PWC Python Course: Visualization

Ramses van Zon

SciNet HPC Consortium

December 11, 2014

In this session, we will discuss the following topics:

- Basics of visualization and how to get started.
- How to make your work presentable and professional.
- Advanced plotting techniques.

Matplotlib's pyplot

The most commonly-used Python visualization package is matplotlib.pyplot:

- matplotlib is an add-on package to Python.
- Designed to have look which mimics MATLAB.
- Has all the plotting types you'd expect: line, scatter, bar, pie, contour, polar, box, sankey, *etc.*
- More-advanced functionality includes subplots, inset plots, colourbars, legends, *etc.*
- Control over every aspect of your plot is available, though not necessarily easily or obviously.
- Also has 3D plotting, built-in widgets, animations.

This is the package which we'll be using to produce images today.

Using Ipython with Matplotlib in Eclipse

- Go to `Window`⟶`Preferences`
- On the left, open the `PyDev` section and click on the `Interactive Console`
- One the right, you will see settings for the interactive consoles
- Add the following line to the `Initial interpreter commands`:

```
pylab
```

  (i.e., below the `import sys; print...` line)
- Click OK

Then you can open an interactive prompt (close any existing ones)

- Open any python file, or put the cursor in an existing one.
- Press Ctrl-Alt-Enter
- Select `Python Console`
- Alternatively, find the corresponding GUI icon.

Python vs. Ipython

- This set up the interactive plotting mode using ipython from within eclipse.
- When plotting from a script, things are a bit different:
  - One has to manually import modules, e.g.

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

  - Plots will not immediately be shown on screen. Python/Matplotlib waits to draw the figure until your script needs it.
  - That can involve an uncontrolled combination of the command `plt.show()`, `plt.draw()`, `plt.pause(.1)`, `plt.ion()`, ...
- Safe bet: Use plotting commands in scripts to save figures to files.

SciNet compute • calcul CANADA

Plotting

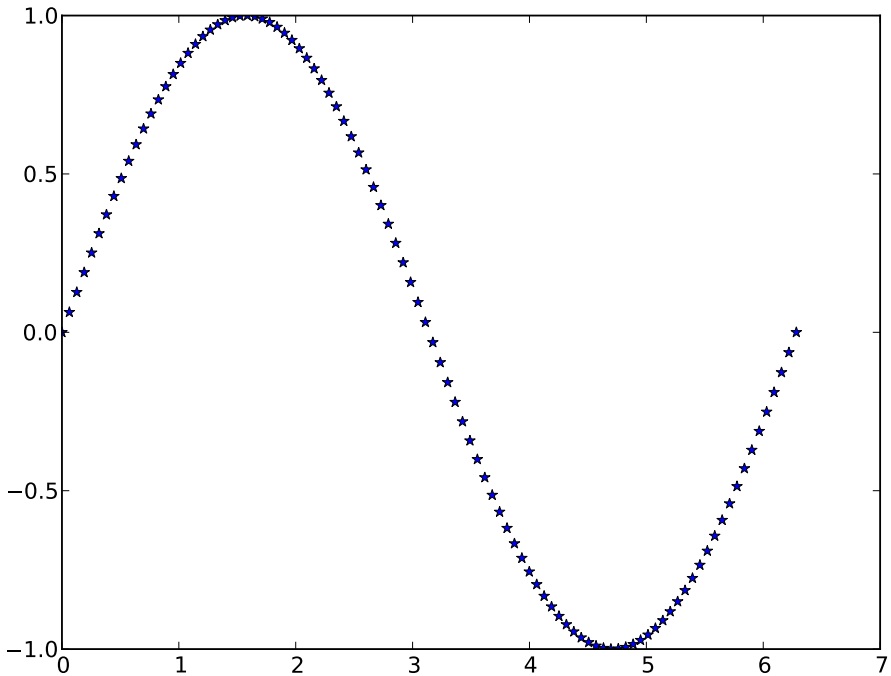
```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>>
```

Plotting

```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>> x=np.linspace(0, 2*np.pi, 100)
>>>
```

