ElectronicsLab

October 27, 2025

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

import scienceplots
```

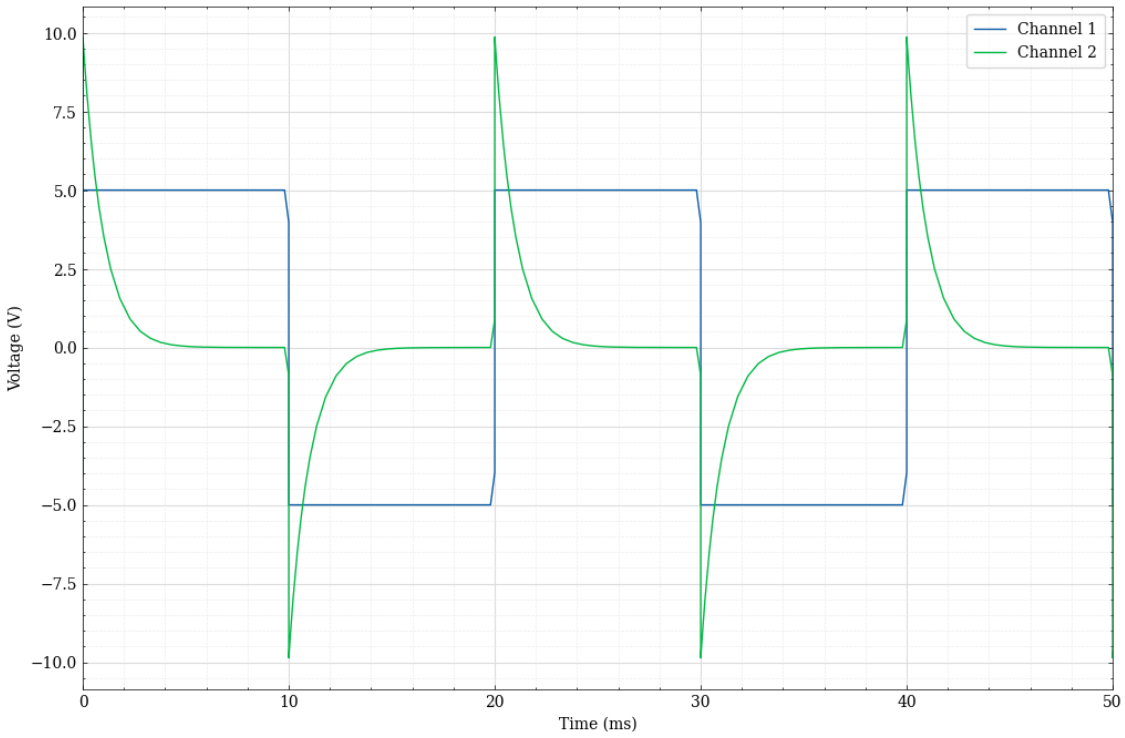
[2]:
```
plt.style.use('science')

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'
])
```

1 Ex 5

[3]:
```
ex5 = np.loadtxt("/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex5.txt", skiprows=1)

ex5_x = ex5[:,0]
ex5_ch1 = ex5[:,1]
ex5_ch2 = ex5[:,2]
```

[4]:
```
fig, ax = plt.subplots()

ax.plot(ex5_x * 1e3, ex5_ch1, label='Channel 1')
plt.xlim([0,50])

ax.plot(ex5_x * 1e3, ex5_ch2, label='Channel 2')

plt.xlabel('Time (ms)')
plt.ylabel('Voltage (V)')

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

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

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

plt.show()
```

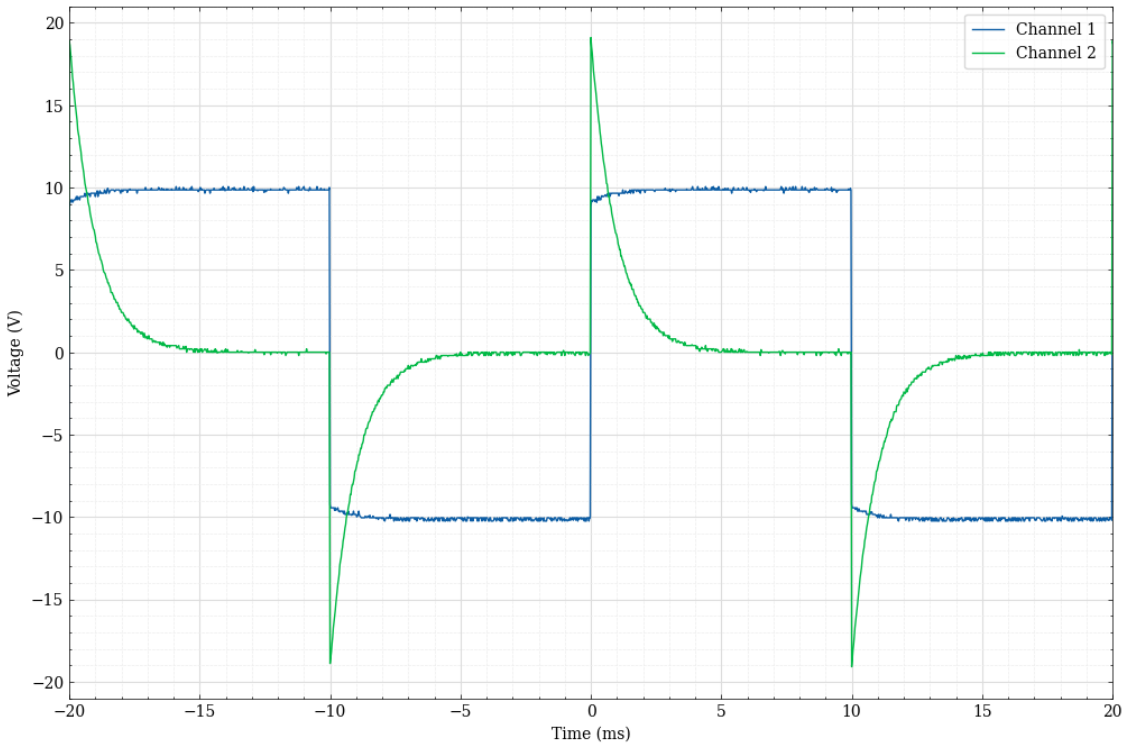


[5]:
```
ex5 = np.loadtxt("/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/exp5_osl2.txt", skiprows=2)

ex5_x_osc = ex5[:,0]
ex5_ch1_osc = ex5[:,1]
ex5_ch2_osc = ex5[:,2]
```

[6]:
```
"""
# Add markers to see where indices are
for i in range(0, len(ex5_x_osc), 100):
    plt.text(ex5_x_osc[i]*1e3, ex5_ch2_osc[i], str(i), fontsize=8)
"""
```

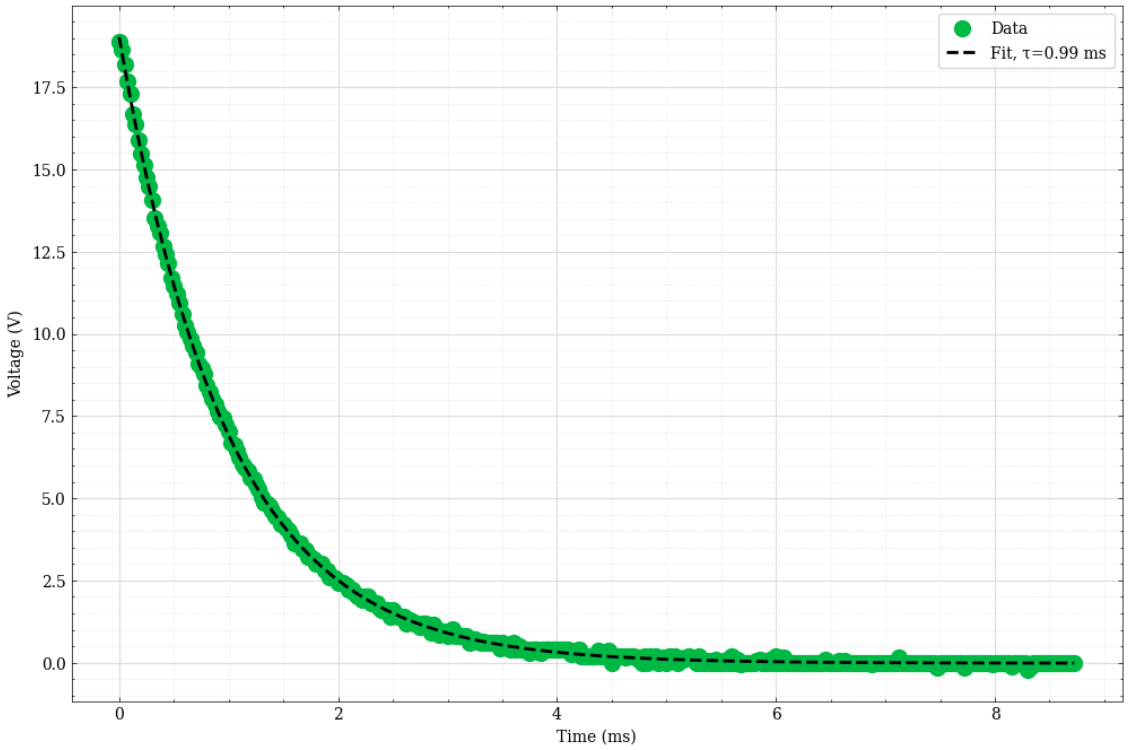
[6]:
```
'\n# Add markers to see where indices are\nfor i in range(0, len(ex5_x_osc),
100):\n    plt.text(ex5_x_osc[i]*1e3, ex5_ch2_osc[i], str(i), fontsize=8)\n'
```

[7]:
```
fig, ax = plt.subplots()

ax.plot(ex5_x_osc * 1e3, ex5_ch1_osc, label='Channel 1')
plt.xlim([-20,20])

ax.plot(ex5_x_osc * 1e3, ex5_ch2_osc, label='Channel 2')

plt.xlabel('Time (ms)')
plt.ylabel('Voltage (V)')

