CSC 223 - Advanced Scientific Programming

Matplotlib

Matplotlib

- Matplotlib is a visualization library built on Numpy arrays
- Convention for importing Matplotlib

```
import matplotlib as mpl
import matplotlib.pyplot as plt

plt.style.use('classic')
```

- Matplotlib was originally written as a Python alternative for MATLAB and has two interfaces:
  - A MATLAB style interface (pyplot)
  - An object oriented interface (Figure, Axes)

Displaying Plots

- Plotting from a script

```
plt.show()
```

- Plotting from an IPython shell

```
%matplotlib
import matplotlib.pyplot as plt

# use plt.draw() to force an update
```

Matplotlib Script Example

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

plt.show()
```

Matplotlib IPython Example

```
%matplotlib

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
fig = plt.figure() # create a plot window
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
```

MATLAB-style Interface Example

```
plt.figure() # create a plot

# create a first panel and set the axis
plt.subplot(2, 1, 1) # (rows, columns, panel #)
plt.plot(x, np.sin(x))

# create a second panel and set the axis
plt.subplot(2, 1, 2)
plt.plot(x, np.cos(x))
```

Object-oriented Interface Example

```
# create a grid of subplots
fig, ax = plt.subplots(2)
# fig is the figure
# ax is an array of Axes objects

# Call the plot method on each Axes object
ax[0].plot(x, np.sin(x))
ax[1].plot(x, np.cos(x))
```

Basic Plot Customization

- Plot style
- Line colors
- Line styles
- Axes limits
- Labels

Line Color

- The `color` parameter can be used to adjust line color

```
plt.plot(x, y, color='blue') # name
plt.plot(x, y, color='g') # color code
plt.plot(x, y, color='0.75') # grayscale
plt.plot(x, y, color='#FF0000') # hex
plt.plot(x, y, color=(1.0, 0.0, 0.0)) # RBG
plt.plot(x, y, color='papayawhip') # HTML
```

Line Style

- The `linestyle` parameter can be used to adjust line style

```
plt.plot(x, y, linestyle='solid')
plt.plot(x, y, linestyle='dashed')
plt.plot(x, y, linestyle='dashdot')
plt.plot(x, y, linestyle='dotted')

# OR

plt.plot(x, y, linestyle='-')
plt.plot(x, y, linestyle='--')
plt.plot(x, y, linestyle='-.')
plt.plot(x, y, linestyle=':')
```

Axes Limits

- The limits of the axes can be set in various ways:

```
# configure each axis individually
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)

# all at once
plt.axis([xmin, xmax, ymin, ymax])

# automatically compute limits
plt.axis('tight')

# make axis units equal
plt.axis('equal')
```

Labels

- The figure can have various labels

```
# add a title
plt.title("My Awesome Plot")

# label the axes
plt.xlabel("x")
plt.ylabel("y")

# add a legend
plt.legend()
```

Discrepancies Between Interfaces

- Most of the `plt` functions correspond to `ax` methods, but there are some exceptions:

`plt.xlabel`	$\rightarrow$	`ax.set_xlabel()`
`plt.ylabel`	$\rightarrow$	`ax.set_ylabel()`
`plt.xlim`	$\rightarrow$	`ax.set_xlim()`
`plt.ylim`	$\rightarrow$	`ax.set_ylim()`
`plt.title`	$\rightarrow$	`ax.set_title()`

Scatter Plots

- The `plot` method can accept a marker shape
- Example

```
plt.plot(x, y, 'o', color='black')

# common markers:
# 'o', '.', ',', 'x', '+', 'v',
# '^', '<', '>', 's', 'd'
```

- The `scatter` method is specialized for scatter plots
- Example

```
plt.scatter(x, y, 'o')
```

Another `plot` Example

- The `plot` method accepts a wide range of arguments
- Example

```
plt.plot(x, y, '-p', color='gray',
         linewidth=3,
         markersize=4,
         markerfacecolor=white,
         markeredgecolor='black',
         markeredgewidth=2)
```

Another `scatter` Example

- The `scatter` method has additional features for scatter plots
- Example

```
rng = np.random.RandomState(0)
x = rng.randn(100)
y = rng.randn(100)
colors = rng.rand(100)
sizes = 1000* rng.rand(100)
plt.scatter(x, y, c=colors, s=sizes,
    alpha=0.3, cmap='viridis')
plt.colorbar()
```

Saving Figures to a File

- The `savefig` method can be used to save a figure to a file
- Example

```
fig = plt.figure() # create a plot window
# add some stuff
fig.savefig('my_figure.png')
```

- Supported image formats depend on which backends are installed

```
# list supported file types
fig.canvas.get_supported_filetypes()
```