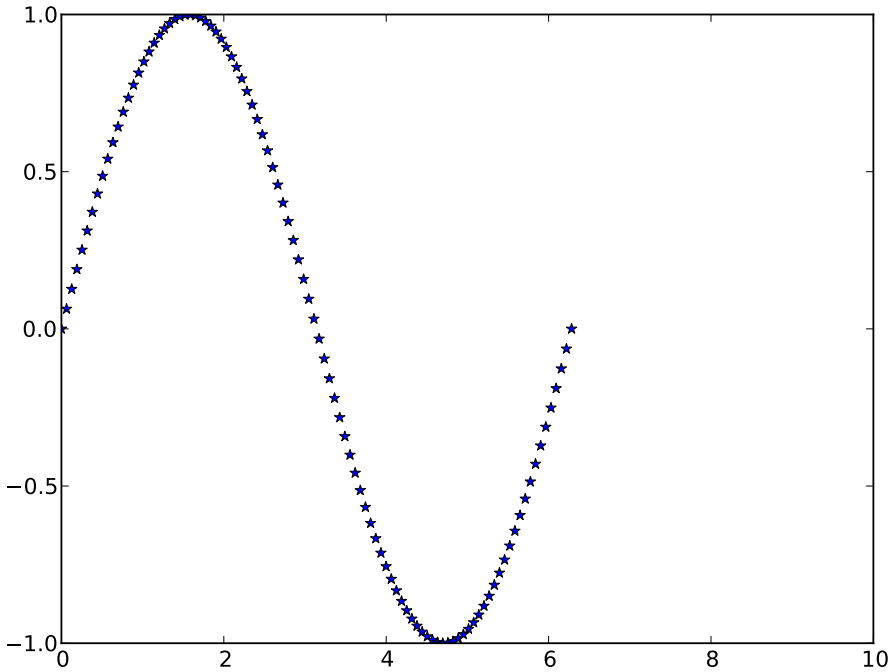
Plotting

```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>> x=np.linspace(0, 2*np.pi, 100)
>>> plt.plot(x, np.sin(x), '*')

>>>
```



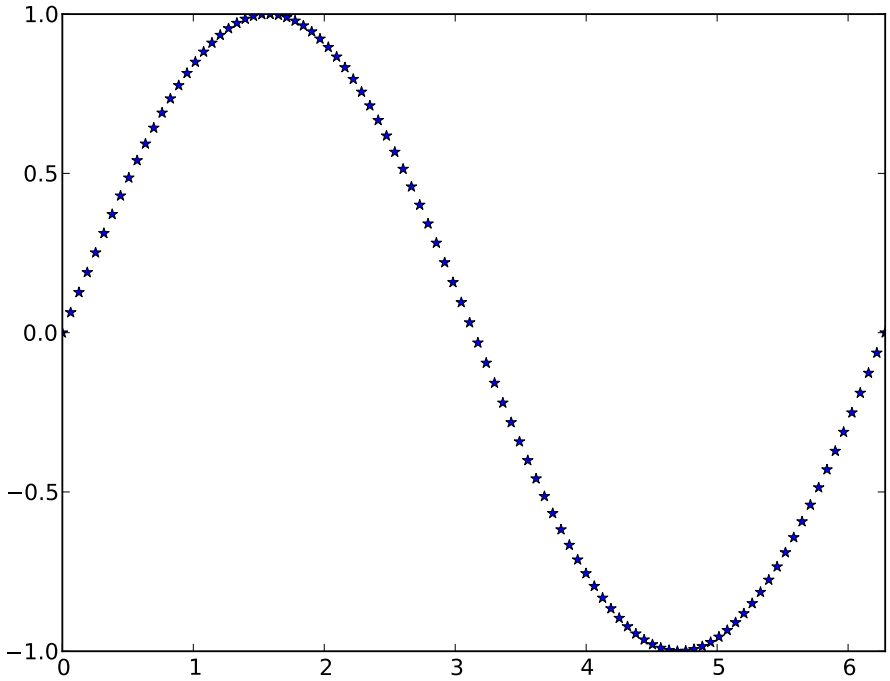
Plotting

```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>> x=np.linspace(0, 2*np.pi, 100)
>>> plt.plot(x, np.sin(x), '*')

>>> plt.xlim(0, 10)
(0, 10)

>>>
```