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

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

plt.show()
```



[8]:
```
t = ex5_x_osc.copy()
ch1 = ex5_ch1_osc.copy()
ch2 = ex5_ch2_osc.copy()
```

[9]:
```
def exp_model(t, Vinf, A, tau):
    return Vinf - A*np.exp(-t/tau) # Want decay to Vinf, therefore negative sign

start, end = 200, 550   # adjust indices
t_fit = t[start:end] - t[start]
v_fit = ch2[start:end]

p0 = (np.mean(v_fit[-50:]), v_fit[0] - np.mean(v_fit[-50:]), 0.01)
popt, pcov = curve_fit(exp_model, t_fit, v_fit, p0=p0)
Vinf, A, tau = popt

plt.plot(t_fit * 1e3, v_fit, 'o', label='Data', color='#00B945', markersize=10)
plt.plot(t_fit * 1e3, exp_model(t_fit, *popt), '--', label=f'Fit,  ={tau * 1e3:.
 ↪2f} ms', color='black', linewidth=2)

plt.xlabel('Time (ms)')
plt.ylabel('Voltage (V)')

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"Decay constant   = {tau:.4e}s  ({tau*1e3:.2f} ms)")
```




```
Decay constant   = 9.8894e-04s  (0.99 ms)
```

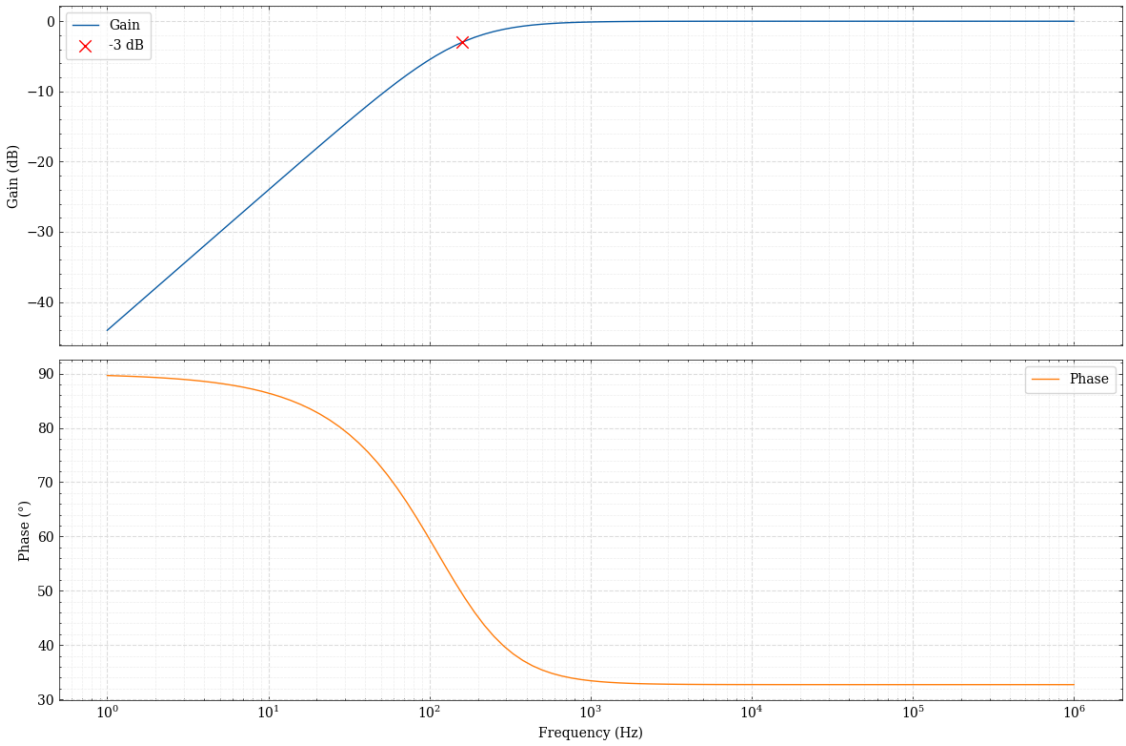
2 Ex 7

[10]:
```
ex7 = np.loadtxt("/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex7.txt", skiprows=1)

ex7_freq = ex7[:,0]
ex7_gain = ex7[:,1]
ex7_phase = ex7[:,1]

ex7_gain = 20 * np.log10(ex7_gain)
ex7_phase = 90 - np.degrees(ex7_phase)
```

[11]:
```
def find_cutoff(f, mag, target):
    # Convert to numpy
    f = np.array(f)
    mag = np.array(mag)

    # Find indices where the curve crosses target
    idx = np.where(np.diff(np.sign(mag - target)))[0]
```

```
    if len(idx) == 0:
        return None  # No crossing found

    i = idx[0]
    # Linear interpolation between the two points around the crossing
    f1, f2 = f[i], f[i+1]
    m1, m2 = mag[i], mag[i+1]

    return f1 + (target - m1) * (f2 - f1) / (m2 - m1)

cutoff = find_cutoff(ex7_freq, ex7_gain, -3)
print("Cutoff frequency  ", cutoff, "Hz")

fig, ax = plt.subplots(2, 1, sharex=True)

ax[0].semilogx(ex7_freq, ex7_gain, label='Gain')
ax[0].scatter(cutoff, -3, color='red', s=80, marker='x', zorder=4, label='-3␣
 ↪dB')
ax[0].set_ylabel('Gain (dB)')
ax[0].grid(which='both', linestyle='--', linewidth=0.5)

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

ax[1].semilogx(ex7_freq, ex7_phase, label='Phase', color='tab:orange')
ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Phase (°)')
ax[1].grid(which='both', linestyle='--', linewidth=0.5)

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

plt.tight_layout()

plt.show()
```

```
Cutoff frequency   160.13176634130096 Hz
```

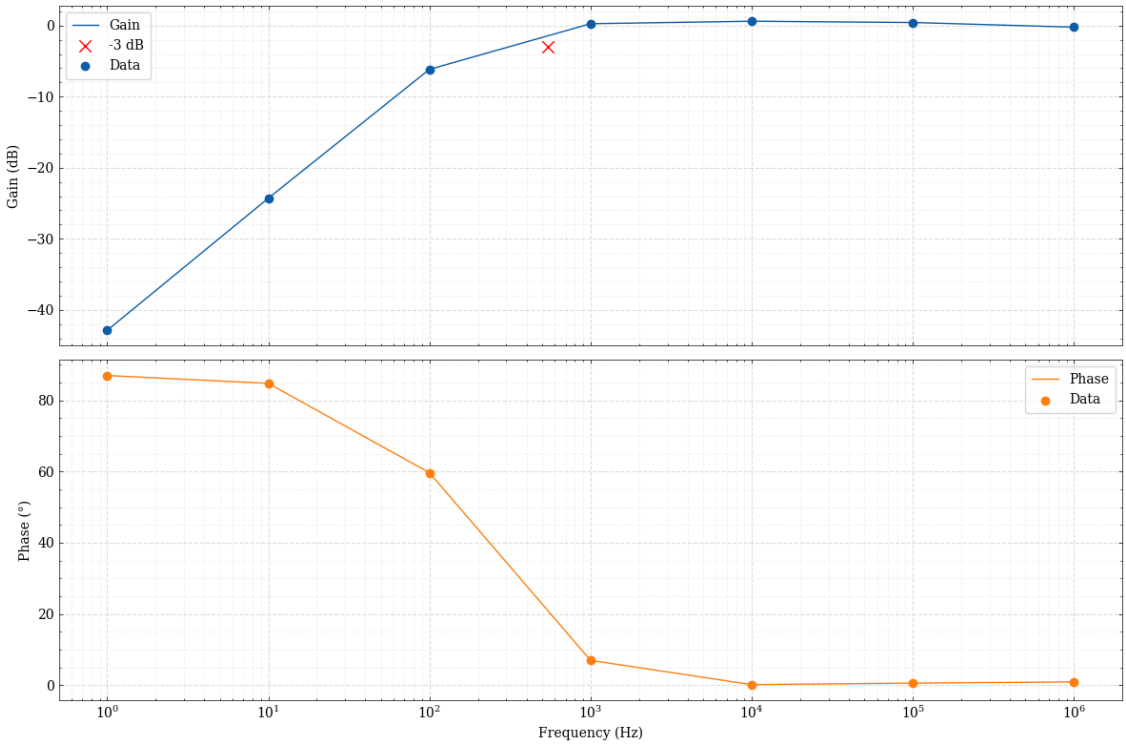


[12]:
```
ex7_f = np.array([1, 10, 100, 1000, 10000, 100000, 1000000])
ex7_Vin = np.array([8.52, 1.01, 1.02, 970e-3, 950e-3, 970e-3, 930e-3])
ex7_Vout = np.array([61.1e-3, 61.6e-3, 502e-3, 1, 1.02, 1.02, 907e-3])
ex7_Phase = np.array([87, 84.8, 59.68, 6.95, 0.14, 0.56, 0.9])

ex7_PPhase = 90 - ex7_Phase

ex7_Gain = 20 * np.log10(ex7_Vout / ex7_Vin)
```

[13]:
```
def find_cutoff(f, mag, target):
    # Convert to numpy
    f = np.array(f)
    mag = np.array(mag)

    # Find indices where the curve crosses target
    idx = np.where(np.diff(np.sign(mag - target)))[0]

    if len(idx) == 0:
        return None  # No crossing found

    i = idx[0]
    # Linear interpolation between the two points around the crossing
    f1, f2 = f[i], f[i+1]
```

```
    m1, m2 = mag[i], mag[i+1]

    return f1 + (target - m1) * (f2 - f1) / (m2 - m1)

cutoff = find_cutoff(ex7_f, ex7_Gain, -3)
print("Cutoff frequency  ", cutoff, "Hz")

fig, ax = plt.subplots(2, 1, sharex=True)

ax[0].semilogx(ex7_f, ex7_Gain, label='Gain')
ax[0].scatter(cutoff, -3, color='red', s=80, marker='x', zorder=4, label='-3␣
 ↪dB')
ax[0].scatter(ex7_f, ex7_Gain, zorder=3, label='Data')
ax[0].set_ylabel('Gain (dB)')
ax[0].grid(which='both', linestyle='--', linewidth=0.5)

