modex_graphcode

November 9, 2025

[71]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
from scipy.optimize import curve_fit

import scienceplots
```

[72]:
```
plt.style.use('science')

plt.rcParams['figure.figsize'] = [12, 8]
plt.rcParams['text.usetex'] = False

plt.rcParams['axes.prop_cycle'] = plt.cycler(color=[
'#0C5DA5', '#00B945', '#FF9500', '#FF2C00', '#845B97', '#474747', '#9e9e9e'
])
```

[73]:
```
T = np.array([60, 90, 120, 150, 180])
```

1 Modulus of Rigidity

[74]:
```
# eq.5
def mod_rigid(I, l, omega0, r):
    return (2 * I * l * omega0**2)/(np.pi * r**4)

# I = moment of inertia
# l = length
# omega0 = angular frequency
# r = radius
```

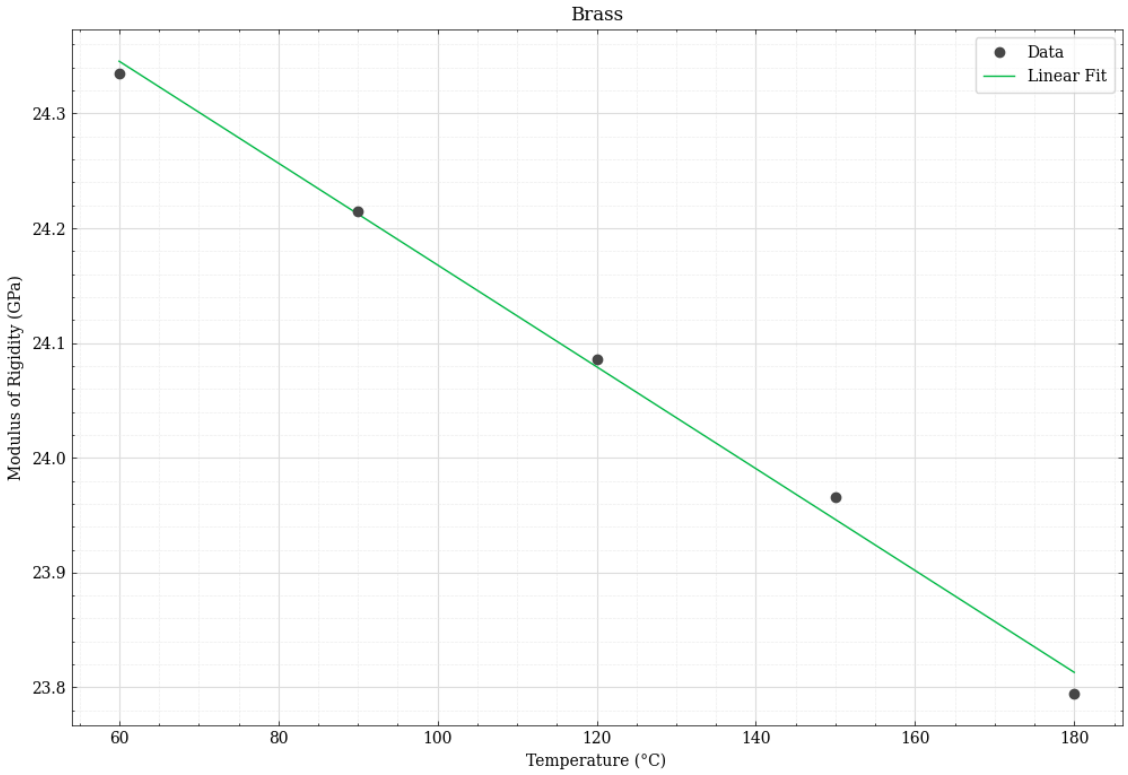
[75]:
```
mag_m = 54.54e-3
mag_r = 9.35e-3 / 2
```

[133]:
```
# brass - T= 60, 90, 120, 150, 180

m =  23.55e-3
l =  35.5e-2
r = 3.20e-3 / 2
```

```
I = 0.5 * m * r**2 + 0.5 * mag_m * mag_r**2
fres = np.array([195.558052, 195.075226, 194.552084, 194.069886, 193.373124])
omega0 = 2 * np.pi * fres

BG = mod_rigid(I, l, omega0, r)

print(BG * 1e-9)

print(np.sum(fres)/len(fres))
```

```
[32.5998737  32.43909653 32.26534305 32.10560174 31.87548045]
194.5256744
```

[77]:
```
Bval = np.polyfit(T, BG, 1)
Bfit = np.poly1d(Bval) * 1e-9

plt.plot(T, BG * 1e-9, 'o', label='Data', zorder=3, color='#474747')
plt.plot(T, Bfit(T), label='Linear Fit', color='#00B945')

plt.xlabel('Temperature (°C)')
plt.ylabel('Modulus of Rigidity (GPa)')
plt.title('Brass')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```

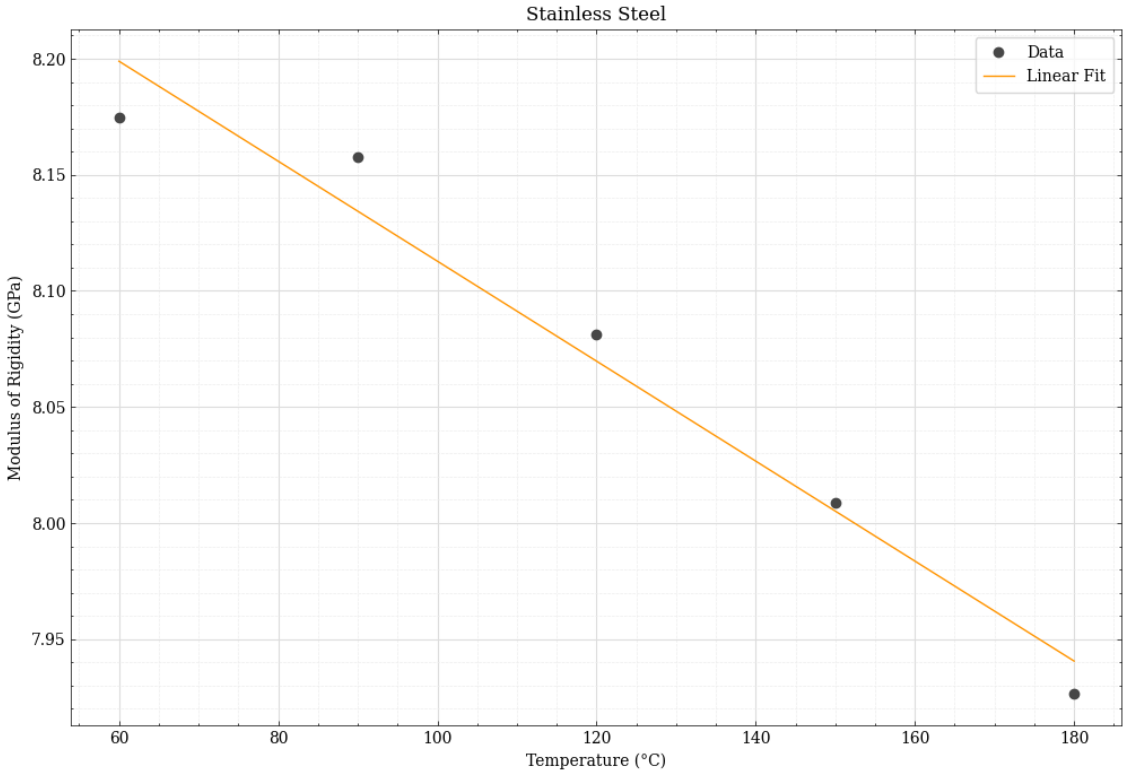


[ ]:
```
m =  22.23e-3
l =  35e-2
r = 3.10e-3 / 2
I = 0.5 * m * r**2 + 0.5 * mag_m * mag_r**2
fres = np.array([92.813146, 92.714482, 92.279800, 91.864033, 91.391520])
omega0 = 2 * np.pi * fres

SSG = mod_rigid(I, l, omega0, r)

print(SSG)

print(np.sum(fres)/len(fres))
```

```
[8.17492213e+09 8.15755084e+09 8.08123853e+09 8.00858248e+09
 7.92640826e+09]
92.2125962
```

[79]:
```
SSval = np.polyfit(T, SSG, 1)
SSfit = np.poly1d(SSval) * 1e-9

plt.plot(T, SSG * 1e-9, 'o', label='Data', zorder=3, color='#474747')
plt.plot(T, SSfit(T), label='Linear Fit', color='#FF9500')
```

```
plt.xlabel('Temperature (°C)')
plt.ylabel('Modulus of Rigidity (GPa)')
plt.title('Stainless Steel')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```

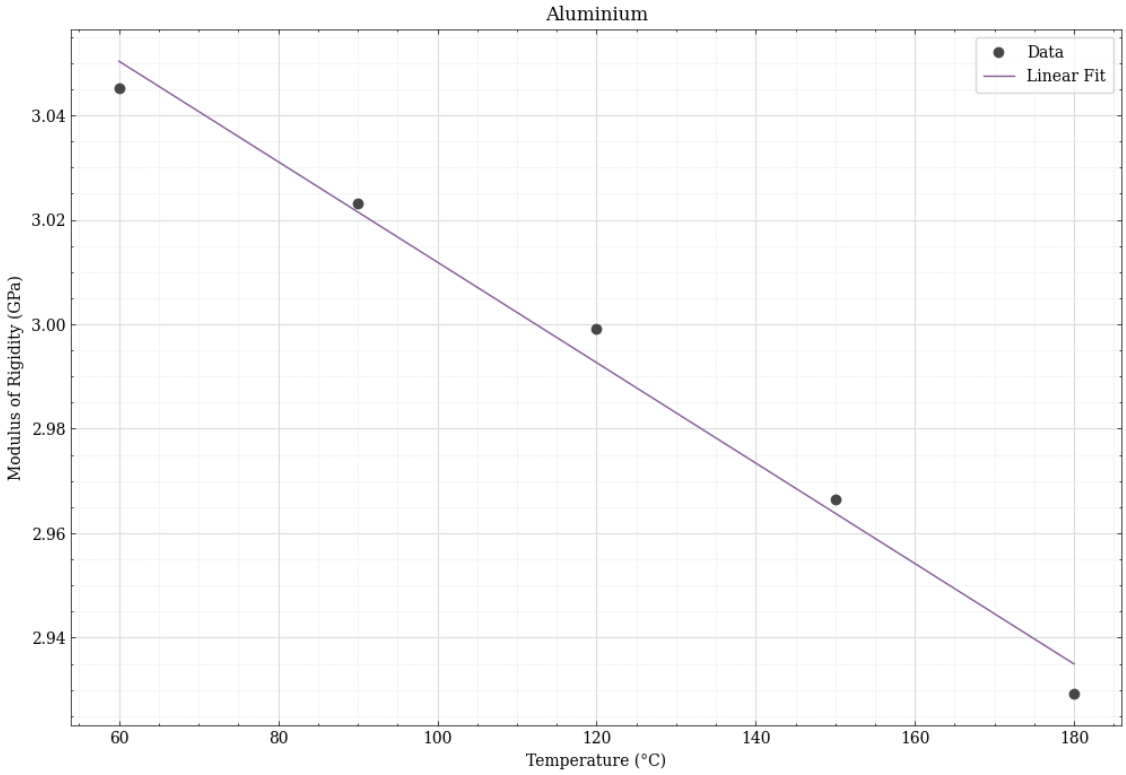


[132]:
```
# aluminium - l.grey - T= 60, 90, 120, 150, 180

m = 7.71e-3
l = 35.6e-3
r = 3.10e-3 / 2
I = 0.5 * m * r**2 + 0.5 * mag_m * mag_r**2
fres = np.array([180.156115, 179.506483, 178.792279, 177.814923, 176.694473])
omega0 = 2 * np.pi * fres
```

