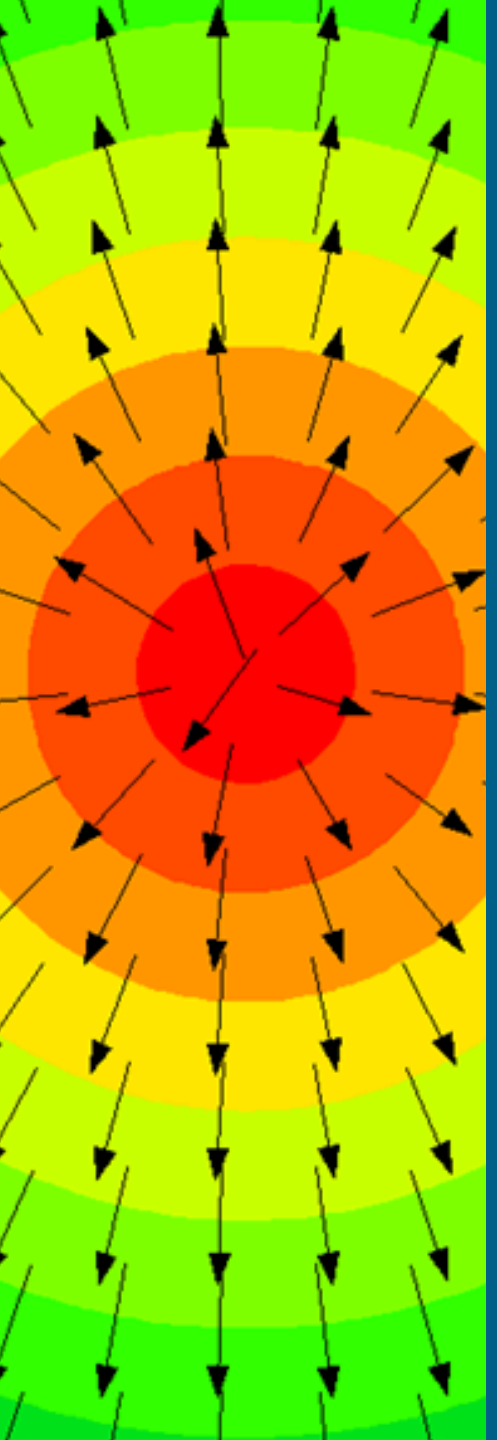
MARCH 28TH, 2025

Plotting Continued

CE 311K - L30



Customizing Figures

Axes **limits** can be modified using *xlim()* and *ylim()*

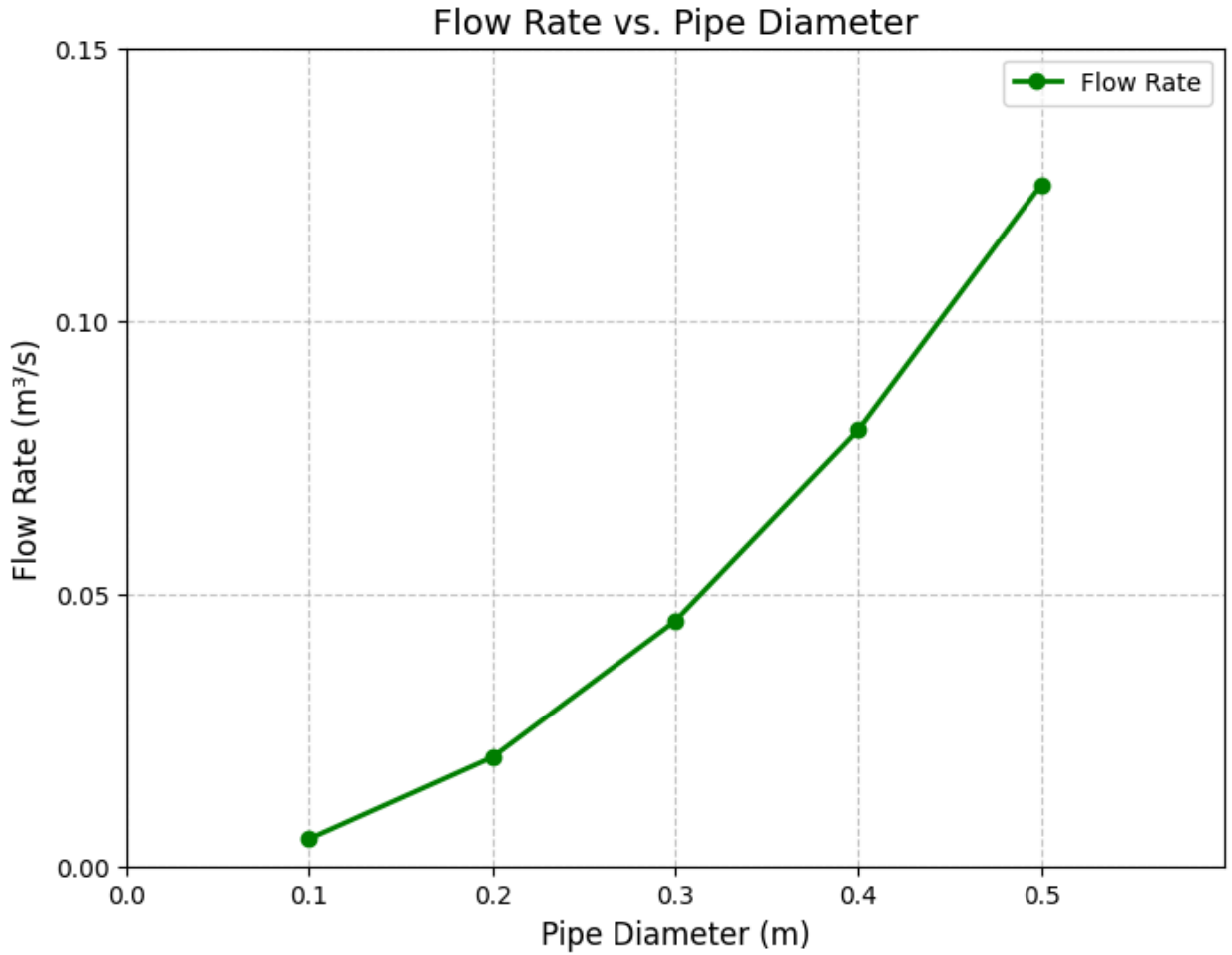
```
plt.plot([0, 1, 2, 3], [0, 10, 20, 30])
plt.xlim(1, 3)  # Limit x-axis from 1 to 3
plt.ylim(10, 30)  # Limit y-axis from 10 to 30
```

Axes **ticks** can be modified with *xticks()* and *yticks()*

```
plt.plot([1, 2, 3], [4, 5, 6])
plt.xticks([1, 2, 3], ['Low', 'Medium', 'High'])  # Custom x-axis ticks
plt.yticks([4, 5, 6], ['Slow', 'Moderate', 'Fast'])  # Custom y-axis ticks
```

Customizing Figures

```
diameter = [0.1, 0.2, 0.3, 0.4, 0.5] # m
flow_rate = [0.005, 0.02, 0.045, 0.08, 0.125]  # m3/s

plt.figure(figsize=(8, 6))
plt.plot(diameter, flow_rate, marker='o',
color='green', label="Flow Rate", linewidth=2)

plt.xlim(0, 0.6)  # Focus on pipe diameter range
plt.ylim(0, 0.15)  # Highlight flow rate range
plt.xticks([0, 0.1, 0.2, 0.3, 0.4, 0.5])
plt.yticks([0, 0.05, 0.1, 0.15])

plt.xlabel("Pipe Diameter (m)", fontsize=12)
plt.ylabel("Flow Rate (m³/s)", fontsize=12)
plt.title("Flow Rate vs. Pipe Diameter", fontsize=14)

# Add grid and legend
plt.grid(which='both', linestyle='--', alpha=0.7)
plt.legend()
```



Subplots

plt.subplots() is the most versatile way to create subplots. It returns:

A *figure* object (top-level container)

A set of *axes* objects (individual subplots)