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

ax[1].semilogx(ex7_f, ex7_Phase, label='Phase', color='tab:orange')
ax[1].scatter(ex7_f, ex7_Phase, label='Data', color='tab:orange', zorder=3)
ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Phase (°)')
ax[1].grid(which='both', linestyle='--', linewidth=0.5)

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

plt.tight_layout()

plt.show()
```

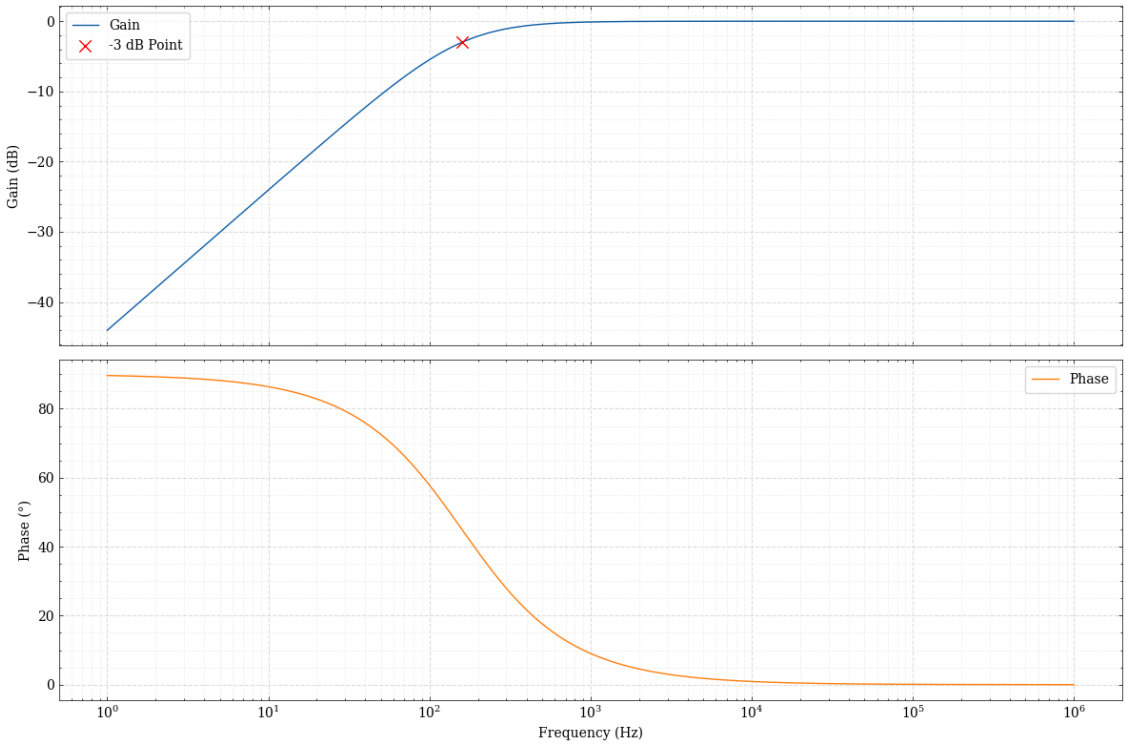
```
Cutoff frequency   542.5284014261945 Hz
```



[14]:
```
V0 = 1
C = 1e-6
R = 1e3

f = np.logspace(0, 6, 1000) # 1 Hz to 1 MHz log-scale, 1000 points between
omega = 2 * np.pi * f

H = 1j * omega * R * C / (1 + 1j * omega * R * C)

mag = np.abs(H)
mag = 20 * np.log10(mag)
phase = np.angle(H, deg=True)

def find_cutoff(f, mag, target):
    # Convert to numpy
    f = np.array(f)
    mag = np.array(mag)

    # Find indices where the curve crosses target
    idx = np.where(np.diff(np.sign(mag - target)))[0]

    if len(idx) == 0:
        return None  # No crossing found
```

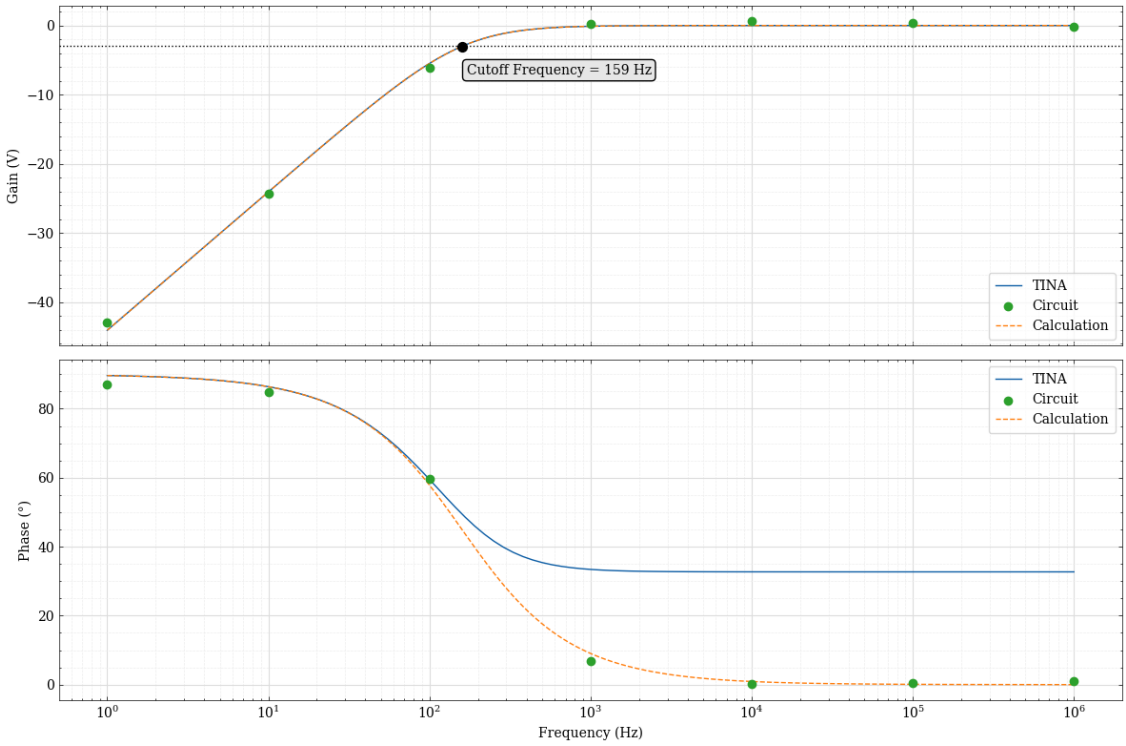
```
    i = idx[0]
    # Linear interpolation between the two points around the crossing
    f1, f2 = f[i], f[i+1]
    m1, m2 = mag[i], mag[i+1]

    return f1 + (target - m1) * (f2 - f1) / (m2 - m1)

cutoff = find_cutoff(f, mag, -3)
print("Cutoff frequency  ", cutoff, "Hz")

fig, ax = plt.subplots(2, 1, sharex=True)

ax[0].semilogx(f, mag, label='Gain')
ax[0].scatter(cutoff, -3, color='red', s=80, marker='x', zorder=4, label='-3 dB␣
 ↪Point')
ax[0].set_ylabel('Gain (dB)')
ax[0].grid(which='both', linestyle='--', linewidth=0.5)

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

ax[1].semilogx(f, phase, label='Phase', color='tab:orange')
ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Phase (°)')
ax[1].grid(which='both', linestyle='--', linewidth=0.5)

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

plt.tight_layout()

plt.show()
```

```
Cutoff frequency   159.53860495032134 Hz
```



[15]:
```
gain_3db = np.interp(159, f, mag)

fig, ax = plt.subplots(2, 1, sharex=True)

ax[0].semilogx(ex7_freq, ex7_gain, label='TINA')
ax[0].scatter(ex7_f, ex7_Gain, label='Circuit', color='tab:green', marker='o',␣
 ↪zorder=3)
ax[0].scatter(159, gain_3db, color='black', s=50, zorder=4)
ax[0].annotate('Cutoff Frequency = 159 Hz', xy=(159, gain_3db), xytext=(170,␣
 ↪gain_3db-4), bbox=dict(fc='0.9', boxstyle='round'))
ax[0].axhline(y=-3, linestyle=':', color='black')
ax[0].semilogx(f, mag, label='Calculation', color='tab:orange', linestyle='--')

ax[0].set_ylabel('Gain (V)')
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].legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)


ax[1].semilogx(ex7_freq, ex7_phase, label='TINA')
```

```
ax[1].scatter(ex7_f, ex7_Phase, label='Circuit', color='tab:green', marker='o',␣
 ↪zorder=3)
ax[1].semilogx(f, phase, label='Calculation', color='tab:orange',␣
 ↪linestyle='--')

ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Phase (°)')
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].legend(frameon=True, fancybox=True, framealpha=0.75, borderpad=0.5)

plt.tight_layout()

plt.show()
```



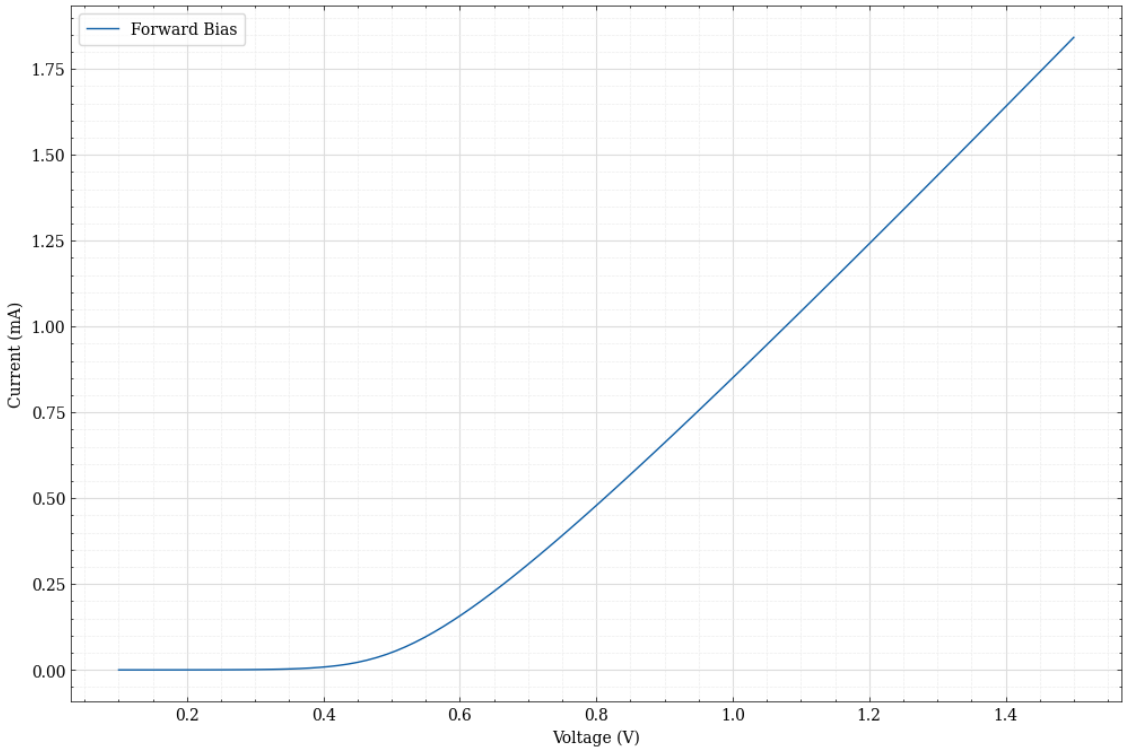
3 Ex 12

3.1 a.

3.1.1 Theory

[16]:
```
ex12a = np.loadtxt("/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex12a.txt", skiprows=1)

V_for = ex12a[:,0]
I_for = ex12a[:,1]
```

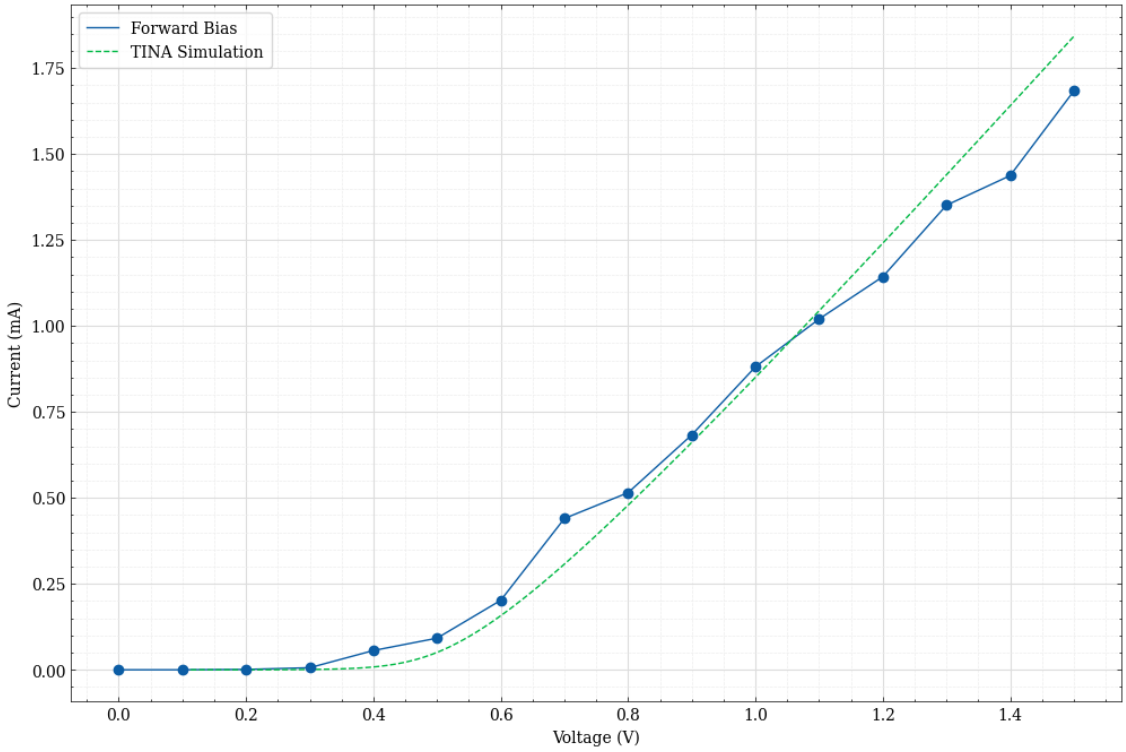
[17]:
```
plt.plot(V_for , I_for *1e3, label='Forward Bias')

plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")

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()
```



3.1.2 Ammeter

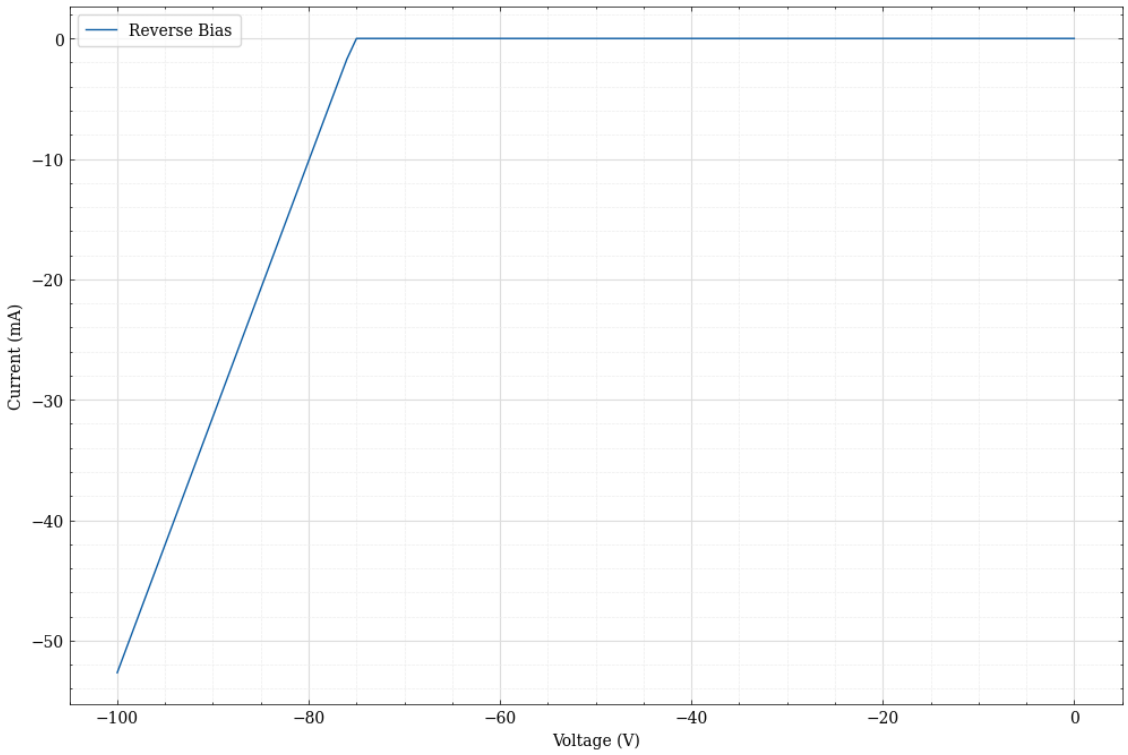
[18]:
```
ex12_for_am_V = np.array([0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1,1.2,1.
↪3,1.4,1.5])
ex12_for_am_I = np.array([0,0,0.001,0.006,0.056,0.092,0.202,0.440,0.515,0.683,0.
↪881,1.020,1.143,1.351,1.437,1.683])
```

[19]:
```
plt.plot(ex12_for_am_V, ex12_for_am_I, label='Forward Bias')
plt.scatter(ex12_for_am_V, ex12_for_am_I, color='#0C5DA5', zorder=3)

plt.plot(V_for , I_for *1e3, label='TINA Simulation', linestyle='--')

plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")

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()
```



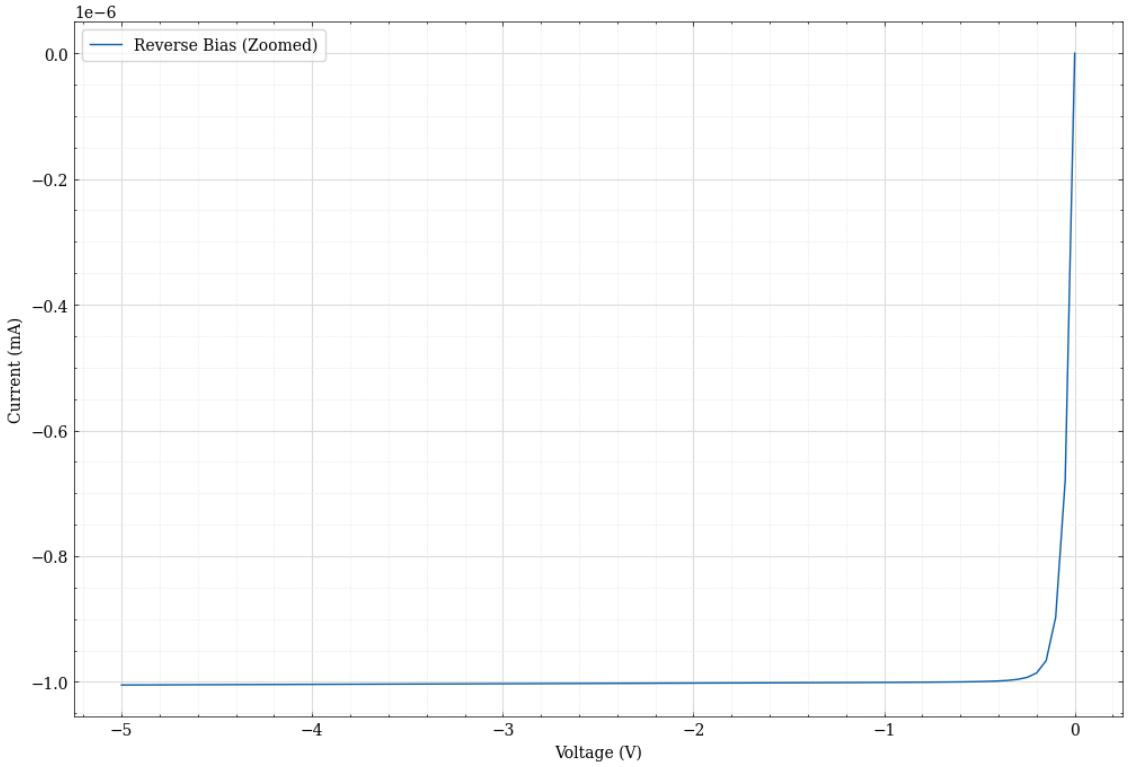
[20]:
```
ex12a_rev = np.loadtxt("/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex12a_rev.txt", skiprows=1)