```
AG = mod_rigid(I, l, omega0, r)

print(AG * 1e-9)

print(np.sum(fres)/len(fres))
```

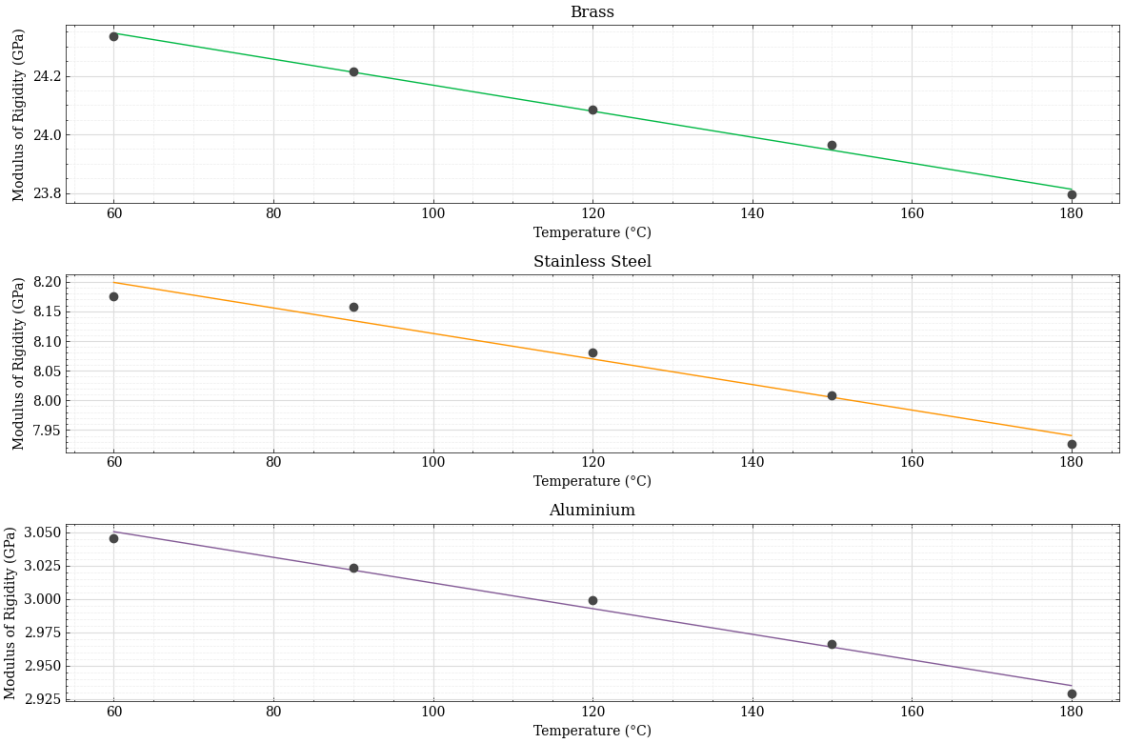
```
[3.04513328 3.02321174 2.99920264 2.96650239 2.92923503]
178.5928546
```

[81]:
```
Aval = np.polyfit(T, AG, 1)
Afit = np.poly1d(Aval) * 1e-9


plt.plot(T, AG * 1e-9, 'o', label='Data', zorder=3, color='#474747')
plt.plot(T, Afit(T), label='Linear Fit', color='#845B97')

plt.xlabel('Temperature (°C)')
plt.ylabel('Modulus of Rigidity (GPa)')
plt.title('Aluminium')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```



[82]:
```
fig, ax = plt.subplots(3, 1)

ax[0].plot(T, BG * 1e-9, 'o', label='Data', zorder=3, color='#474747')
ax[0].plot(T, Bfit(T), label='Linear Fit', color='#00B945')
ax[0].minorticks_on()
ax[0].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[0].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[0].set_ylabel('Modulus of Rigidity (GPa)')
ax[0].set_xlabel('Temperature (°C)')
ax[0].set_title('Brass')

ax[1].plot(T, SSG * 1e-9, 'o', label='Data', zorder=3, color='#474747')
ax[1].plot(T, SSfit(T), label='Linear Fit', color='#FF9500')
ax[1].minorticks_on()
ax[1].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[1].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[1].set_ylabel('Modulus of Rigidity (GPa)')
ax[1].set_xlabel('Temperature (°C)')
ax[1].set_title('Stainless Steel')
```

```
ax[2].plot(T, AG *1e-9, 'o', label='Data', zorder=3, color='#474747')
ax[2].plot(T, Afit(T), label='Linear Fit', color='#845B97')
ax[2].minorticks_on()
ax[2].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[2].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[2].set_ylabel('Modulus of Rigidity (GPa)')
ax[2].set_xlabel('Temperature (°C)')
ax[2].set_title('Aluminium')

plt.tight_layout()
plt.show()
```



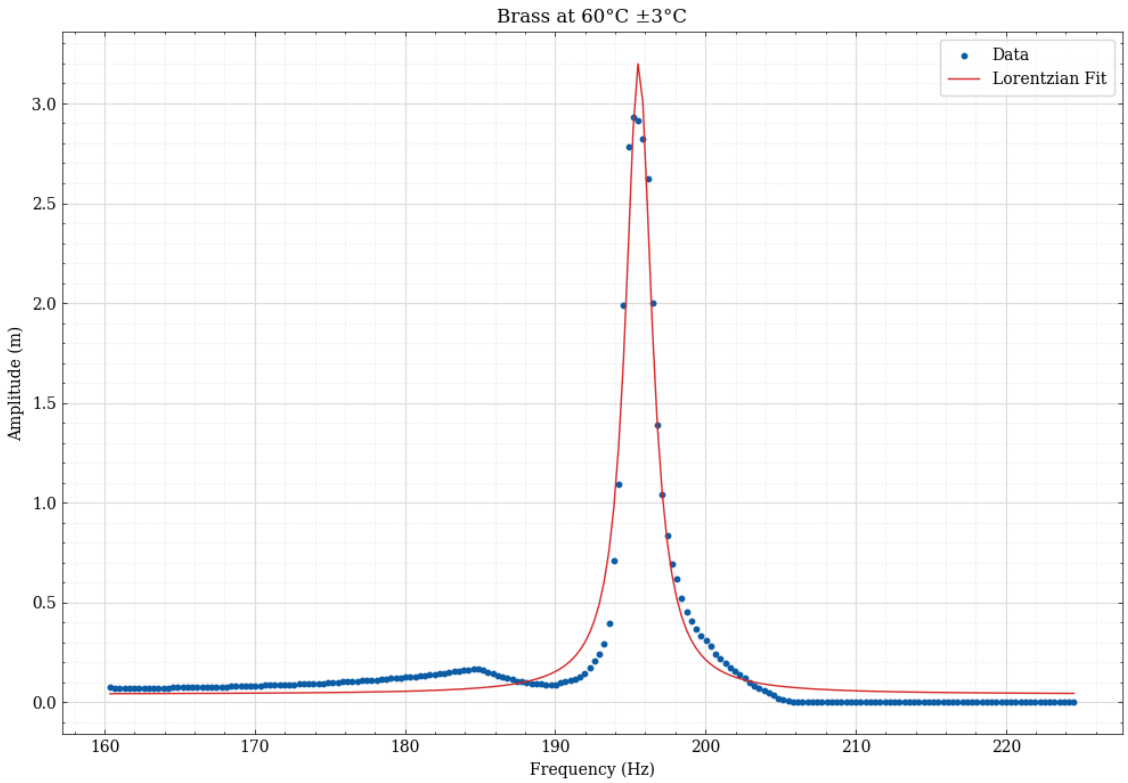
2 Fitting

[83]:
```
def lorentz_fit(x, x0, omega, A, y0):
    return y0 + (2 * A)/(np.pi) * omega/(4 * (x-x0)**2 + omega**2)

# A = area under curve
# omega = width of curve at half max A
# y0 = initial value
# x0 = peak
```

2.1 Brass

[84]:
```
B60C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/brass 60C')

VB60C = np.array(B60C[0]) * 64.15
ampB60C = np.array(B60C[1])
```

[85]:
```
initial_guess = [VB60C[np.argmax(ampB60C)], min(ampB60C), (max(ampB60C) -␣
 ↪min(ampB60C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB60C, ampB60C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VB60C, ampB60C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VB60C, lorentz_fit(VB60C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Brass at 60°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

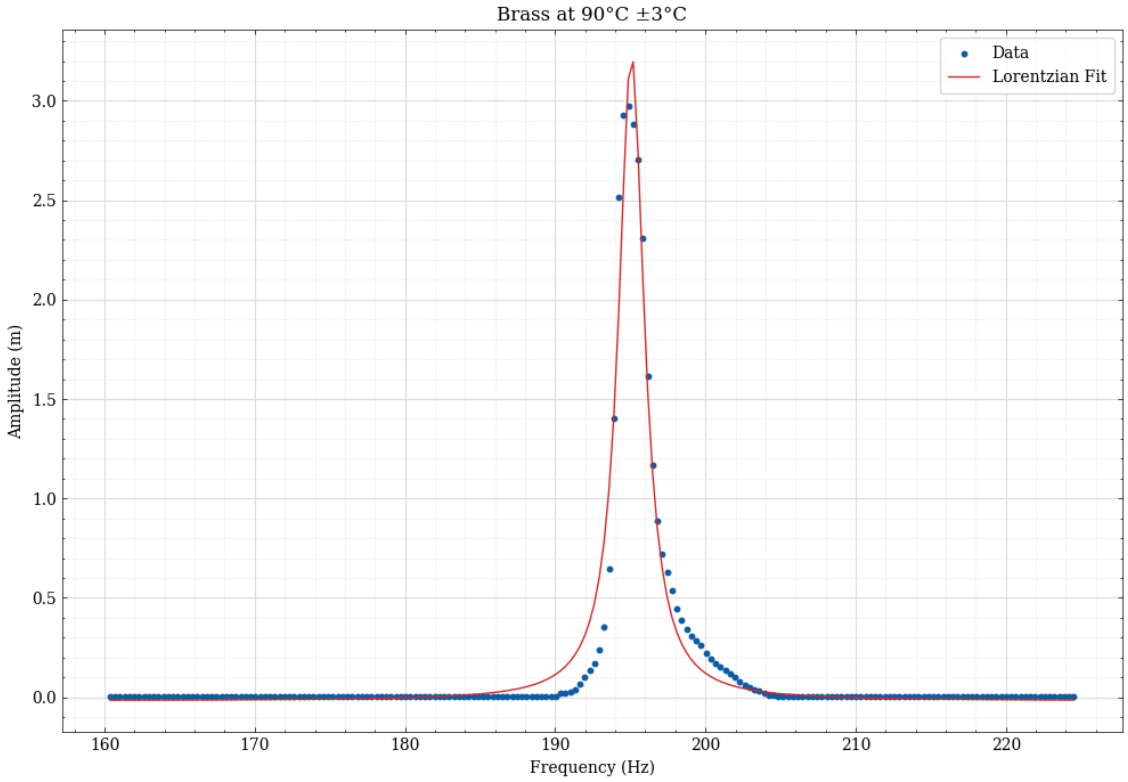



```
Peak from fit: 195.558052
Width of peak: 2.142843
```

[86]:
```
B90C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/brass 90C')

VB90C = np.array(B90C[0]) * 64.15
ampB90C = np.array(B90C[1])
```

[87]:
```
initial_guess = [VB90C[np.argmax(ampB90C)], min(ampB90C), (max(ampB90C) -␣
 ↪min(ampB90C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB90C, ampB90C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VB90C, ampB90C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VB90C, lorentz_fit(VB90C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Brass at 90°C ±3°C')
```

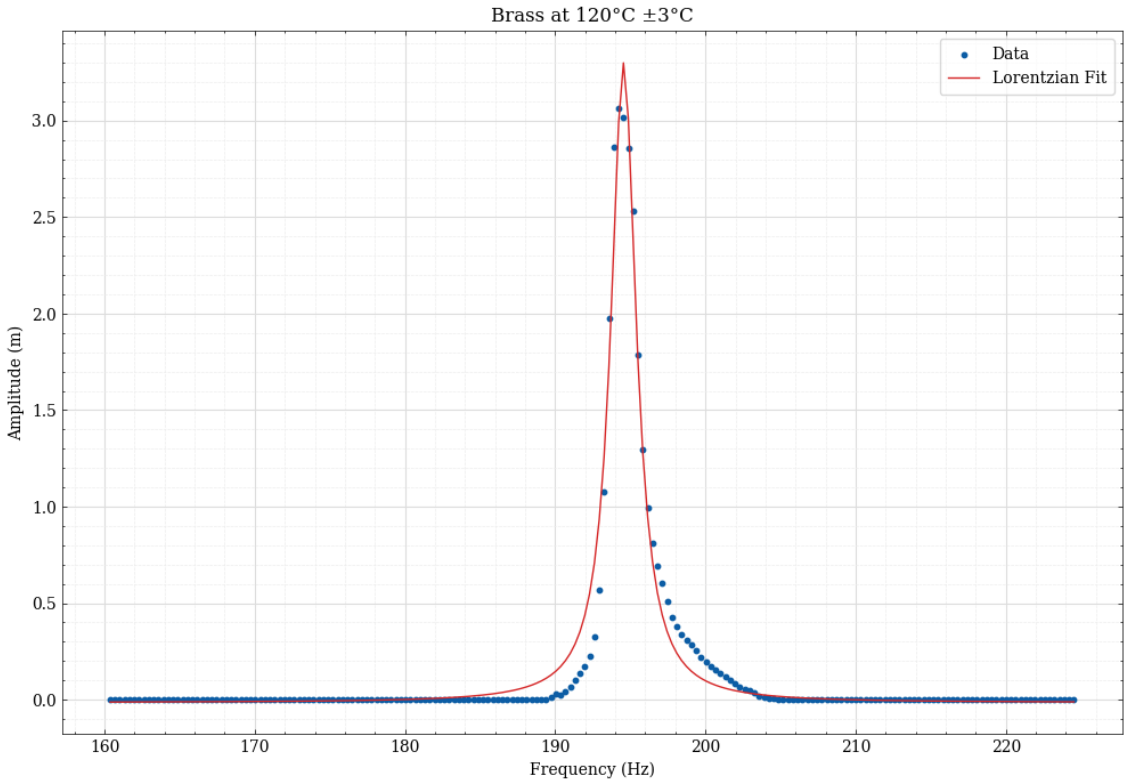
```
plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```




```
Peak from fit: 195.075226
Width of peak: 2.077667
```

