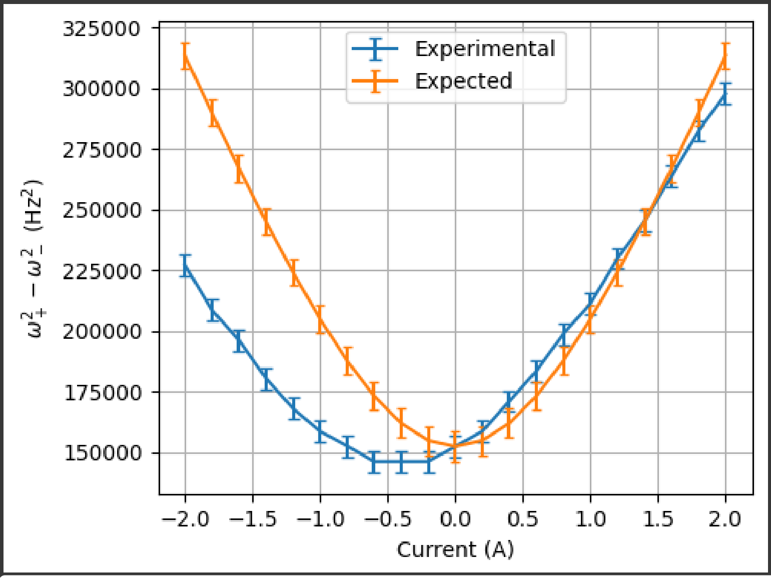
```python
plt.rcParams['figure.figsize'] = [10, 6]

plt.plot(current, Omega1/(2*np.pi),"--", label=r'$\Omega_1/(2\pi)$', color="0")
plt.plot(current, Omega2/(2*np.pi),"--", label=r'$\Omega_2/(2\pi)$', color="0.5")

plt.errorbar(current, f_lo, yerr=f_lo_unc, capsize=3, label='$f_-$ (experimental)', color="orange")
plt.errorbar(current, f_lo_calc, yerr=f_lo_calc_unc, capsize=2, label='$f_-$ (calculated)', color="red")
#plt.plot(current, f_lo_calc, label='$f_-$ (calculated)', color="red")


plt.errorbar(current, f_hi, yerr=f_hi_unc, capsize=3, label='$f_+$ (experimental)', color="teal")
plt.errorbar(current, f_hi_calc, yerr=f_hi_calc_unc, capsize=2, label='$f_+$ (calculated)', color="blue")



plt.xlabel('Current (A)')
plt.ylabel('Frequency (Hz)')
plt.title('Resonant frequencies vs. Current')
plt.legend()
plt.grid(True)
plt.savefig("avoided_crossing.svg", format="svg")
```



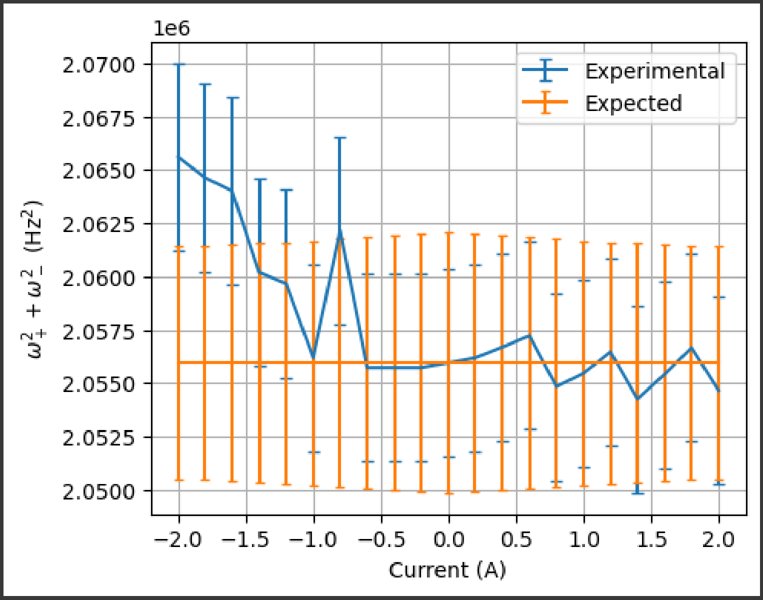
```python
omega_hi_squared = (2 * np.pi * f_hi)**2
omega_hi_squared_unc = 8 * np.pi**2 * f_hi * f_hi_unc

omega_lo_squared = (2 * np.pi * f_lo)**2
omega_lo_squared_unc = 8 * np.pi**2 * f_lo * f_lo_unc

omega_squared_diff = omega_hi_squared - omega_lo_squared
omega_squared_diff_unc = np.sqrt(omega_hi_squared_unc**2 + omega_lo_squared_unc**2)

omega_squared_sum = omega_hi_squared + omega_lo_squared
omega_squared_sum_unc = np.sqrt(omega_hi_squared_unc**2 + omega_lo_squared_unc**2)
```

```python
omega_hi_squared_calc = (2 * np.pi * f_hi_calc)**2
omega_hi_squared_calc_unc = 8 * np.pi**2 * f_hi_calc * f_hi_calc_unc

omega_lo_squared_calc = (2 * np.pi * f_lo_calc)**2
omega_lo_squared_calc_unc = 8 * np.pi**2 * f_lo_calc * f_lo_calc_unc

omega_squared_diff_calc = omega_hi_squared_calc - omega_lo_squared_calc
omega_squared_diff_calc_unc = np.sqrt(omega_hi_squared_calc_unc**2 + omega_lo_squared_calc_unc**2)

omega_squared_sum_calc = omega_hi_squared_calc + omega_lo_squared_calc
omega_squared_sum_calc_unc = np.sqrt(omega_hi_squared_calc_unc**2 + omega_lo_squared_calc_unc**2)
```

```python
plt.rcParams['figure.figsize'] = [5, 4]


# Plot the experimental values with error bars
plt.errorbar(current, omega_squared_diff, yerr=omega_squared_diff_unc, capsize=3, label='Experimental')

# Plot the expected values
plt.errorbar(current, omega_squared_diff_calc, yerr=omega_squared_diff_calc_unc, capsize=2, label='Expected')

# Label the axes and add a title
plt.xlabel('Current (A)')
plt.ylabel(r'$\omega_+^2 - \omega_-^2$ (Hz$^2$)')
#plt.title("Difference of squared frequencies")

# Add a legend and grid
plt.legend()
plt.grid(True)

# Display the plot
plt.savefig("omegadiff.svg", format="svg")
```



```
# Plot the experimental values with error bars
plt.errorbar(current, omega_squared_sum, yerr=omega_squared_sum_unc, capsize=3, label='Experimental')

# Plot the expected values
plt.errorbar(current, omega_squared_sum_calc, yerr=omega_squared_sum_calc_unc, capsize=2, label='Expected')

# Label the axes and add a title
plt.xlabel('Current (A)')
plt.ylabel(r'$\omega_+^2 + \omega_-^2$ (Hz$^2$)')
#plt.title("Sum of squared frequencies")

# Add a legend and grid
plt.legend()
plt.grid(True)

# Display the plot
plt.savefig("omegasum.svg", format="svg")
```


1e6
Experimental
Expected
$\omega_+^2 + \omega_-^2$ (Hz$^2$)
Current (A)