V_rev = ex12a_rev[:,1]
I_rev = ex12a_rev[:,2]
```

[21]:
```
plt.plot(V_rev , I_rev * 1e3, label='Reverse Bias')

plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")

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()
```



[22]:
```
ex12a_rev_zoom = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex12a_rev_zoom.txt', skiprows=1)

V_rev_zoom = ex12a_rev_zoom[:,1]
I_rev_zoom = ex12a_rev_zoom[:,2]
```

[23]:
```
plt.plot(V_rev_zoom , I_rev_zoom * 1e3, label='Reverse Bias (Zoomed)')

plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")

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()
```

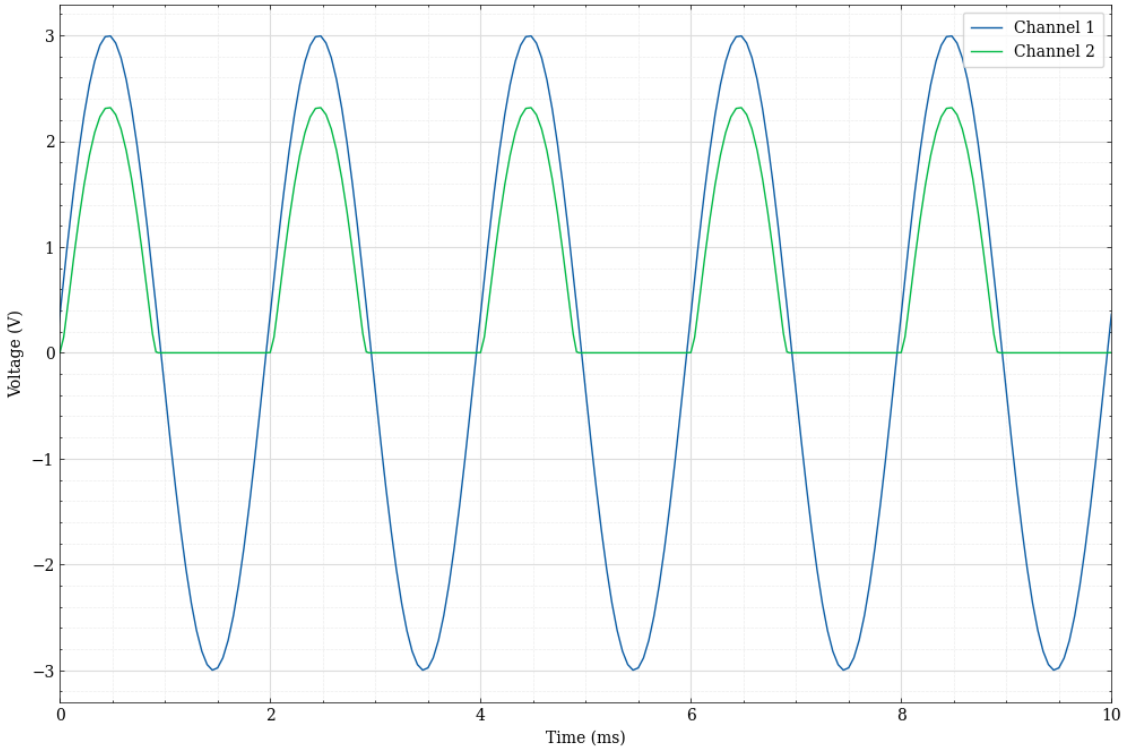


3.2 b.

3.2.1 Theory

[24]:
```
ex12b_nocap = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/exp12b.txt', skiprows=1)

ex12b_nocap_time = ex12b_nocap[:,0]
ex12b_nocap_ch1 = ex12b_nocap[:,1]
ex12b_nocap_ch2 = ex12b_nocap[:,2]
```

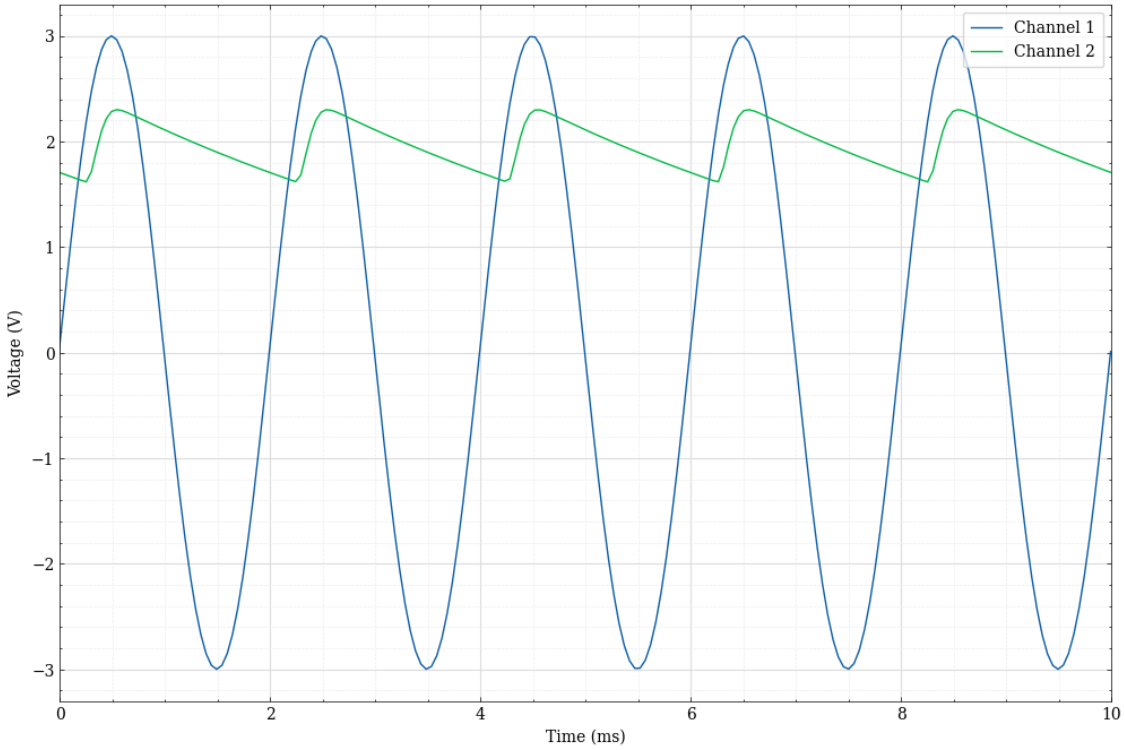
[25]:
```
fig, ax = plt.subplots()

ax.plot(ex12b_nocap_time * 1e3, ex12b_nocap_ch1, label='Channel 1')
plt.xlim([0,10])

ax.plot(ex12b_nocap_time * 1e3, ex12b_nocap_ch2, label='Channel 2')

plt.xlabel('Time (ms)')
plt.ylabel('Voltage (V)')

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

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

plt.show()
```



[26]:
```
ex12b_cap = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
↪labcode_s3/exp12b_cap.txt', skiprows=1)

ex12b_cap_time = ex12b_cap[:,0]
ex12b_cap_ch1 = ex12b_cap[:,1]
ex12b_cap_ch2 = ex12b_cap[:,2]
```

[57]:
```
fig, ax = plt.subplots()

ax.plot(ex12b_cap_time * 1e3, ex12b_cap_ch1, label='Channel 1')
plt.xlim([0,10])

ax.plot(ex12b_cap_time * 1e3, ex12b_cap_ch2, label='Channel 2')

plt.xlabel('Time (ms)')
plt.ylabel('Voltage (V)')