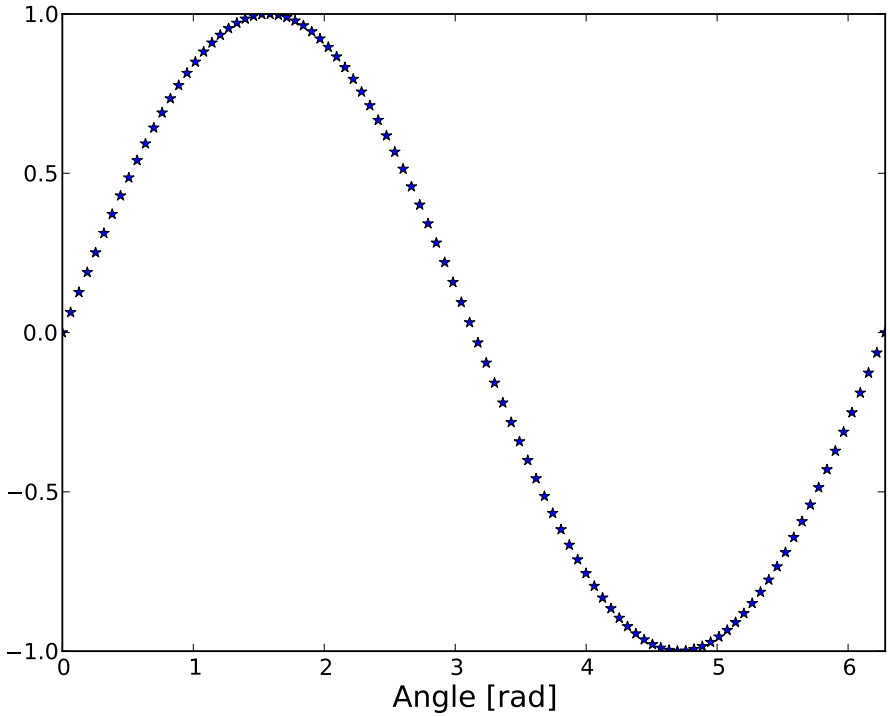
Plotting

```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>> x=np.linspace(0, 2*np.pi, 100)
>>> plt.plot(x, np.sin(x), '*')

>>> plt.xlim(0, 10)
(0, 10)

>>> plt.xlim(0, 2 * np.pi)
(0, 6.2831853071795862)

>>>
```



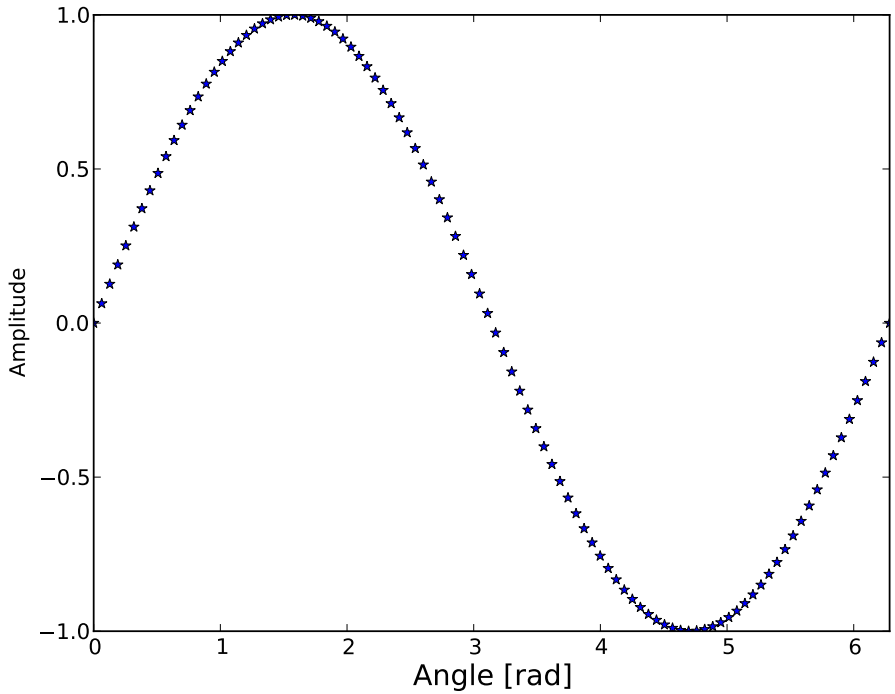
Plotting

```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>> x=np.linspace(0, 2*np.pi, 100)
>>> plt.plot(x, np.sin(x), '*')

>>> plt.xlim(0, 10)
(0, 10)

>>> plt.xlim(0, 2 * np.pi)
(0, 6.2831853071795862)

>>> plt.xlabel('Angle [rad]',
  fontsize = 16)

>>>
```