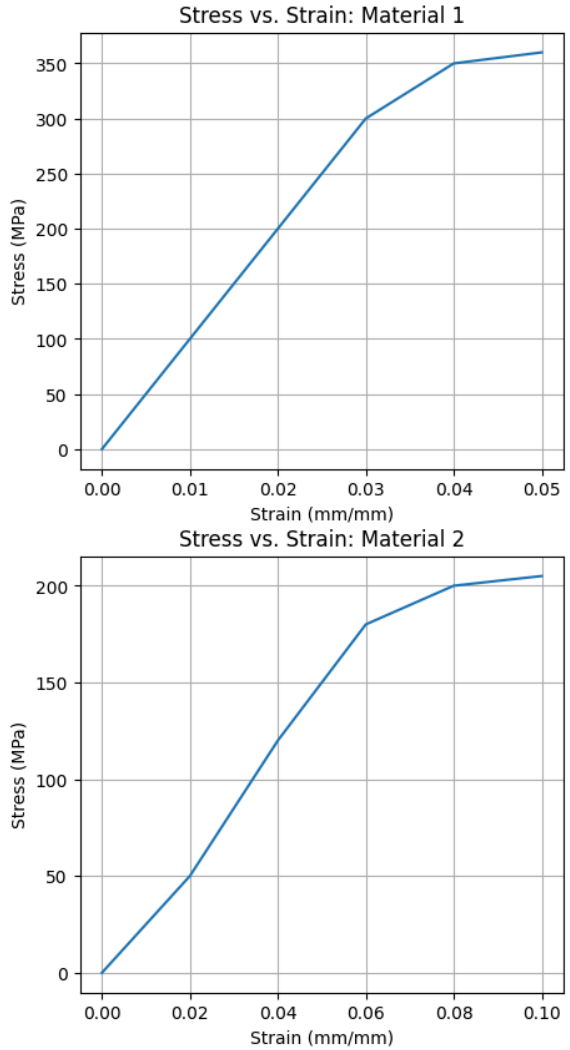
```
fig, axs = plt.subplots(nrows, ncols, figsize=(width, height))
```

nrows, *ncols*: Number of rows and columns of subplots

figsize: Size of the entire figure in inches

Subplots

```
strain_1 = [0, 0.01, 0.02, 0.03, 0.04, 0.05]
stress_1 = [0, 100, 200, 300, 350, 360]
strain_2 = [0, 0.02, 0.04, 0.06, 0.08, 0.1]
stress_2 = [0, 50, 120, 180, 200, 205]

fig, axs = plt.subplots(2, 1, figsize=(5, 10))

axs[0].plot(strain_1, stress_1, label='Material 1')
axs[0].set_title("Stress vs. Strain: Material 1")
axs[0].set_xlabel("Strain (mm/mm)")
axs[0].set_ylabel("Stress (MPa)")
axs[0].grid(True)

axs[1].plot(strain_2, stress_2, label='Material 2')
axs[1].set_title("Stress vs. Strain: Material 2")
axs[1].set_xlabel("Strain (mm/mm)")
axs[1].set_ylabel("Stress (MPa)")
axs[1].grid(True)
```



Twin Axes

Axes.twinx() allow you to overlay two plots with different y-axes but a shared x-axis