[88]:
```
B120C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/brass 120C')

VB120C = np.array(B120C[0]) * 64.15
ampB120C = np.array(B120C[1])
```

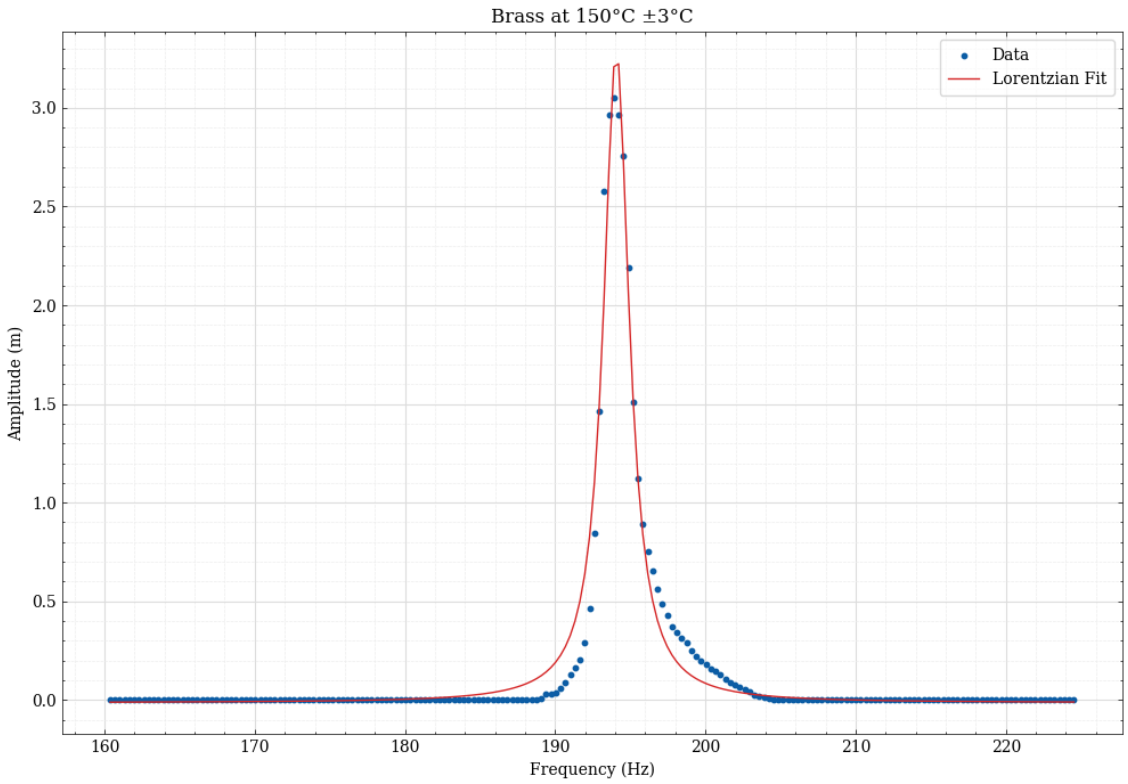
[89]:
```
initial_guess = [VB120C[np.argmax(ampB120C)], min(ampB120C), (max(ampB120C) -␣
 ↪min(ampB120C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB120C, ampB120C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VB120C, ampB120C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VB120C, lorentz_fit(VB120C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Brass at 120°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```



```
Peak from fit: 194.552084
Width of peak: 2.049796
```

[90]:
```
B150C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/brass 150C')

VB150C = np.array(B150C[0]) * 64.15
ampB150C = np.array(B150C[1])
```

[91]:
```
initial_guess = [VB150C[np.argmax(ampB150C)], min(ampB150C), (max(ampB150C) -␣
 ↪min(ampB150C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB150C, ampB150C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VB150C, ampB150C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VB150C, lorentz_fit(VB150C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Brass at 150°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

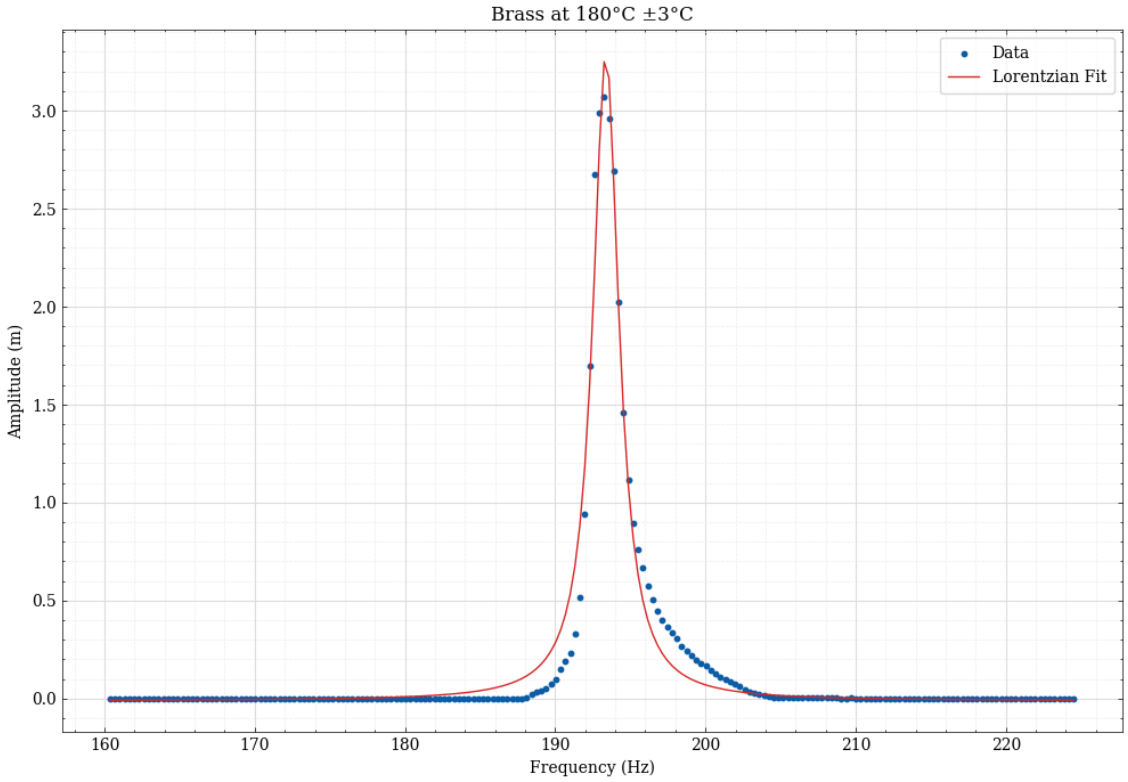



```
Peak from fit: 194.069886
Width of peak: 2.080339
```

[92]:
```
B180C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/brass 180C')

VB180C = np.array(B180C[0]) * 64.15
ampB180C = np.array(B180C[1])
```

[93]:
```
initial_guess = [VB180C[np.argmax(ampB180C)], min(ampB180C), (max(ampB180C) -␣
 ↪min(ampB180C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB180C, ampB180C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VB180C, ampB180C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VB180C, lorentz_fit(VB180C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Brass at 180°C ±3°C')
```

```
plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

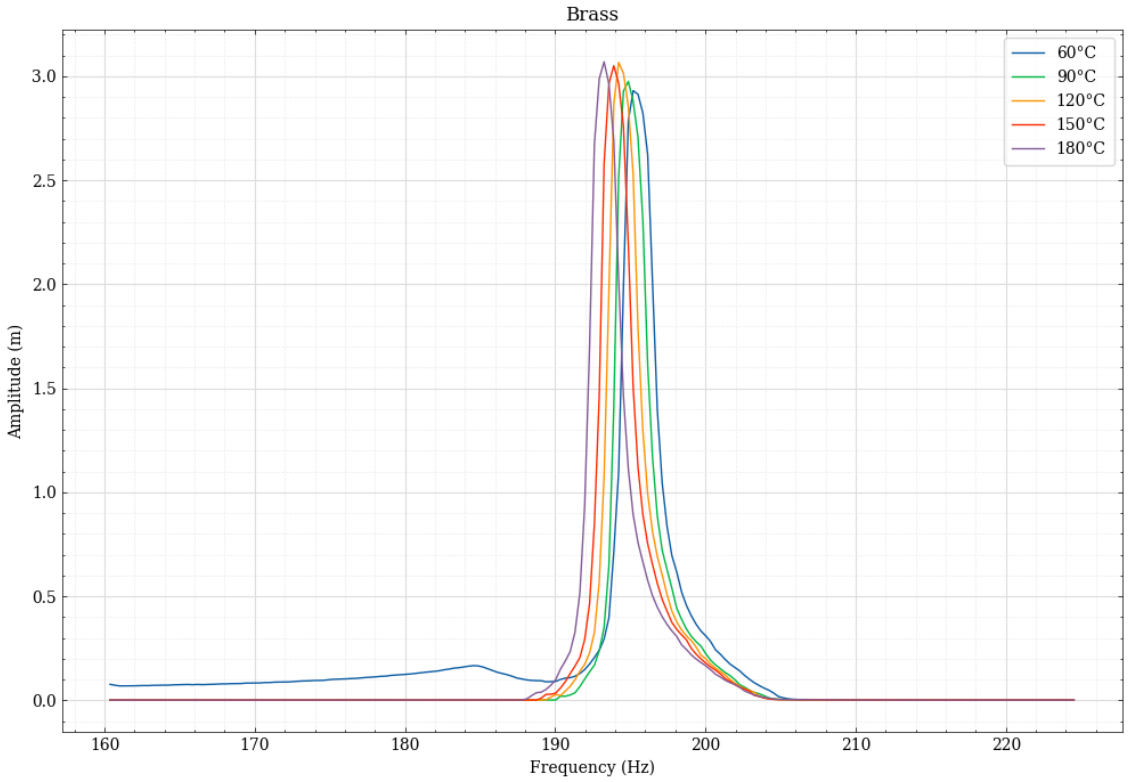



```
Peak from fit: 193.373124
Width of peak: 2.110216
```

2.1.1 ALL

[94]:
```
plt.plot(VB60C, ampB60C, label='60°C', zorder=3)
plt.plot(VB90C, ampB90C, label='90°C', zorder=4)
plt.plot(VB120C, ampB120C, label='120°C', zorder=5)
plt.plot(VB150C, ampB150C, label='150°C', zorder=6)
plt.plot(VB180C, ampB180C, label='180°C', zorder=7)
```

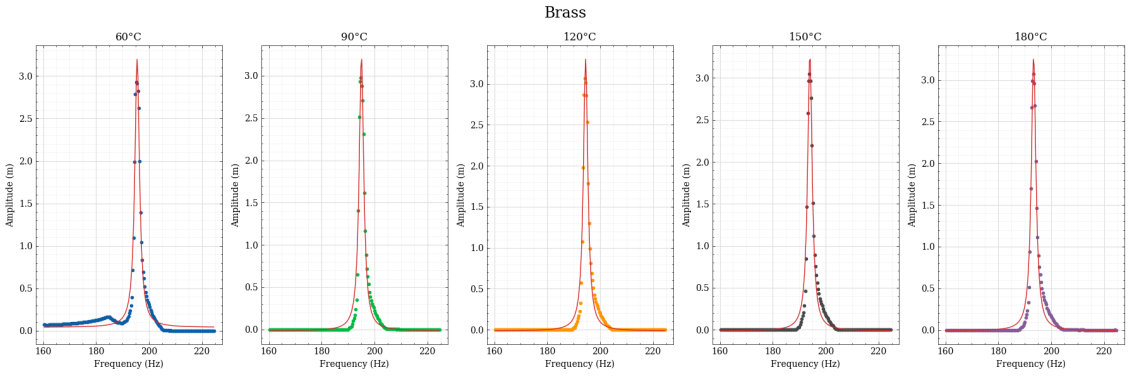
```
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Brass')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```