Plotting

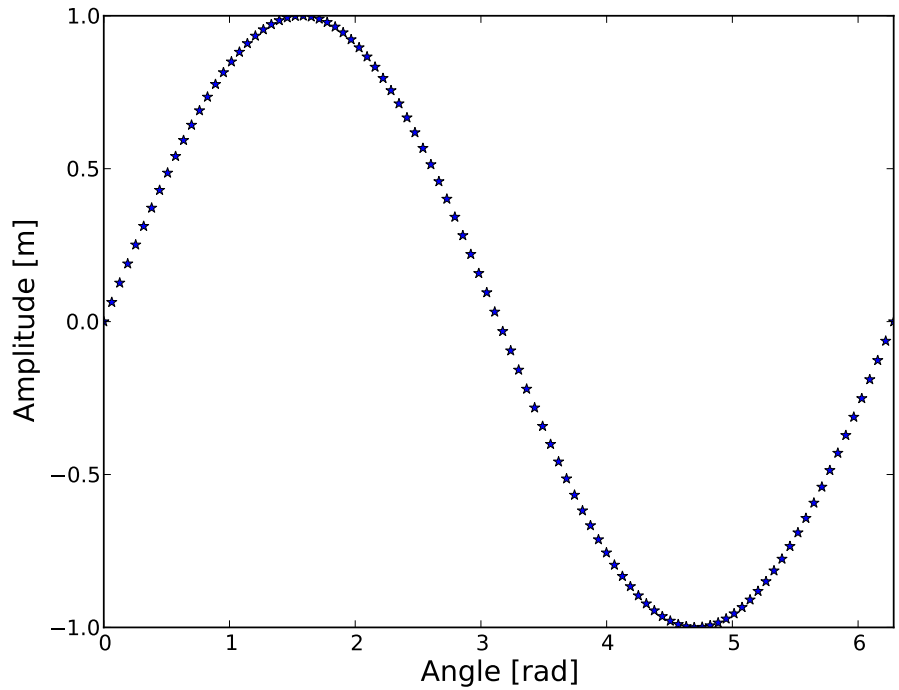
```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>> x=np.linspace(0, 2*np.pi, 100)
>>> plt.plot(x, np.sin(x), '*')

>>> plt.xlim(0, 10)
(0, 10)

>>> plt.xlim(0, 2 * np.pi)
(0, 6.2831853071795862)

>>> plt.xlabel('Angle [rad]',
  fontsize = 16)

>>> plt.ylabel('Amplitude')
```




```
>>>
```

Plotting

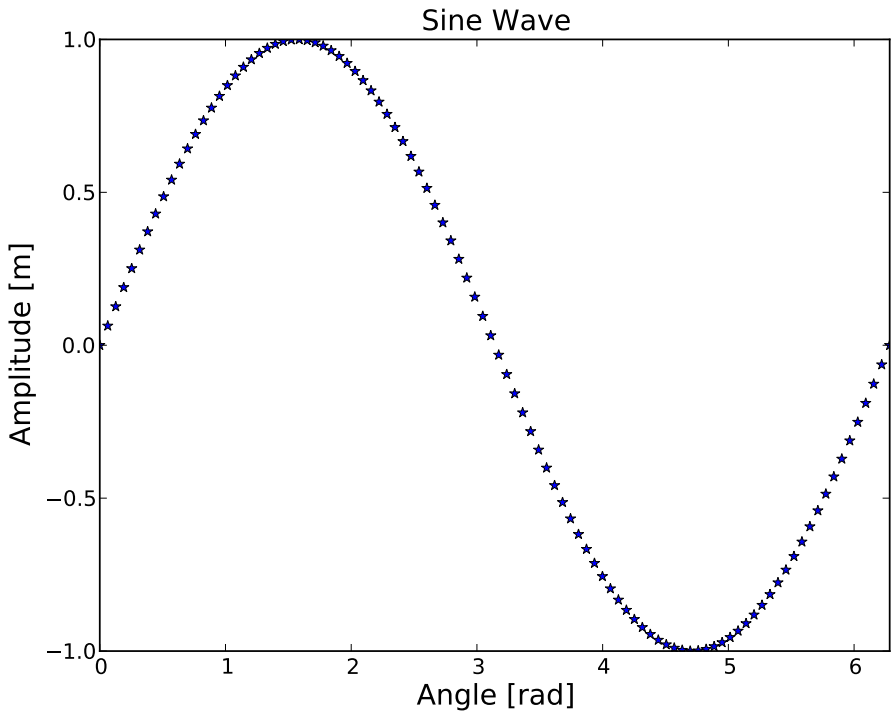
```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>> x=np.linspace(0, 2*np.pi, 100)
>>> plt.plot(x, np.sin(x), '*')

>>> plt.xlim(0, 10)
(0, 10)

>>> plt.xlim(0, 2 * np.pi)
(0, 6.2831853071795862)

>>> plt.xlabel('Angle [rad]',
  fontsize = 16)

>>> plt.ylabel('Amplitude')
```




```
>>> plt.ylabel('Amplitude [m]',
  fontsize = 16)

>>>
```