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

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

plt.show()
```

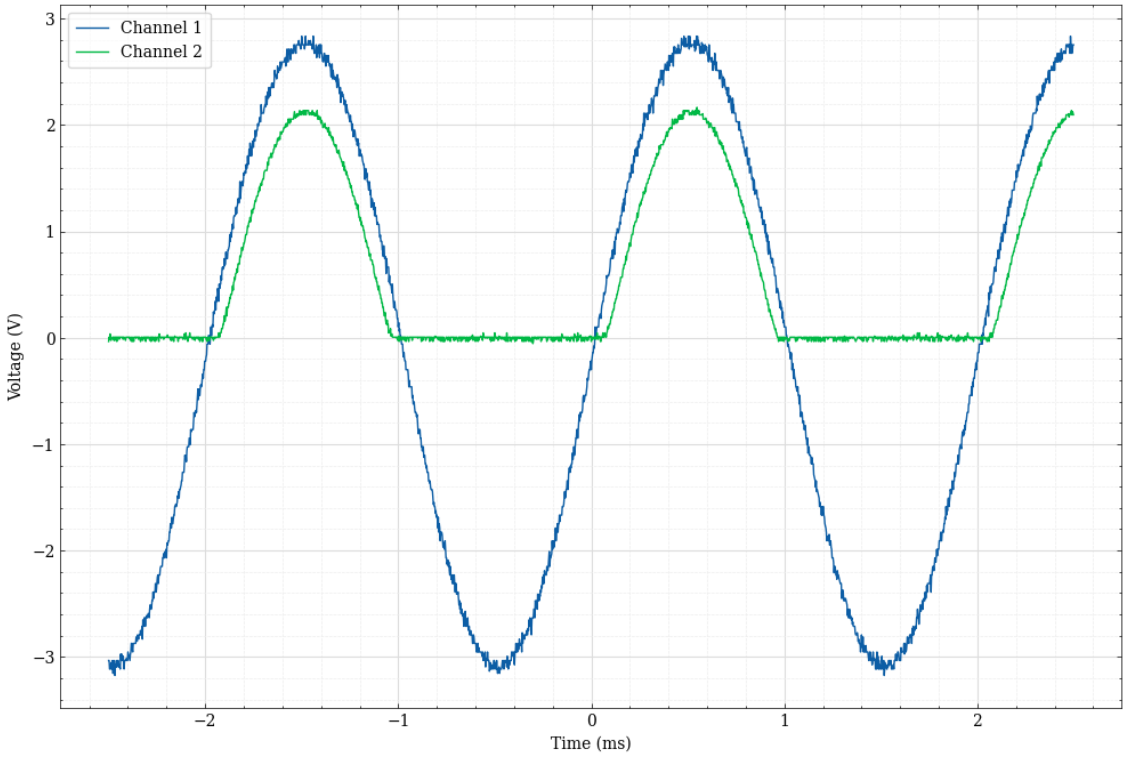


3.2.2 Oscilloscope

[28]:
```
ex12b_nocap_osc = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
↪labcode_s3/osc_ex12b.txt', skiprows=1)

ex12b_nocap_time_osc = ex12b_nocap_osc[:,0]
ex12b_nocap_ch1_osc = ex12b_nocap_osc[:,1]
ex12b_nocap_ch2_osc = ex12b_nocap_osc[:,2]
```

[50]:
```
fig, ax = plt.subplots()

ax.plot(ex12b_nocap_time_osc * 1e3, ex12b_nocap_ch1_osc, label='Channel 1')

ax.plot(ex12b_nocap_time_osc * 1e3, ex12b_nocap_ch2_osc, label='Channel 2')

plt.xlabel('Time (ms)')
plt.ylabel('Voltage (V)')
```

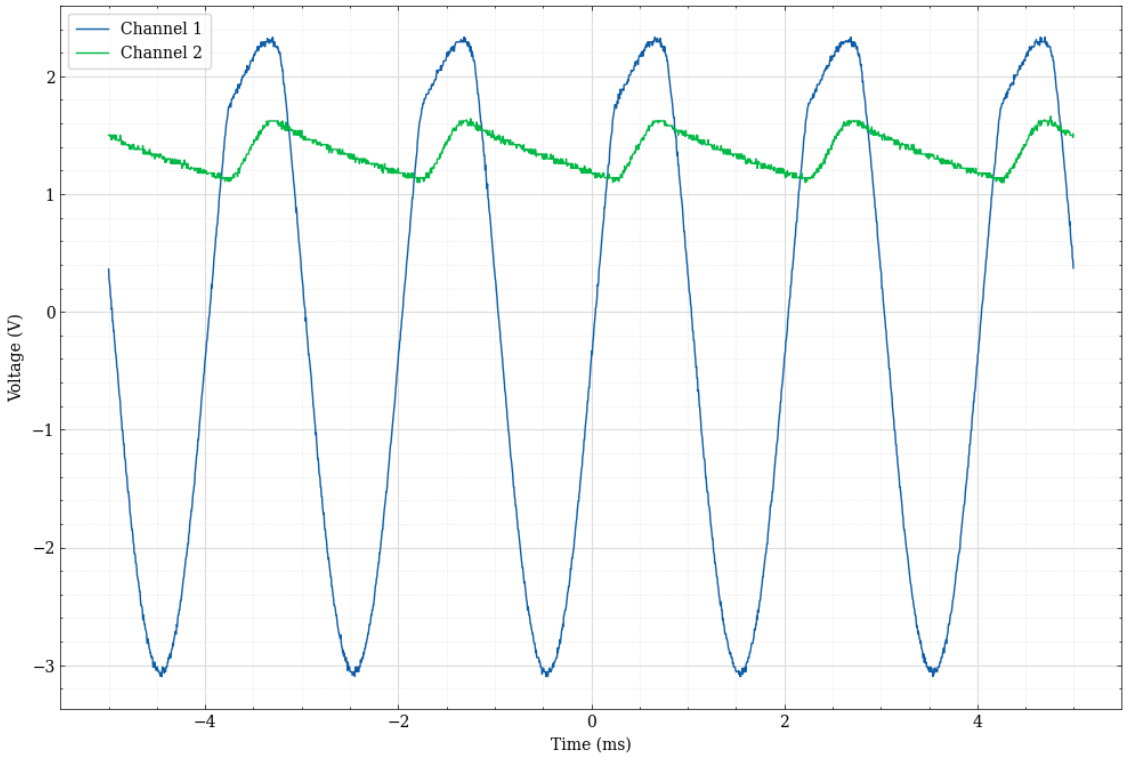
```
ax.minorticks_on()
ax.grid(True, which="major", linewidth=0.8, color="#DDDDDD", zorder=2)
ax.grid(True, which="minor", linewidth=0.5, color="#EEEEEE", linestyle="--",␣
 ↪zorder=1)

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

plt.show()
```



[59]:
```
ex12b_cap_osc = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/osc_ex12b_cap.txt', skiprows=1)

ex12b_cap_time_osc = ex12b_cap_osc[:,0]
ex12b_cap_ch1_osc = ex12b_cap_osc[:,1]
ex12b_cap_ch2_osc = ex12b_cap_osc[:,2]
```

[58]:
```
fig, ax = plt.subplots()

ax.plot(ex12b_cap_time_osc * 1e3, ex12b_cap_ch1_osc, label='Channel 1')

ax.plot(ex12b_cap_time_osc * 1e3, ex12b_cap_ch2_osc, label='Channel 2')
```

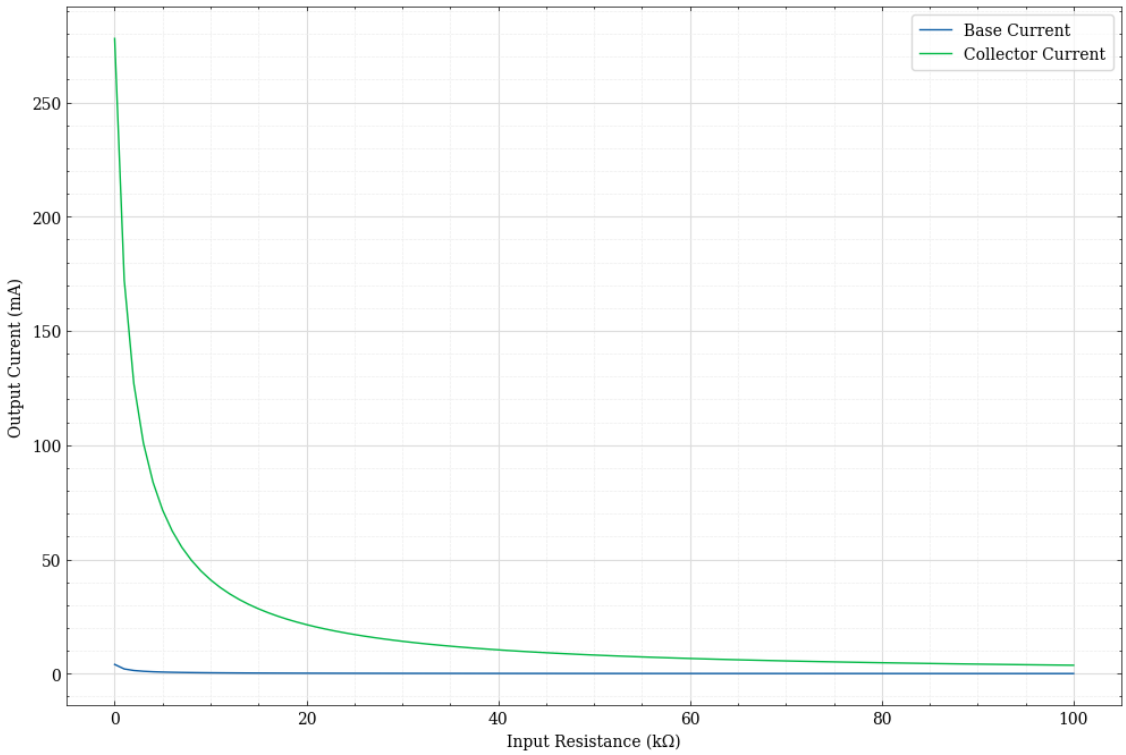
```
plt.xlabel('Time (ms)')
plt.ylabel('Voltage (V)')

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

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

plt.show()
```



4 Ex 19

[32]:
```
ex19 = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex19.txt', skiprows=1)

input = ex19[:,0]
am1 = ex19[:,1]
am2 = ex19[:,2]
```

[63]:
```
fig, ax = plt.subplots()

ax.plot(input * 1e-3, am1 * 1e3, label='Base Current')

ax.plot(input * 1e-3, am2 * 1e3, label='Collector Current')

plt.xlabel('Input Resistance (kΩ)')
plt.ylabel('Output Curent (mA)')