[95]:
```
fig, ax = plt.subplots(1,5, figsize=(18,6))
ax = ax.flatten()

initial_guess = [VB60C[np.argmax(ampB60C)], min(ampB60C), (max(ampB60C) -␣
 ↪min(ampB60C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB60C, ampB60C, p0=initial_guess)
x0, omega, A, y0 = popt
```

```
ax[0].plot(VB60C, ampB60C, zorder=3, marker='.', linestyle='none', label='60°C')
ax[0].plot(VB60C, lorentz_fit(VB60C, *popt), color='tab:red', zorder=4)
ax[0].minorticks_on()
ax[0].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[0].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[0].set_xlabel('Frequency (Hz)')
ax[0].set_ylabel('Amplitude (m)')
ax[0].set_title('60°C')

initial_guess = [VB90C[np.argmax(ampB90C)], min(ampB90C), (max(ampB90C) -␣
 ↪min(ampB90C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB90C, ampB90C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[1].plot(VB90C, ampB90C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#00B945', label='90°C')
ax[1].plot(VB90C, lorentz_fit(VB90C, *popt), color='tab:red', zorder=4)
ax[1].minorticks_on()
ax[1].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[1].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Amplitude (m)')
ax[1].set_title('90°C')

initial_guess = [VB120C[np.argmax(ampB120C)], min(ampB120C), (max(ampB120C) -␣
 ↪min(ampB120C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB120C, ampB120C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[2].plot(VB120C, ampB120C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#FF9500', label='120°C')
ax[2].plot(VB120C, lorentz_fit(VB120C, *popt), color='tab:red', zorder=4)
ax[2].minorticks_on()
ax[2].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[2].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[2].set_xlabel('Frequency (Hz)')
ax[2].set_ylabel('Amplitude (m)')
ax[2].set_title('120°C')

initial_guess = [VB150C[np.argmax(ampB150C)], min(ampB150C), (max(ampB150C) -␣
 ↪min(ampB150C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB150C, ampB150C, p0=initial_guess)
x0, omega, A, y0 = popt
```

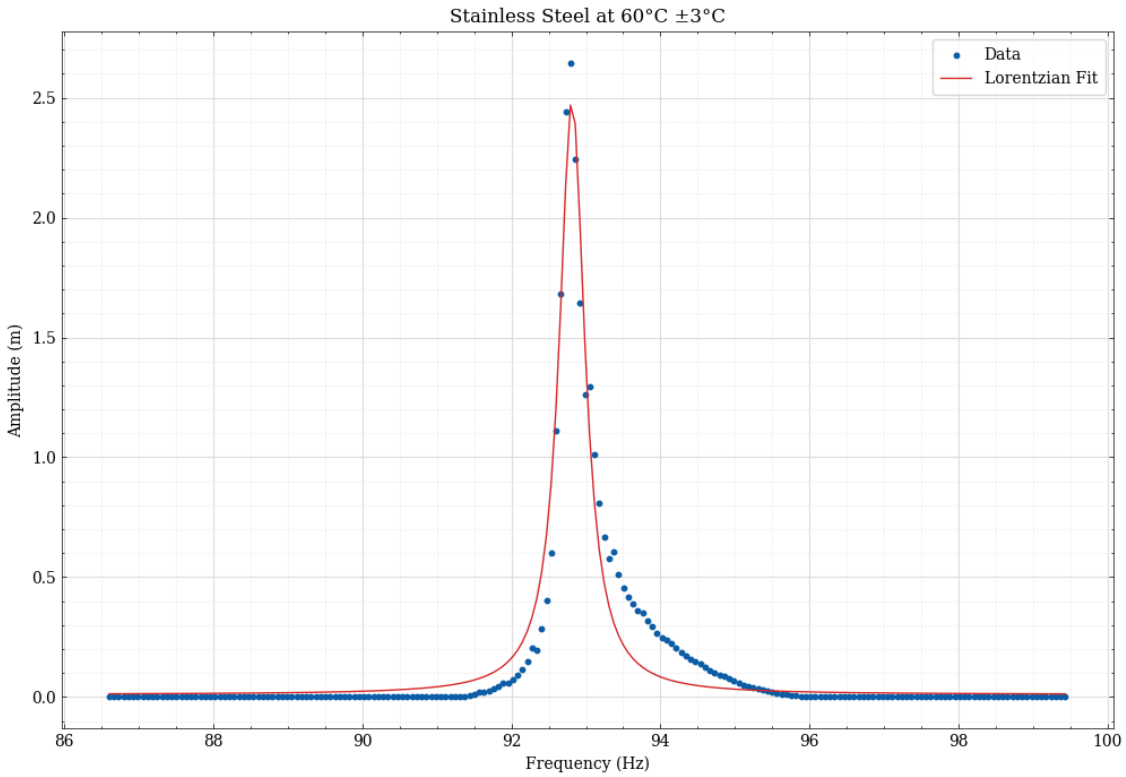
```
ax[3].plot(VB150C, ampB150C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#474747', label='150°C')
ax[3].plot(VB150C, lorentz_fit(VB150C, *popt), color='tab:red', zorder=4)
ax[3].minorticks_on()
ax[3].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[3].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[3].set_xlabel('Frequency (Hz)')
ax[3].set_ylabel('Amplitude (m)')
ax[3].set_title('150°C')

initial_guess = [VB180C[np.argmax(ampB180C)], min(ampB180C), (max(ampB180C) -␣
 ↪min(ampB180C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VB180C, ampB180C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[4].plot(VB180C, ampB180C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#845B97', label='180°C')
ax[4].plot(VB180C, lorentz_fit(VB180C, *popt), color='tab:red', zorder=4, ␣
 ↪label='Lorentzian Fit')
ax[4].minorticks_on()
ax[4].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[4].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[4].set_xlabel('Frequency (Hz)')
ax[4].set_ylabel('Amplitude (m)')
ax[4].set_title('180°C')

fig.suptitle('Brass', size=17)
fig.tight_layout()

plt.show()
```



2.2 Stainless Steel

[96]:
```
SS60C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/stainless steel 60C2')

VSS60C = np.array(SS60C[0]) * 64.15
ampSS60C = np.array(SS60C[1])
```

[97]:
```
initial_guess = [VSS60C[np.argmax(ampSS60C)], min(ampSS60C), (max(ampSS60C) -␣
 ↪min(ampSS60C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS60C, ampSS60C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VSS60C, ampSS60C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VSS60C, lorentz_fit(VSS60C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Stainless Steel at 60°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

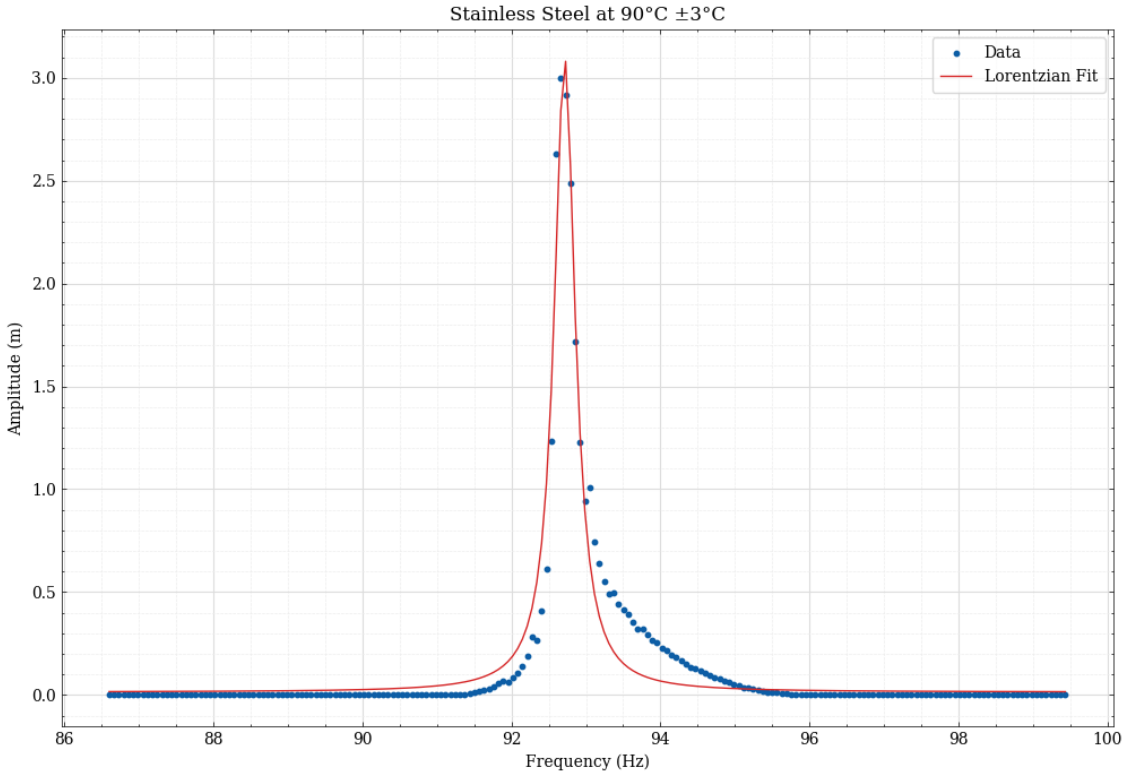



```
Peak from fit: 92.813146
Width of peak: 0.414041
```

[98]:
```
SS90C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/stainless steel 90C2')

VSS90C = np.array(SS90C[0]) * 64.15
ampSS90C = np.array(SS90C[1])
```

[99]:
```
initial_guess = [VSS90C[np.argmax(ampSS90C)], min(ampSS90C), (max(ampSS90C) -␣
 ↪min(ampSS90C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS90C, ampSS90C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VSS90C, ampSS90C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VSS90C, lorentz_fit(VSS90C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Stainless Steel at 90°C ±3°C')
```

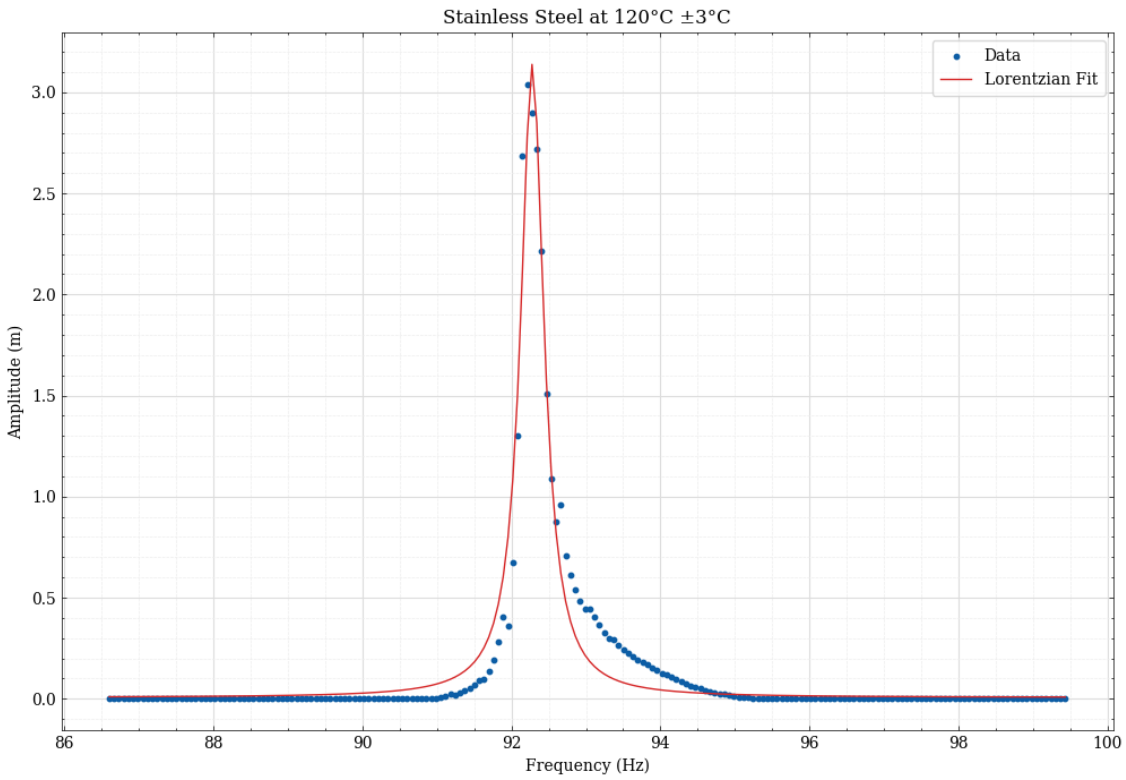
```
plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```




```
Peak from fit: 92.714482
Width of peak: 0.342378
```

[100]:
```
SS120C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/stainless steel 120C2')

VSS120C = np.array(SS120C[0]) * 64.15
ampSS120C = np.array(SS120C[1])
```