Plotting

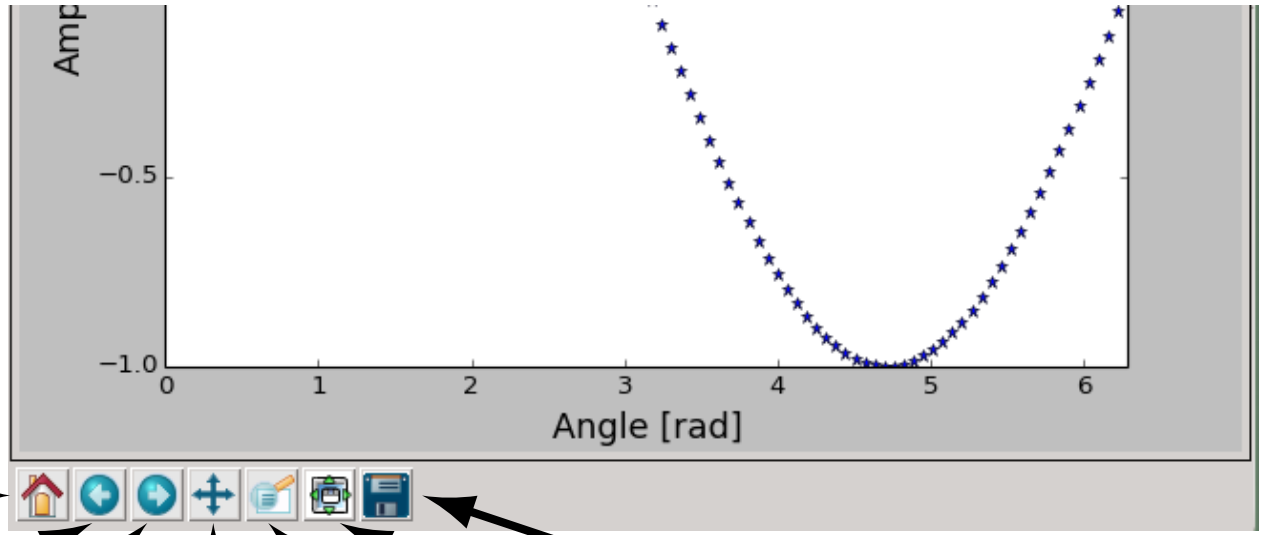
```
>>> import numpy as np

>>> import matplotlib.pyplot as plt
>>> x=np.linspace(0, 2*np.pi, 100)
>>> plt.plot(x, np.sin(x), '*')

>>> plt.xlim(0, 10)
(0, 10)

>>> plt.xlim(0, 2 * np.pi)
(0, 6.2831853071795862)

>>> plt.xlabel('Angle [rad]',
  fontsize = 16)

>>> plt.ylabel('Amplitude')
```




```
>>> plt.ylabel('Amplitude [m]',
  fontsize = 16)

>>> plt.title('Sine Wave',
  fontsize = 16)
```

SciNet compute • calcul CANADA

What are those buttons?



Return to where you started.

Go back a step.

Go forward a step.

Grab the plot and move it around.

Zoom in on a section.