Useful for comparing datasets with different scales

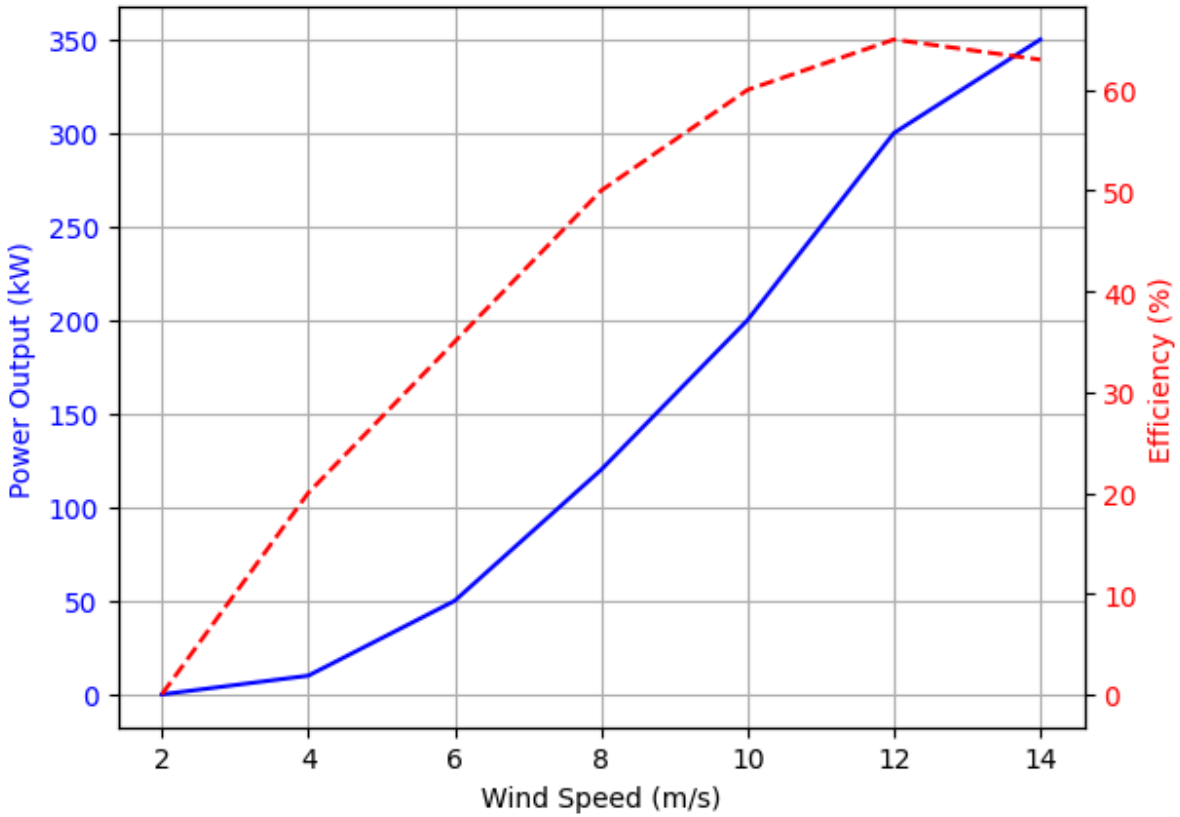
```
ax2 = ax1.twinx()
```

*twinx() c*reates a second y-axis on the right-hand side of the plot

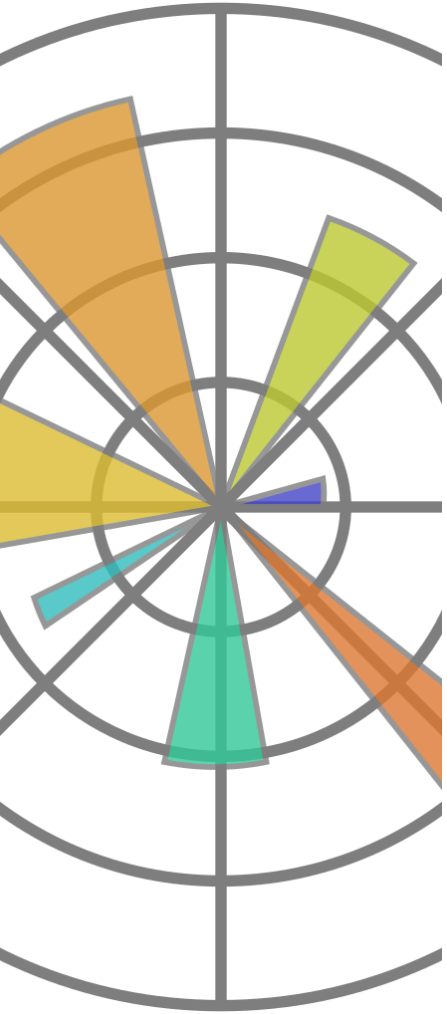
Each y-axis can have its own scale, label, and color

Twin Axes

```
wind_speed = [2, 4, 6, 8, 10, 12, 14]  # m/s
power_output = [0, 10, 50, 120, 200, 300, 350]  # kW
efficiency = [0, 20, 35, 50, 60, 65, 63] # %

fig, ax1 = plt.subplots()

ax1.plot(wind_speed, power_output, 'b-')
ax1.set_xlabel("Wind Speed (m/s)")
ax1.set_ylabel("Power Output (kW)", color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
ax1.grid(True)

ax2 = ax1.twinx()
ax2.plot(wind_speed, efficiency, 'r--',
label='Efficiency')
ax2.set_ylabel("Efficiency (%)", color='red')
ax2.tick_params(axis='y', labelcolor='red')
```



Summary



Limits

Use *xlim()* and *ylim()* to specify the range of your axes

Ticks

Customize the location and label of the tick marks with *xticks()* and *yticks()*

plt.subplots()

Returns *fig* and *axes* handles to work with one or multiple plot

twinx()

Create two plots together with the same x-axis

y-axes can be modified separately for varying scales and format