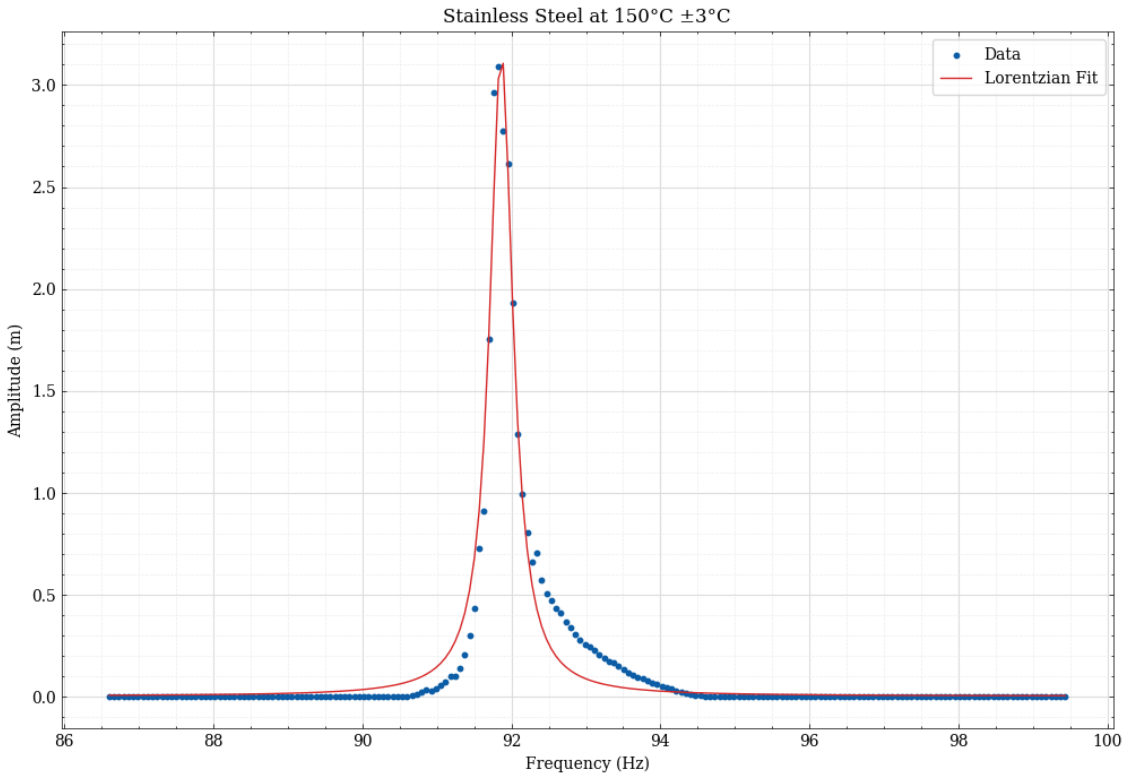
[101]:
```
initial_guess = [VSS120C[np.argmax(ampSS120C)], min(ampSS120C), (max(ampSS120C)␣
 ↪- min(ampSS120C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS120C, ampSS120C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VSS120C, ampSS120C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VSS120C, lorentz_fit(VSS120C, *popt), color='tab:red',␣
 ↪label='Lorentzian Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Stainless Steel at 120°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```



```
Peak from fit: 92.279800
Width of peak: 0.378768
```

[102]:
```
SS150C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/stainless steel 150C2')

VSS150C = np.array(SS150C[0]) * 64.15
ampSS150C = np.array(SS150C[1])
```

[103]:
```
initial_guess = [VSS150C[np.argmax(ampSS150C)], min(ampSS150C), (max(ampSS150C)␣
 ↪- min(ampSS150C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS150C, ampSS150C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VSS150C, ampSS150C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VSS150C, lorentz_fit(VSS150C, *popt), color='tab:red',␣
 ↪label='Lorentzian Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Stainless Steel at 150°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

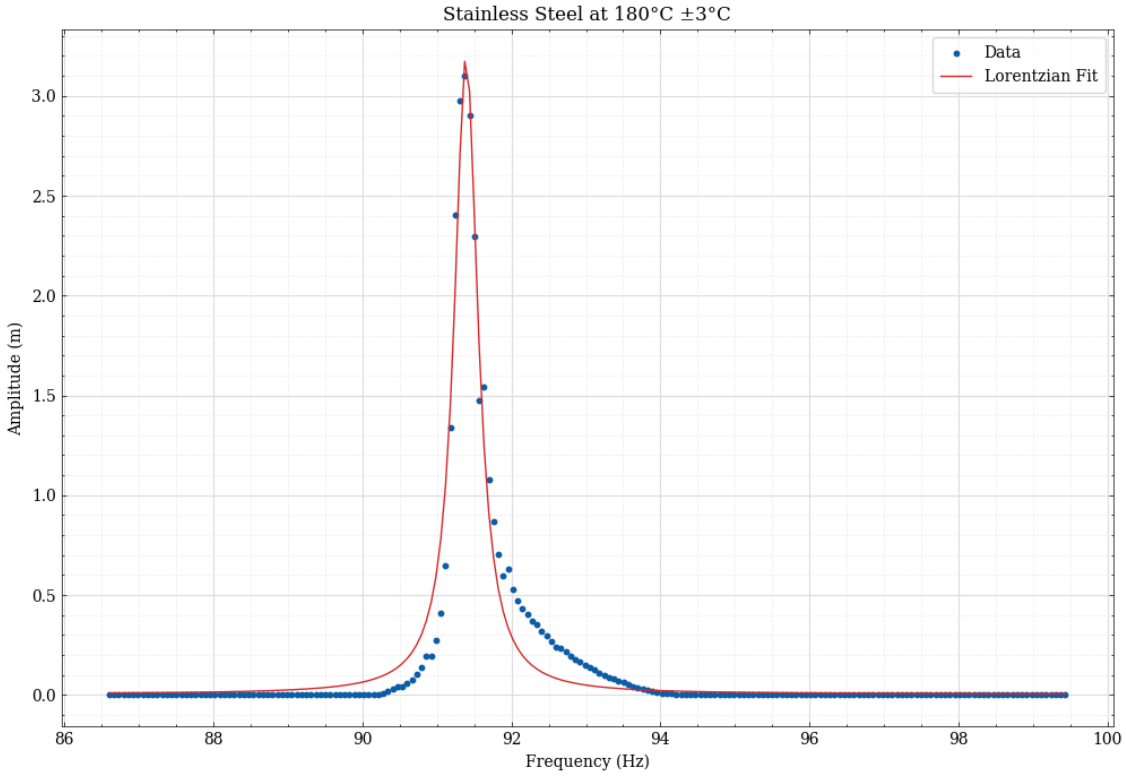



```
Peak from fit: 91.864033
Width of peak: 0.377263
```

[104]:
```
SS180C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/stainless steel 180C2')

VSS180C = np.array(SS180C[0]) * 64.15
ampSS180C = np.array(SS180C[1])
```

[105]:
```
initial_guess = [VSS180C[np.argmax(ampSS180C)], min(ampSS180C), (max(ampSS180C)␣
 ↪- min(ampSS180C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS180C, ampSS180C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VSS180C, ampSS180C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VSS180C, lorentz_fit(VSS180C, *popt), color='tab:red',␣
 ↪label='Lorentzian Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Stainless Steel at 180°C ±3°C')
```

```
plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

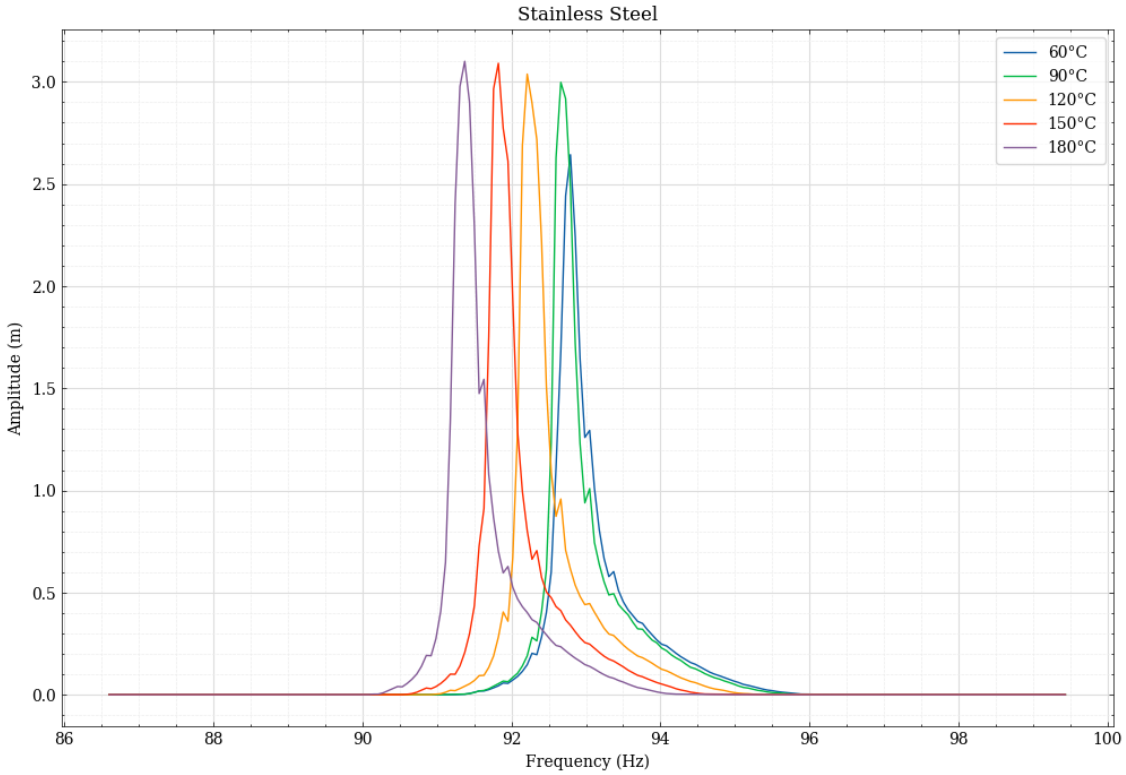



```
Peak from fit: 91.391520
Width of peak: 0.383930
```

2.2.1 ALL

[106]:
```
plt.plot(VSS60C, ampSS60C, label='60°C', zorder=3)
plt.plot(VSS90C, ampSS90C, label='90°C', zorder=4)
plt.plot(VSS120C, ampSS120C, label='120°C', zorder=5)
plt.plot(VSS150C, ampSS150C, label='150°C', zorder=6)
plt.plot(VSS180C, ampSS180C, label='180°C', zorder=7)
```

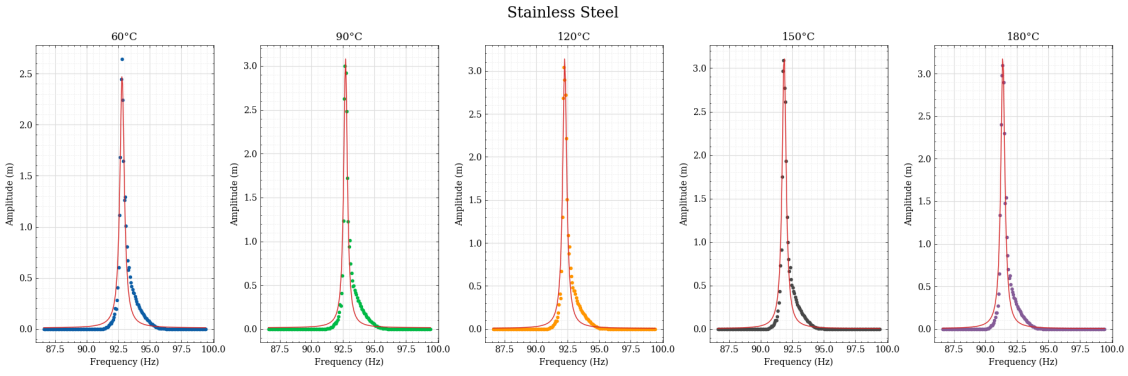
```
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Stainless Steel')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```



[107]:
```
fig, ax = plt.subplots(1,5, figsize=(18,6))
ax = ax.flatten()

initial_guess = [VSS60C[np.argmax(ampSS60C)], min(ampSS60C), (max(ampSS60C) -␣
 ↪min(ampSS60C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS60C, ampSS60C, p0=initial_guess)
x0, omega, A, y0 = popt
```

```
ax[0].plot(VSS60C, ampSS60C, zorder=3, marker='.', linestyle='none',␣
 ↪label='60°C')
ax[0].plot(VSS60C, lorentz_fit(VSS60C, *popt), color='tab:red', zorder=4)
ax[0].minorticks_on()
ax[0].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[0].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[0].set_xlabel('Frequency (Hz)')
ax[0].set_ylabel('Amplitude (m)')
ax[0].set_title('60°C')

initial_guess = [VSS90C[np.argmax(ampSS90C)], min(ampSS90C), (max(ampSS90C) -␣
 ↪min(ampSS90C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS90C, ampSS90C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[1].plot(VSS90C, ampSS90C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#00B945', label='90°C')
ax[1].plot(VSS90C, lorentz_fit(VSS90C, *popt), color='tab:red', zorder=4)
ax[1].minorticks_on()
ax[1].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[1].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Amplitude (m)')
ax[1].set_title('90°C')

initial_guess = [VSS120C[np.argmax(ampSS120C)], min(ampSS120C), (max(ampSS120C)␣
 ↪- min(ampSS120C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS120C, ampSS120C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[2].plot(VSS120C, ampSS120C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#FF9500', label='120°C')
ax[2].plot(VSS120C, lorentz_fit(VSS120C, *popt), color='tab:red', zorder=4)
ax[2].minorticks_on()
ax[2].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[2].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[2].set_xlabel('Frequency (Hz)')
ax[2].set_ylabel('Amplitude (m)')
ax[2].set_title('120°C')

initial_guess = [VSS150C[np.argmax(ampSS150C)], min(ampSS150C), (max(ampSS150C)␣
 ↪- min(ampSS150C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS150C, ampSS150C, p0=initial_guess)
```