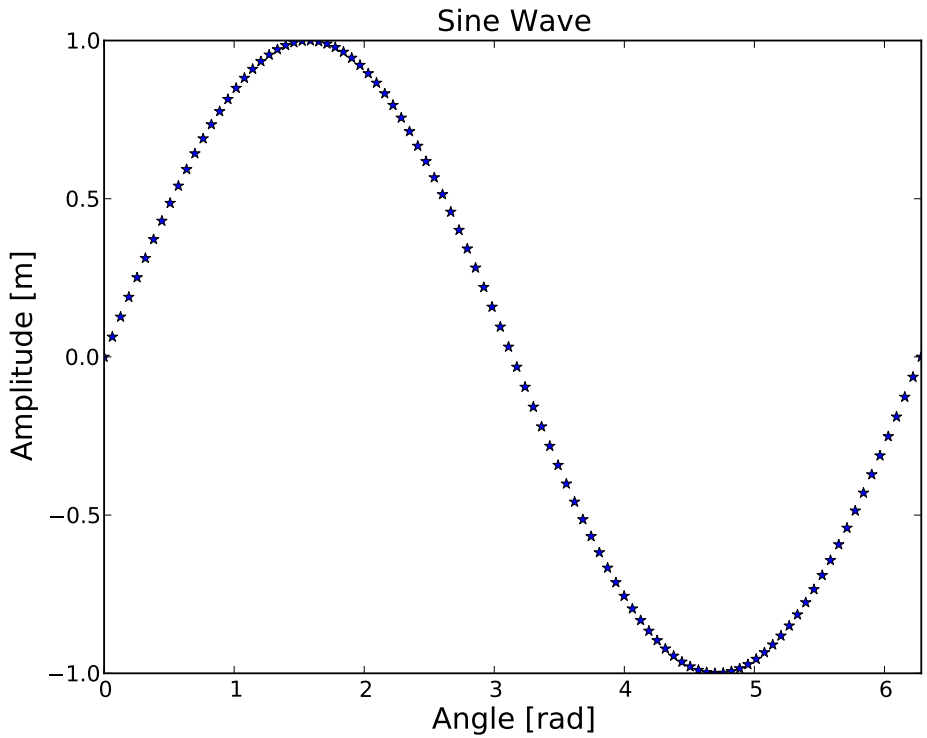
Adjust spacing between plots.

Save the figure.