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

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

plt.show()

hfe = np.average(am2 / am1)
print(hfe)
```

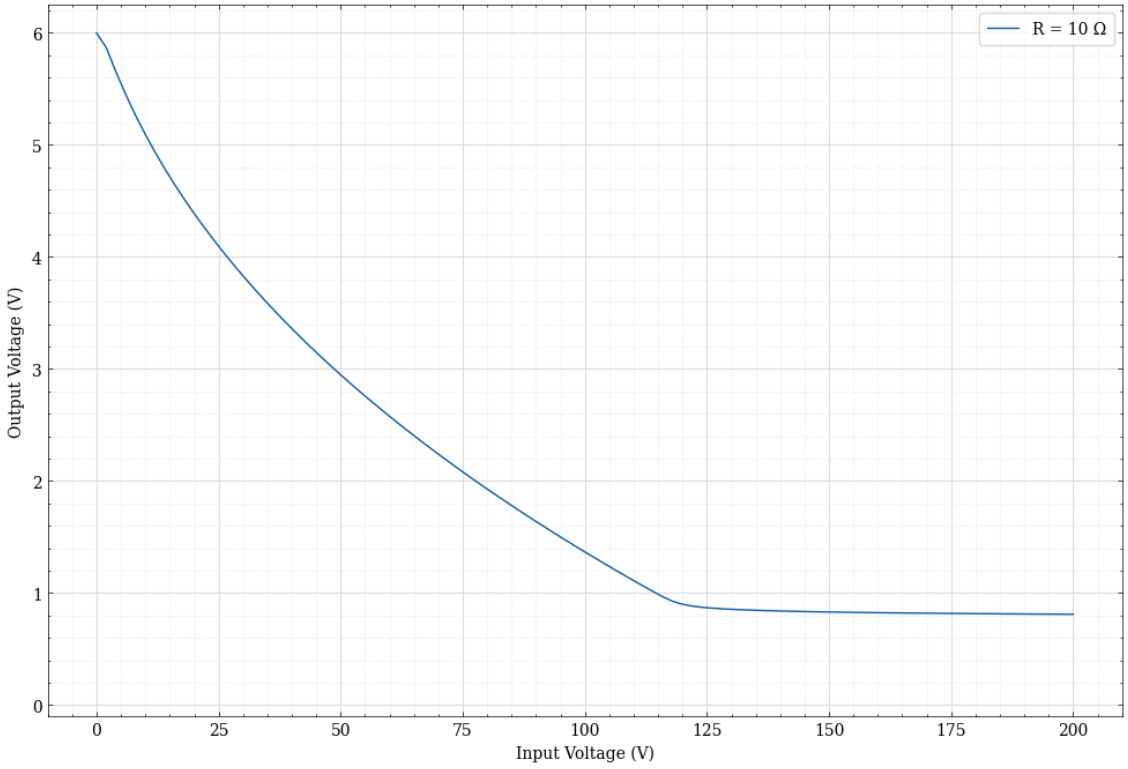



```
96.21463033275784
```

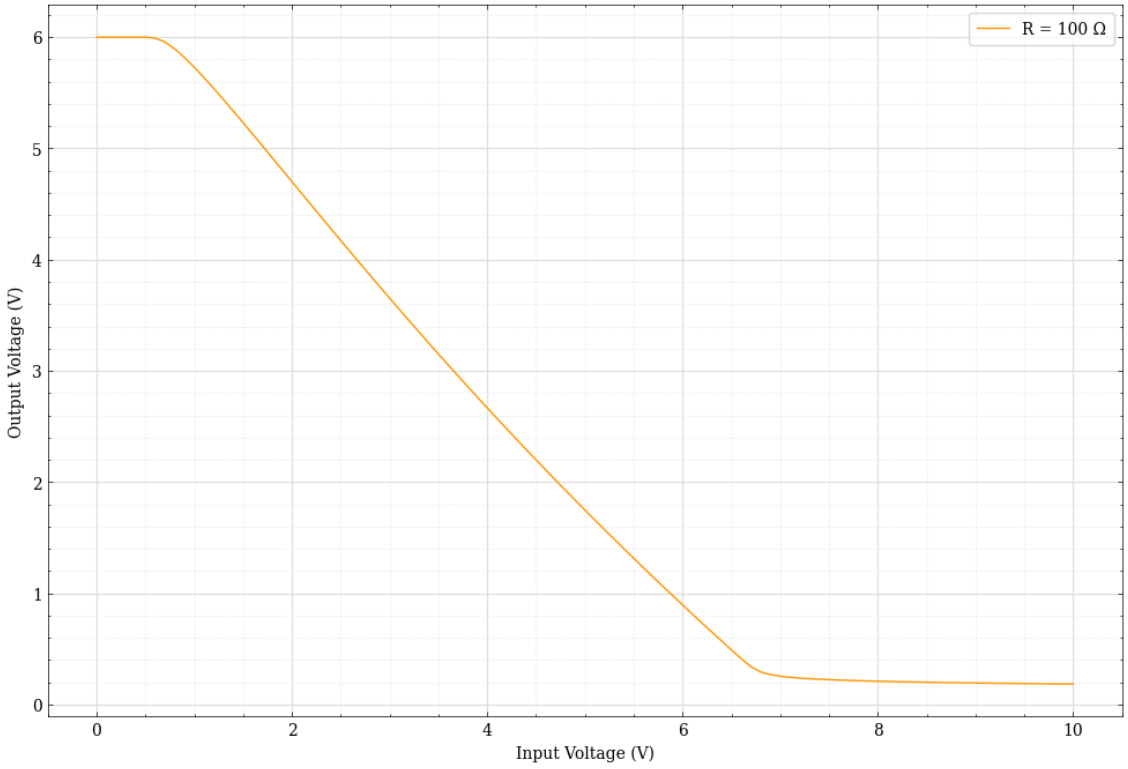
5 Ex 20

[34]:
```
ex20_r10 = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex20_r10.txt', skiprows=1)
ex20_r100 = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex20_r100.txt', skiprows=1)
ex20_r1k = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex20_r1000.txt', skiprows=1)
ex20_r10k = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex20_r10000.txt', skiprows=1)

ex20_input_r10 = ex20_r10[:,0]
ex20_output_r10 = ex20_r10[:,1]
ex20_input_r100 = ex20_r100[:,0]
ex20_output_r100 = ex20_r100[:,1]
ex20_input_r1k = ex20_r1k[:,0]
ex20_output_r1k = ex20_r1k[:,1]
ex20_input_r10k = ex20_r10k[:,0]
ex20_output_r10k = ex20_r10k[:,1]
```

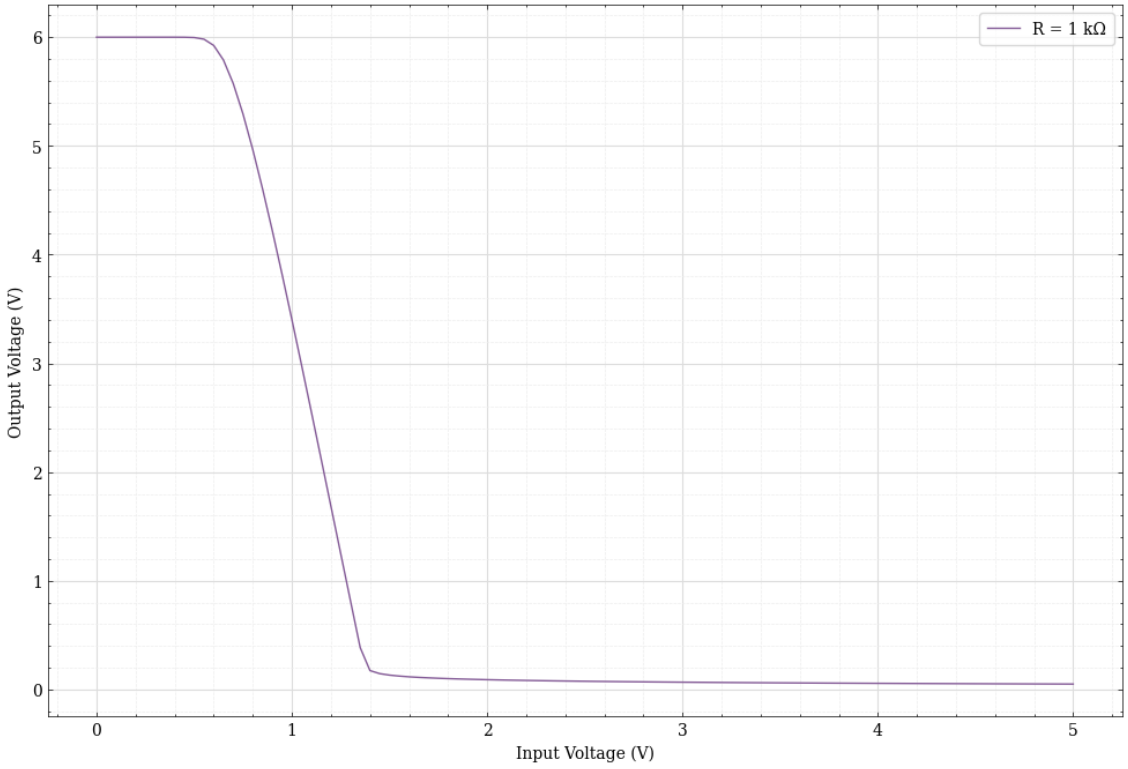
[64]:
```
plt.plot(ex20_input_r10, ex20_output_r10, label='R = 10 Ω')

plt.xlabel('Input Voltage (V)')
plt.ylabel('Output Voltage (V)')

plt.ylim([-0.1,6.25])