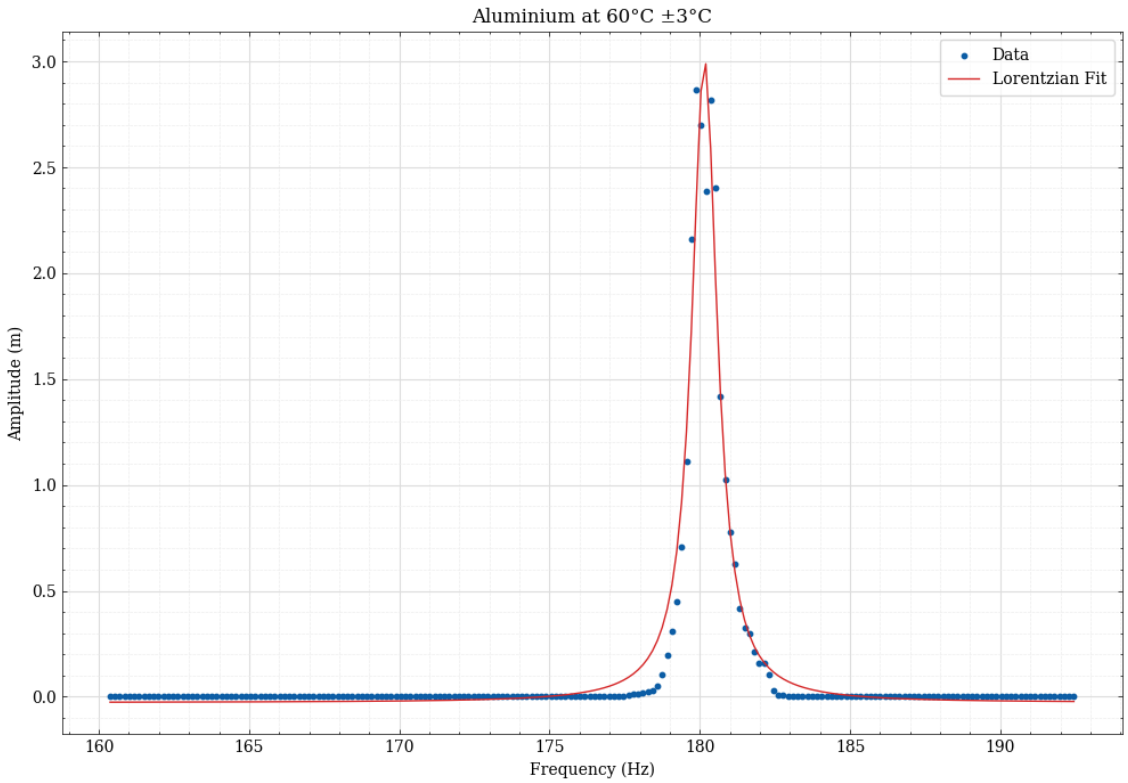
```
x0, omega, A, y0 = popt

ax[3].plot(VSS150C, ampSS150C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#474747', label='150°C')
ax[3].plot(VSS150C, lorentz_fit(VSS150C, *popt), color='tab:red', zorder=4)
ax[3].minorticks_on()
ax[3].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[3].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[3].set_xlabel('Frequency (Hz)')
ax[3].set_ylabel('Amplitude (m)')
ax[3].set_title('150°C')

initial_guess = [VSS120C[np.argmax(ampSS180C)], min(ampSS180C), (max(ampSS180C)␣
 ↪- min(ampSS180C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VSS180C, ampSS180C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[4].plot(VSS180C, ampSS180C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#845B97', label='180°C')
ax[4].plot(VSS180C, lorentz_fit(VSS180C, *popt), color='tab:red', zorder=4, ␣
 ↪label='Lorentzian Fit')
ax[4].minorticks_on()
ax[4].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[4].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[4].set_xlabel('Frequency (Hz)')
ax[4].set_ylabel('Amplitude (m)')
ax[4].set_title('180°C')

fig.suptitle('Stainless Steel', size=17)
fig.tight_layout()

plt.show()
```



2.3 Aluminium

[108]:
```
A60C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/aluminium 60C2')

VA60C = np.array(A60C[0]) * 64.15
ampA60C = np.array(A60C[1])
```

[109]:
```
initial_guess = [VA60C[np.argmax(ampA60C)], min(ampA60C), (max(ampA60C) -␣
 ↪min(ampA60C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA60C, ampA60C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VA60C, ampA60C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VA60C, lorentz_fit(VA60C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Aluminium at 60°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

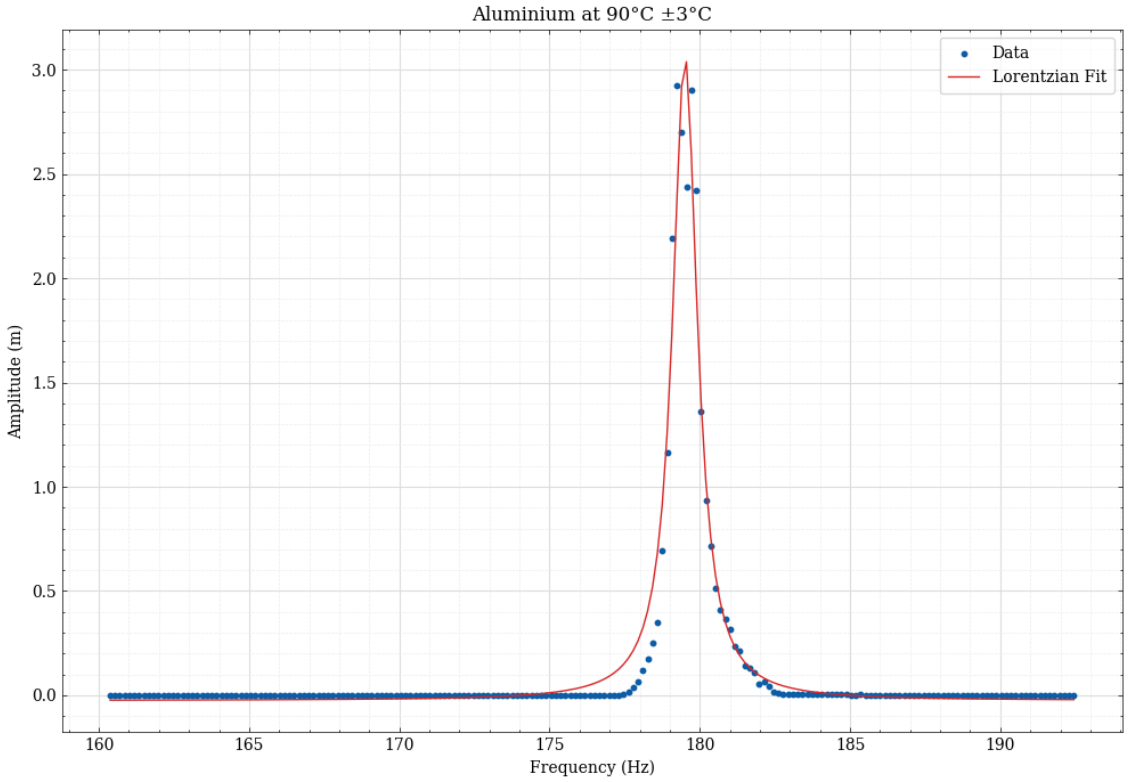



```
Peak from fit: 180.156115
Width of peak: 1.023388
```

[110]:
```
A90C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/aluminium 90C2')

VA90C = np.array(A90C[0]) * 64.15
ampA90C = np.array(A90C[1])
```

[111]:
```
initial_guess = [VA90C[np.argmax(ampA90C)], np.median(ampA90C), (np.
 ↪max(ampA90C) - np.min(ampA90C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA90C, ampA90C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VA90C, ampA90C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VA90C, lorentz_fit(VA90C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Aluminium at 90°C ±3°C')
```

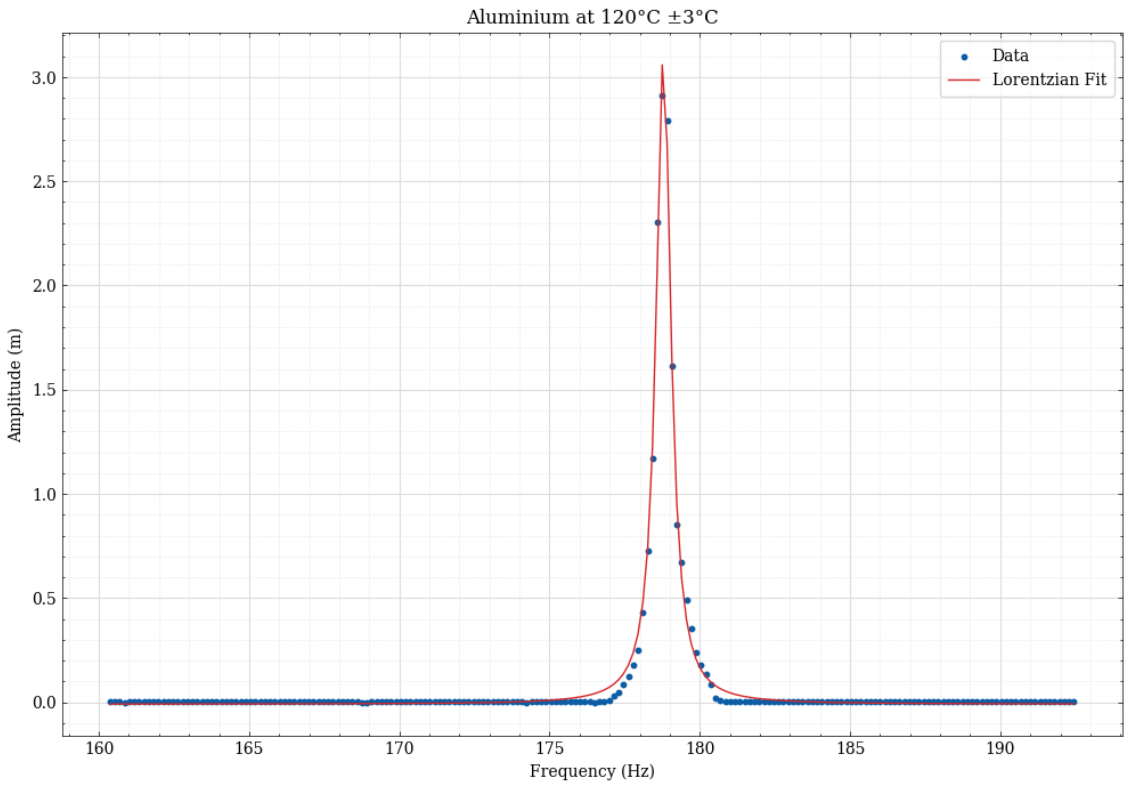
```
plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```




```
Peak from fit: 179.506483
Width of peak: 0.997994
```

[112]:
```
A120C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/aluminium 120C2')

VA120C = np.array(A120C[0]) * 64.15
ampA120C = np.array(A120C[1])
```

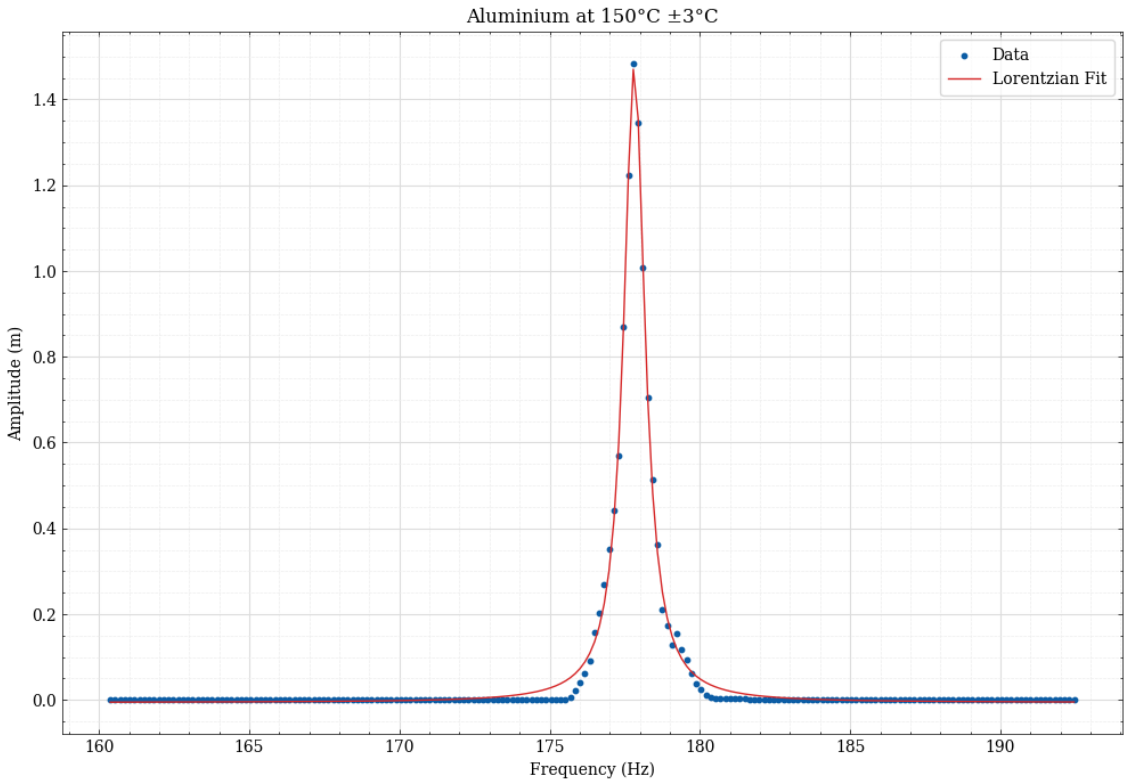
[113]:
```
initial_guess = [VA120C[np.argmax(ampA120C)], min(ampA120C), (max(ampA120C) -␣
 ↪min(ampA120C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA120C, ampA120C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VA120C, ampA120C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VA120C, lorentz_fit(VA120C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Aluminium at 120°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```