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()
```



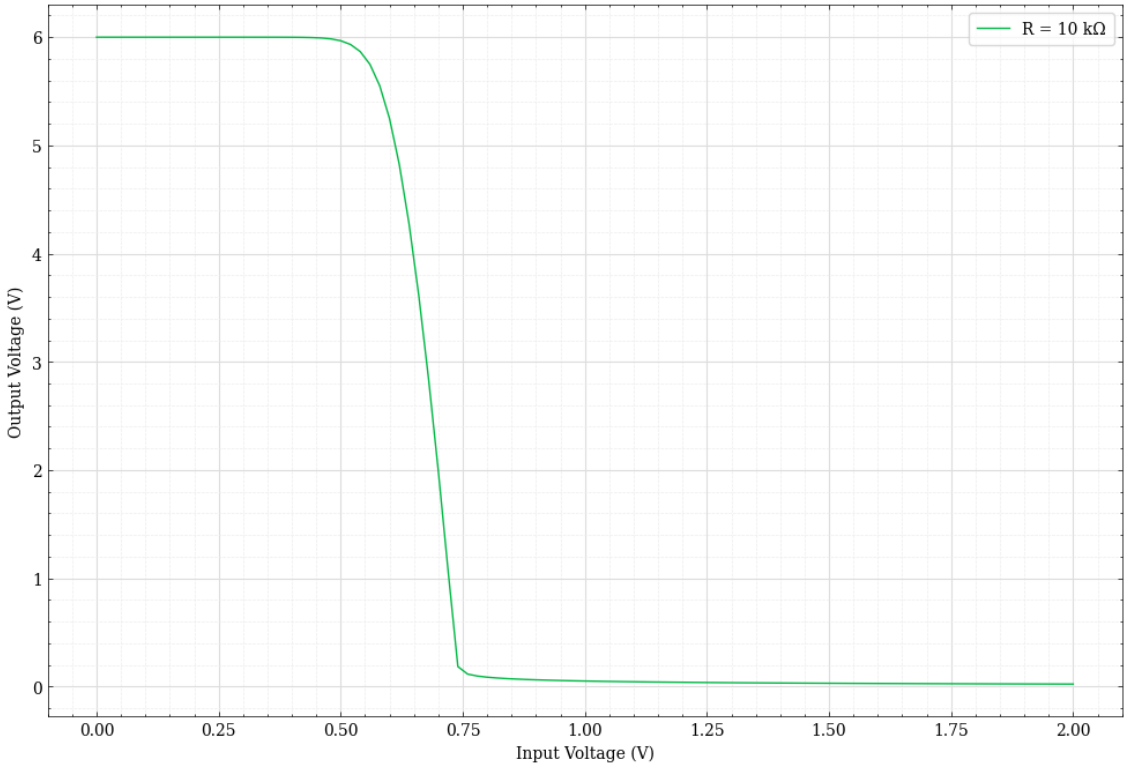
[66]:
```
plt.plot(ex20_input_r100, ex20_output_r100, color='#FF9500', label='R = 100 Ω')

plt.xlabel('Input Voltage (V)')
plt.ylabel('Output Voltage (V)')

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()
```



[67]:
```
plt.plot(ex20_input_r1k, ex20_output_r1k, color='#845B97', label='R = 1 kΩ')

plt.xlabel('Input Voltage (V)')
plt.ylabel('Output Voltage (V)')

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()
```



[68]:
```
plt.plot(ex20_input_r10k, ex20_output_r10k, color='#00B945', label='R = 10 kΩ')

plt.xlabel('Input Voltage (V)')
plt.ylabel('Output Voltage (V)')

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()
```

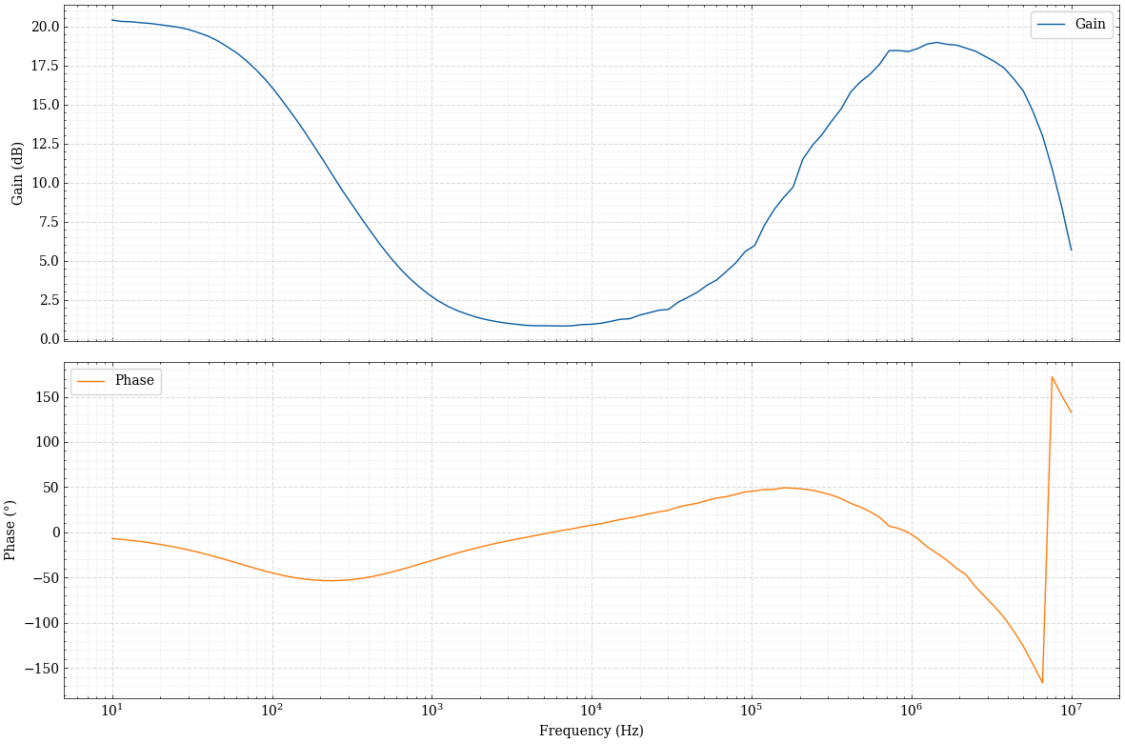


6 Ex 25

[39]:
```
ex25_cap = np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex25_cap.txt', skiprows=1)

ex25_cap_f = ex25_cap[:,0]
ex25_cap_amp = ex25_cap[:,1]
ex25_cap_gain = ex25_cap[:,2]
ex25_cap_phase = ex25_cap[:,3]
```

[40]:
```
fig, ax = plt.subplots(2, 1, sharex=True)

ax[0].semilogx(ex25_cap_f, ex25_cap_gain, label='Gain')
ax[0].set_ylabel('Gain (dB)')
ax[0].grid(which='both', linestyle='--', linewidth=0.5)

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

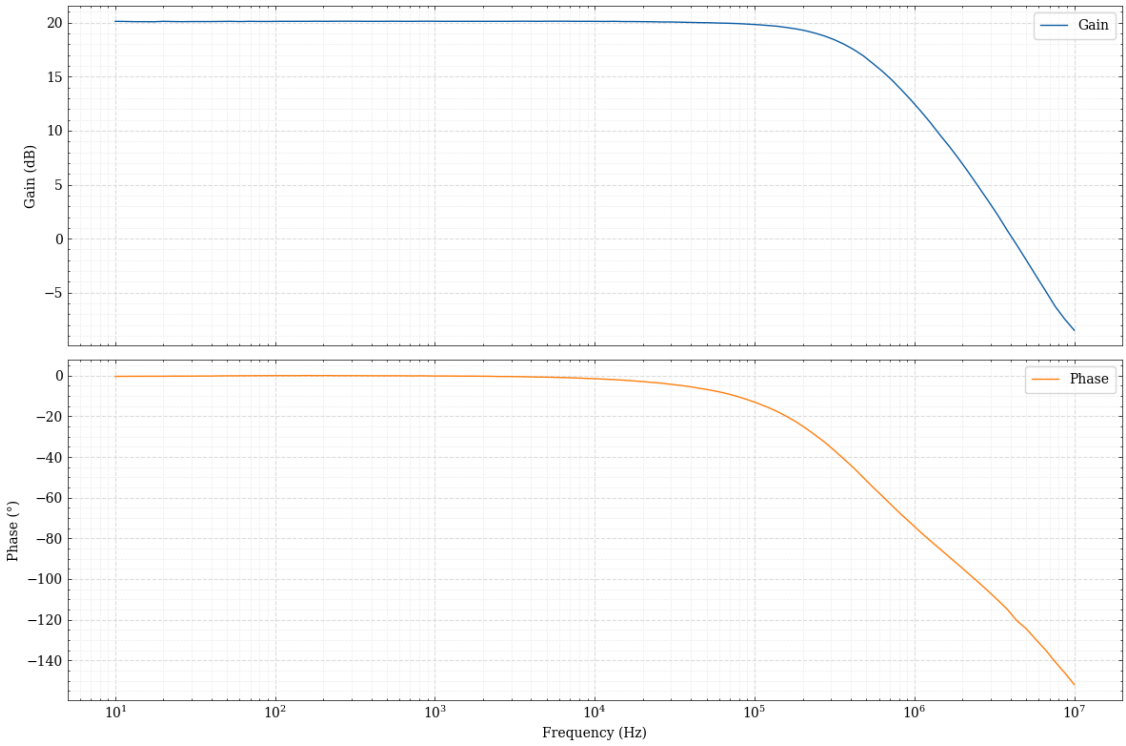
```
ax[1].semilogx(ex25_cap_f, ex25_cap_phase, label='Phase', color='tab:orange')
ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Phase (°)')
ax[1].grid(which='both', linestyle='--', linewidth=0.5)

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

plt.tight_layout()

plt.show()
```



[42]:
```
ex25_nocap =np.loadtxt('/Users/JoanaUCD/Library/CloudStorage/
 ↪OneDrive-UniversityCollegeDublin/Labs/labs-files/labs_code/Labs-Code/
 ↪labcode_s3/ex25_nocap.txt', skiprows=1)

ex25_nocap_f = ex25_nocap[:,0]
ex25_nocap_amp = ex25_nocap[:,1]
ex25_nocap_gain = ex25_nocap[:,2]
```

```
ex25_nocap_phase = ex25_nocap[:,3]
```

[43]:
```
fig, ax = plt.subplots(2, 1, sharex=True)

ax[0].semilogx(ex25_nocap_f, ex25_nocap_gain, label='Gain')
ax[0].set_ylabel('Gain (dB)')
ax[0].grid(which='both', linestyle='--', linewidth=0.5)

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

ax[1].plot(ex25_nocap_f, ex25_nocap_phase, label='Phase', color='tab:orange')
ax[1].set_xlabel('Frequency (Hz)')
ax[1].set_ylabel('Phase (°)')
ax[1].grid(which='both', linestyle='--', linewidth=0.5)

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

plt.tight_layout()

plt.show()
```


Gain
Gain (dB)
20
15
10
5
0
−5
Phase
Phase (°)
0
−20
−40
−60
−80
−100
−120
−140
10^1
10^2
10^3
10^4
10^5
10^6
10^7
Frequency (Hz)