```
Peak from fit: 178.792279
Width of peak: 0.589699
```

[114]:
```
A150C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/aluminium 150C2')

VA150C = np.array(A150C[0]) * 64.15
ampA150C = np.array(A150C[1])
```

[115]:
```
initial_guess = [VA150C[np.argmax(ampA150C)], min(ampA150C), (max(ampA150C) -␣
 ↪min(ampA150C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA150C, ampA150C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VA150C, ampA150C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VA150C, lorentz_fit(VA150C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Aluminium at 150°C ±3°C')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

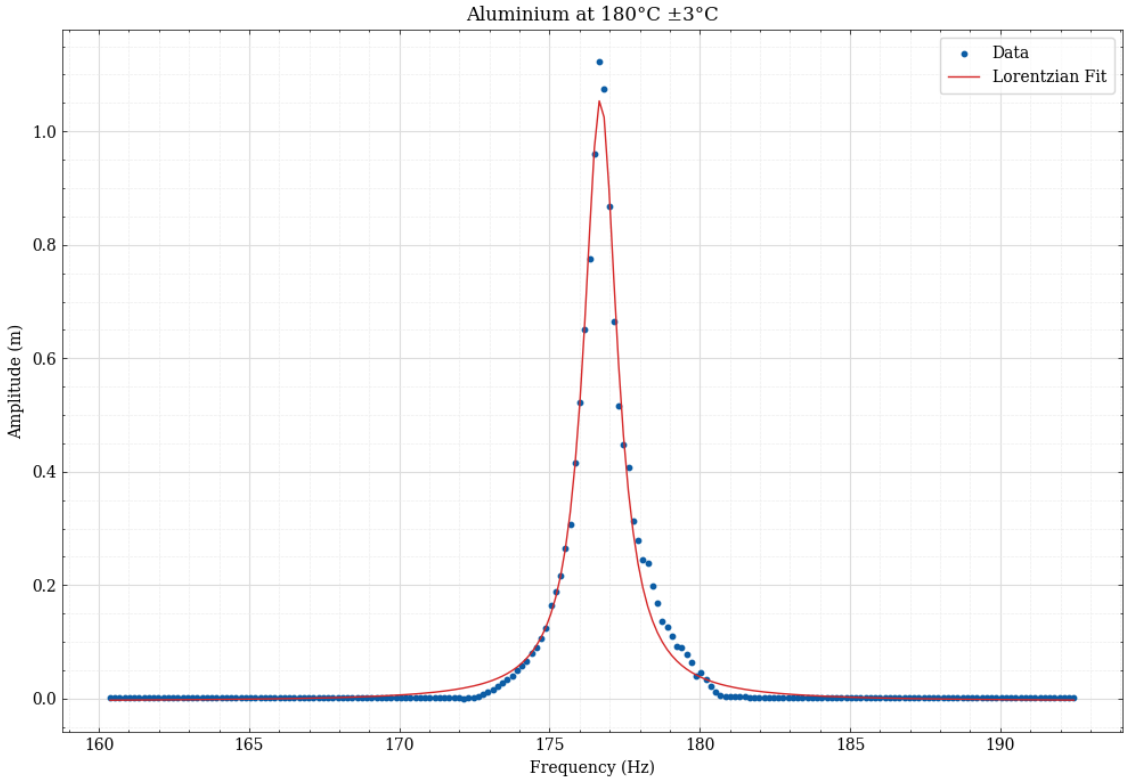



```
Peak from fit: 177.814923
Width of peak: 0.857605
```

[116]:
```
A180C = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/aluminium 180C2')

VA180C = np.array(A180C[0]) * 64.15
ampA180C = np.array(A180C[1])
```

[117]:
```
initial_guess = [VA180C[np.argmax(ampA180C)], min(ampA180C), (max(ampA180C) -␣
 ↪min(ampA180C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA180C, ampA180C, p0=initial_guess)
x0, omega, A, y0 = popt

plt.scatter(VA180C, ampA180C, label='Data', zorder=3, marker='o', s=10)
plt.plot(VA180C, lorentz_fit(VA180C, *popt), color='tab:red', label='Lorentzian␣
 ↪Fit', zorder=4)

plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Aluminium at 180°C ±3°C')
```

```
plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()

print(f'Peak from fit: {x0:.6f}')
print(f'Width of peak: {abs(omega):.6f}')
```

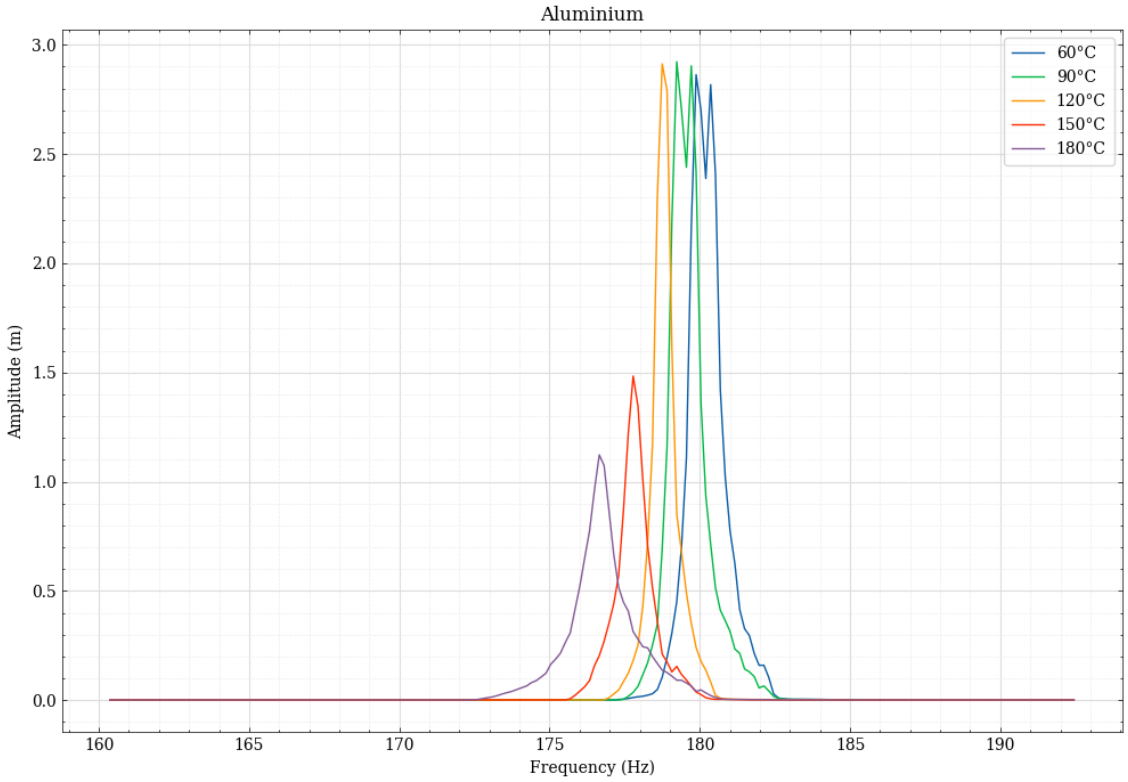



```
Peak from fit: 176.694473
Width of peak: 1.359736
```

2.3.1 ALL

[118]:
```
plt.plot(VA60C, ampA60C, label='60°C', zorder=3)
plt.plot(VA90C, ampA90C, label='90°C', zorder=4)
plt.plot(VA120C, ampA120C, label='120°C', zorder=5)
plt.plot(VA150C, ampA150C, label='150°C', zorder=6)
plt.plot(VA180C, ampA180C, label='180°C', zorder=7)
```

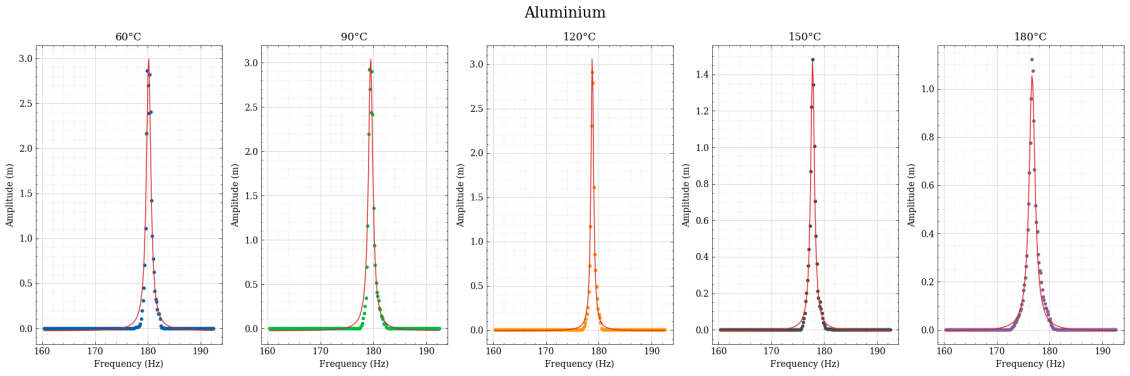
```
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.title('Aluminium')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```



[119]:
```
fig, ax = plt.subplots(1,5, figsize=(18,6))
ax = ax.flatten()

initial_guess = [VA60C[np.argmax(ampA60C)], min(ampA60C), (max(ampA60C) -␣
 ↪min(ampA60C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA60C, ampA60C, p0=initial_guess)
x0, omega, A, y0 = popt
```

```
ax[0].plot(VA60C, ampA60C, zorder=3, marker='.', linestyle='none', label='60°C')
ax[0].plot(VA60C, lorentz_fit(VA60C, *popt), color='tab:red', zorder=4)
ax[0].minorticks_on()
ax[0].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[0].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[0].set_xlabel('Frequency (Hz)')
ax[0].set_ylabel('Amplitude (m)')
ax[0].set_title('60°C')

initial_guess = [VA90C[np.argmax(ampA90C)], np.median(ampA90C), (max(ampA90C) -␣
 ↪min(ampA90C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA90C, ampA90C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[1].plot(VA90C, ampA90C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#00B945', label='90°C')
ax[1].plot(VA90C, lorentz_fit(VA90C, *popt), color='tab:red', zorder=4)
ax[1].minorticks_on()
ax[1].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[1].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Amplitude (m)')
ax[1].set_title('90°C')

initial_guess = [VA120C[np.argmax(ampA120C)], min(ampA120C), (max(ampA120C) -␣
 ↪min(ampA120C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA120C, ampA120C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[2].plot(VA120C, ampA120C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#FF9500', label='120°C')
ax[2].plot(VA120C, lorentz_fit(VA120C, *popt), color='tab:red', zorder=4)
ax[2].minorticks_on()
ax[2].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[2].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[2].set_xlabel('Frequency (Hz)')
ax[2].set_ylabel('Amplitude (m)')
ax[2].set_title('120°C')

initial_guess = [VA150C[np.argmax(ampA150C)], min(ampA150C), (max(ampA150C) -␣
 ↪min(ampA150C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA150C, ampA150C, p0=initial_guess)
x0, omega, A, y0 = popt
```

```
ax[3].plot(VA150C, ampA150C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#474747', label='150°C')
ax[3].plot(VA150C, lorentz_fit(VA150C, *popt), color='tab:red', zorder=4)
ax[3].minorticks_on()
ax[3].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[3].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[3].set_xlabel('Frequency (Hz)')
ax[3].set_ylabel('Amplitude (m)')
ax[3].set_title('150°C')

initial_guess = [VA120C[np.argmax(ampA180C)], min(ampA180C), (max(ampA180C) -␣
 ↪min(ampA180C))*0.5, 0.5]
popt, pcov = curve_fit(lorentz_fit, VA180C, ampA180C, p0=initial_guess)
x0, omega, A, y0 = popt

ax[4].plot(VA180C, ampA180C, zorder=3, marker='.', linestyle='none',␣
 ↪color='#845B97', label='180°C')
ax[4].plot(VA180C, lorentz_fit(VA180C, *popt), color='tab:red', zorder=4, ␣
 ↪label='Lorentzian Fit')
ax[4].minorticks_on()
ax[4].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[4].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[4].set_xlabel('Frequency (Hz)')
ax[4].set_ylabel('Amplitude (m)')
ax[4].set_title('180°C')

fig.suptitle('Aluminium', size=17)
fig.tight_layout()

plt.show()
```



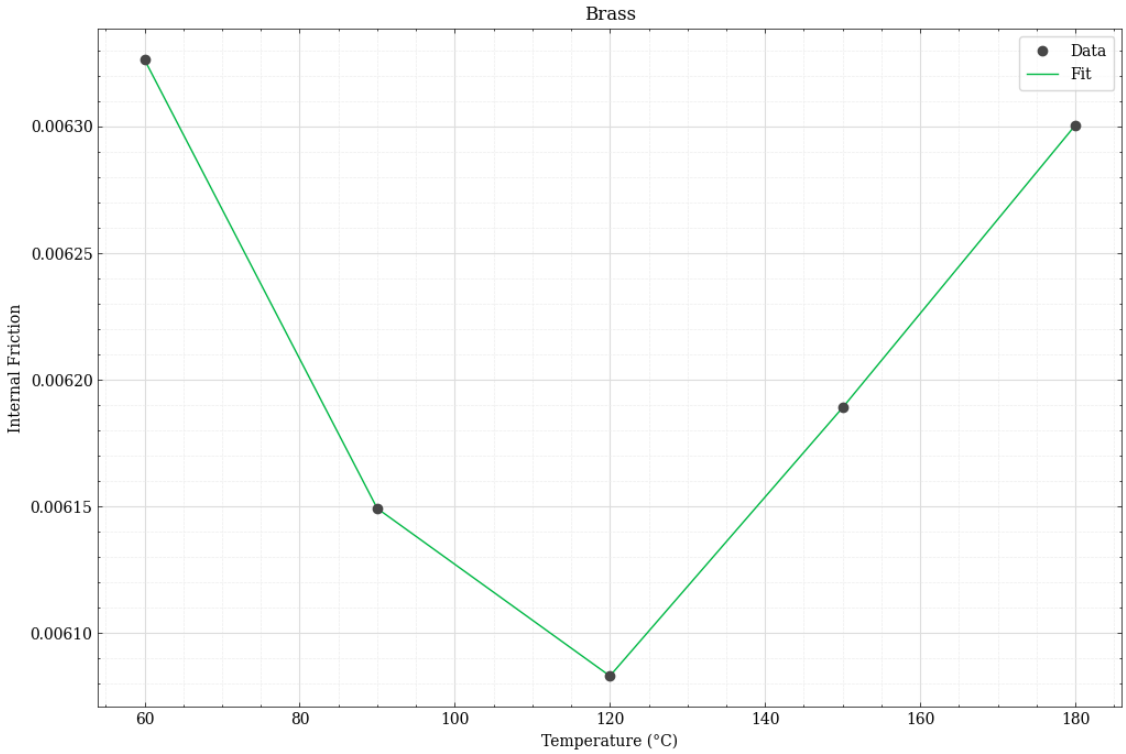
3 Internal Friction

[120]:
```
# eq.6
def int_friction(domega, omega0):
    return domega / (np.sqrt(3) * omega0)

# domega = width of curve at half max A
# omega0 = angular frequency
```

[121]:
```
# brass

fres = np.array([195.558052, 195.075226, 194.552084, 194.069886, 193.373124])
omega0 = 2 * np.pi * fres
omega = np.array([2.142843, 2.077667, 2.049796, 2.080339, 2.110216])
domega = 2 * np.pi * omega

BIF = int_friction(domega, omega0)

print(BIF)
```

```
[0.00632636 0.00614912 0.00608295 0.00618893 0.00630043]
```

[122]:
```
plt.plot(T, BIF, 'o', label='Data', zorder=3, color='#474747')
plt.plot(T, BIF, label='Fit', color='#00B945')

plt.xlabel('Temperature (°C)')
plt.ylabel('Internal Friction')
plt.title('Brass')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```

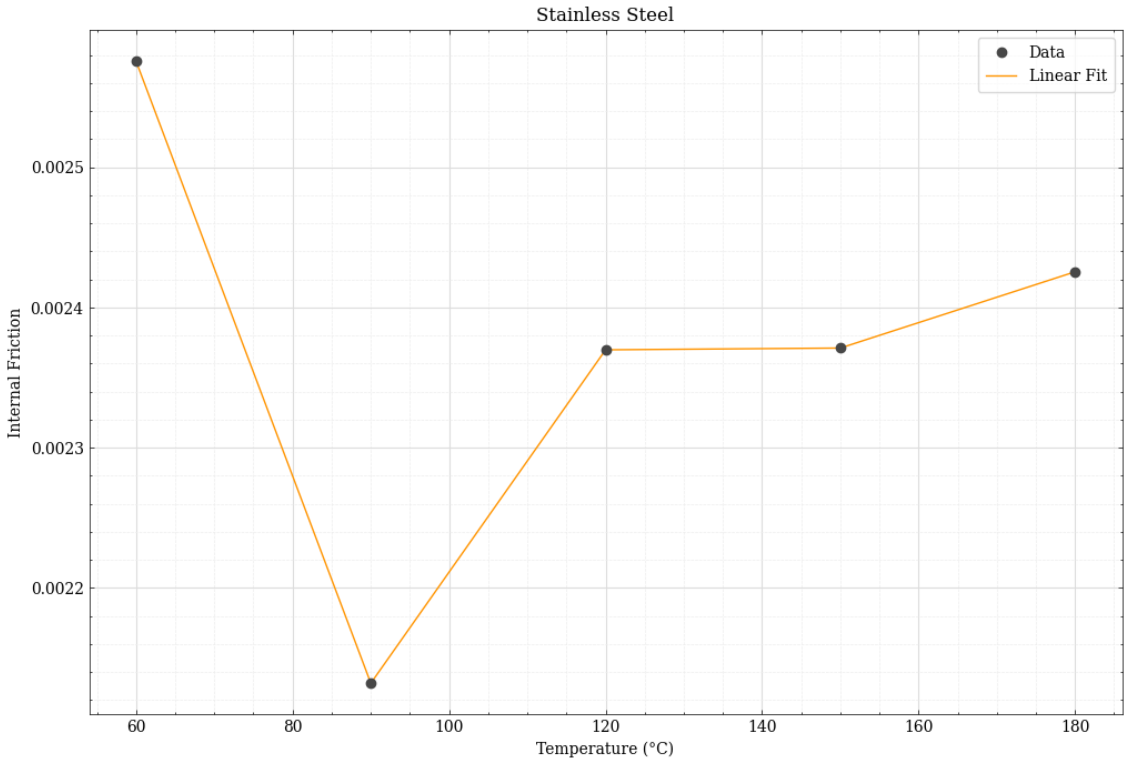


[123]:
```
# steel

fres = np.array([92.813146, 92.714482, 92.279800, 91.864033, 91.391520])
omega0 = 2 * np.pi * fres
omega = np.array([0.414041, 0.342378, 0.378768, 0.377263, 0.383930])
domega = 2 * np.pi * omega

SSIF = int_friction(domega, omega0)

print(SSIF)
```

```
[0.00257557 0.00213205 0.00236977 0.00237104 0.00242541]
```

[124]:
```
plt.plot(T, SSIF, 'o', label='Data', zorder=3, color='#474747')
plt.plot(T, SSIF, label='Linear Fit', color='#FF9500')

plt.xlabel('Temperature (°C)')
plt.ylabel('Internal Friction')
plt.title('Stainless Steel')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
```

```
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```

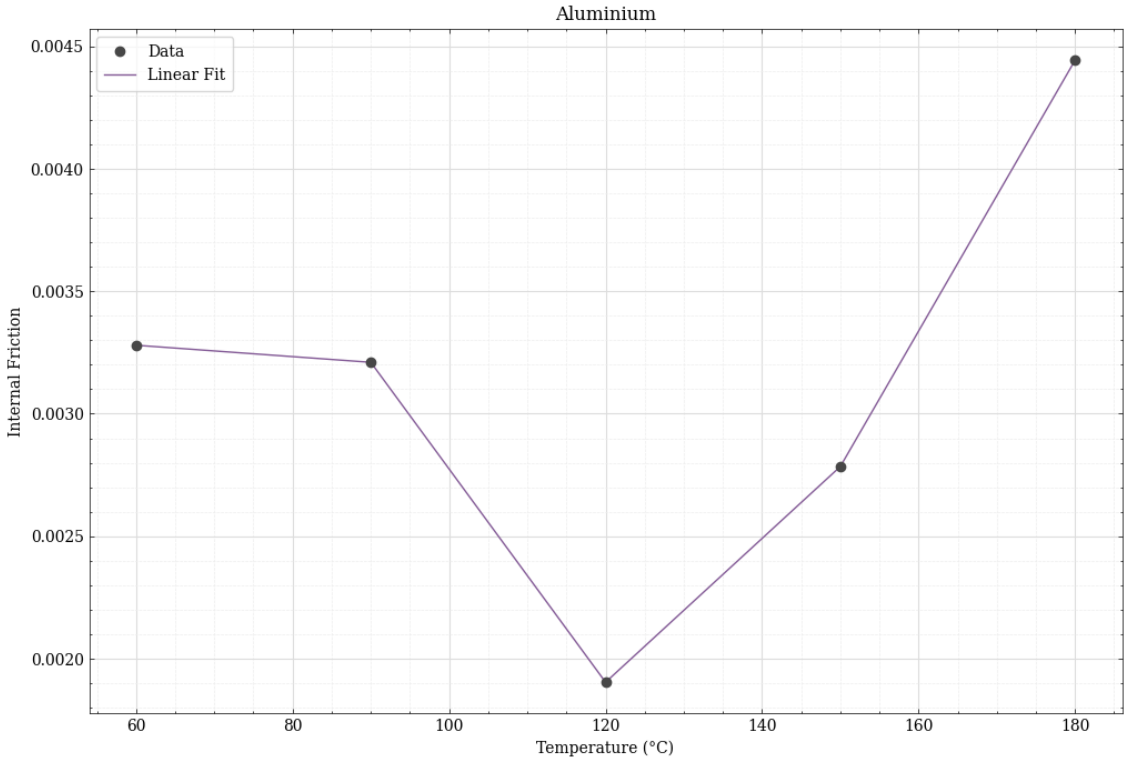


[125]:
```
# aluminium

fres = np.array([180.156115, 179.506483, 178.792279, 177.814923, 176.694473])
omega0 = 2 * np.pi * fres
omega = np.array([1.023388, 0.997994, 0.589699, 0.857605, 1.359736])
domega = 2 * np.pi * omega

AIF = int_friction(domega, omega0)

print(AIF)
```

```
[0.00327967 0.00320987 0.00190424 0.00278457 0.00444295]
```

[126]:
```
plt.plot(T, AIF, 'o', label='Data', zorder=3, color='#474747')
plt.plot(T, AIF, label='Linear Fit', color='#845B97')
```

```
plt.xlabel('Temperature (°C)')
plt.ylabel('Internal Friction')
plt.title('Aluminium')

plt.minorticks_on()
plt.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
plt.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

plt.legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.show()
```



[127]:
```
fig, ax = plt.subplots(3, 1)

ax[0].plot(T, BIF, 'o', label='Data', zorder=3, color='#474747')
ax[0].plot(T, BIF, label='Linear Fit', color='#00B945')
ax[0].minorticks_on()
ax[0].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[0].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[0].set_ylabel('Internal Friction')
```

```
ax[0].set_xlabel('Temperature (°C)')
ax[0].set_title('Brass')

ax[1].plot(T, SSIF, 'o', label='Data', zorder=3, color='#474747')
ax[1].plot(T, SSIF, label='Linear Fit', color='#FF9500')
ax[1].minorticks_on()
ax[1].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[1].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[1].set_ylabel('Internal Friction')
ax[1].set_xlabel('Temperature (°C)')
ax[1].set_title('Stainless Steel')

ax[2].plot(T, AIF, 'o', label='Data', zorder=3, color='#474747')
ax[2].plot(T, AIF, label='Linear Fit', color='#845B97')
ax[2].minorticks_on()
ax[2].grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax[2].grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)
ax[2].set_ylabel('Internal Friction')
ax[2].set_xlabel('Temperature (°C)')
ax[2].set_title('Aluminium')

plt.tight_layout()
plt.show()
```