SciNet compute • calcul CANADA

Playing with subplots

```
>>>
```

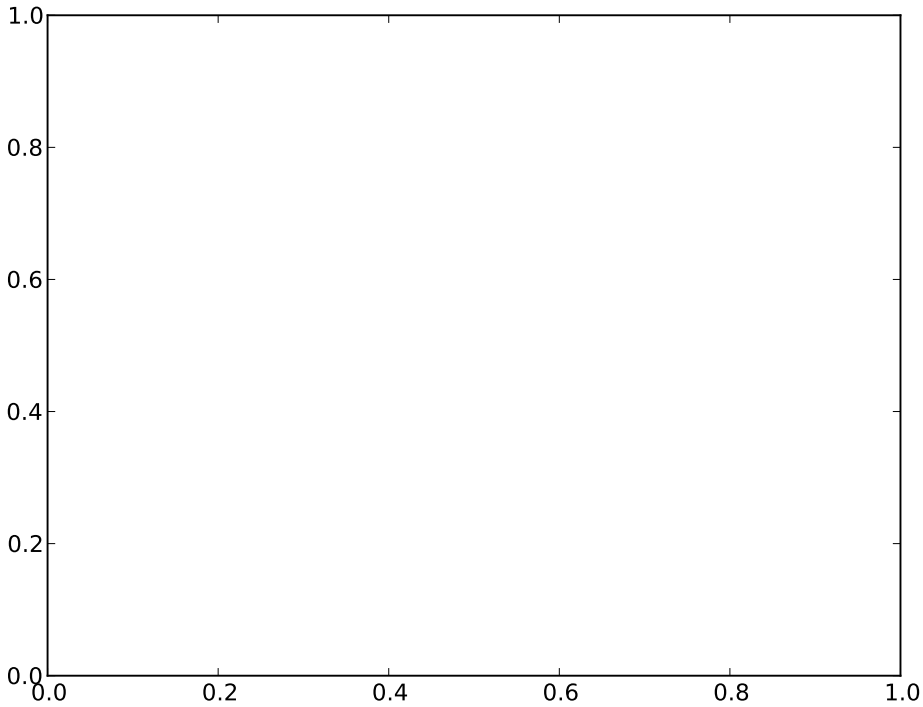


Playing with subplots

```
>>> plt.clf()

>>>
```

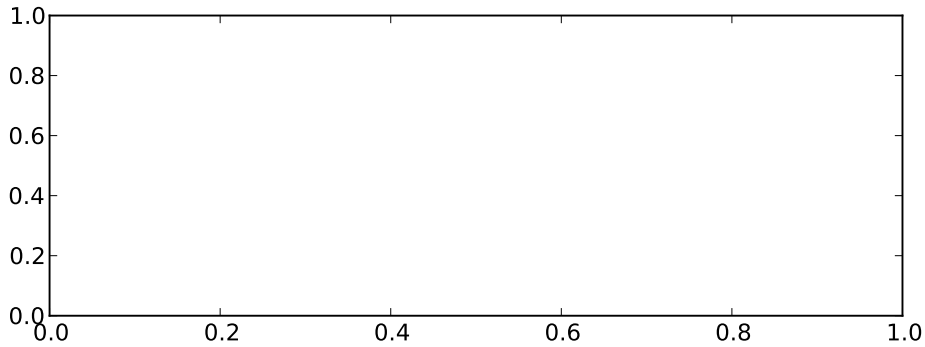
Playing with subplots

```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>>
```



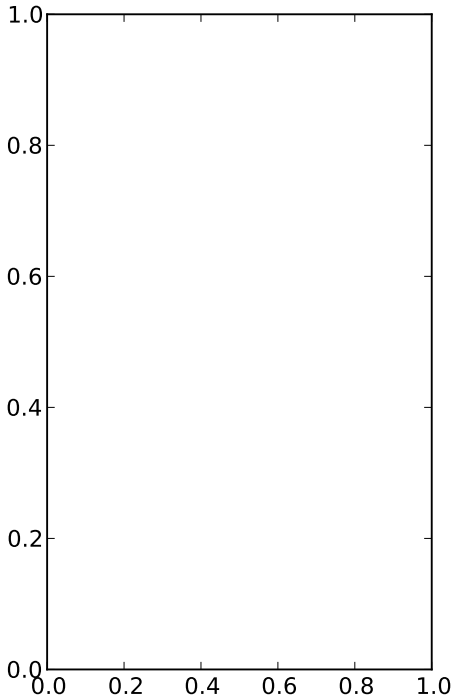
Playing with subplots

```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>>
```



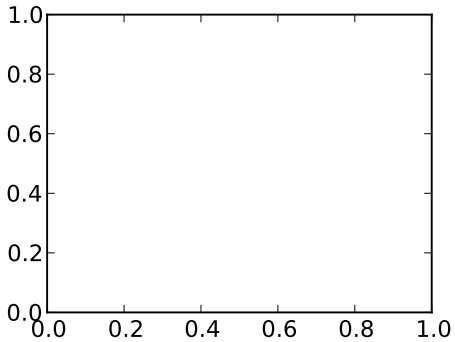
Playing with subplots

```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>>
```



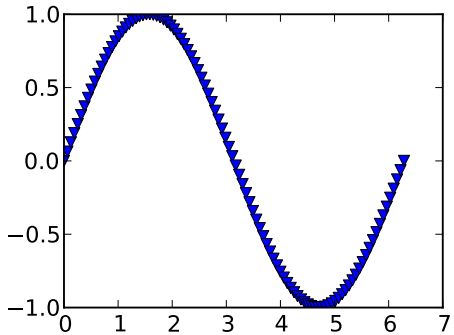
Playing with subplots

```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>> plt.subplot(2,2,1)

>>>
```



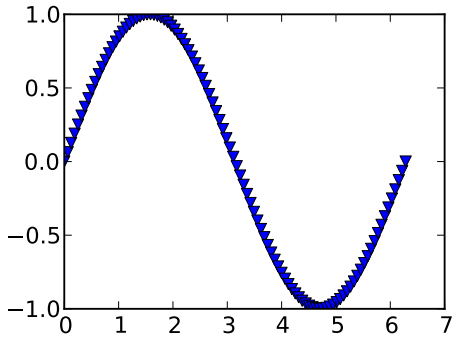
Playing with subplots

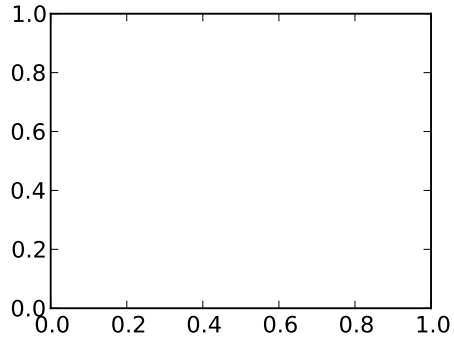
```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>> plt.subplot(2,2,1)

>>> plt.plot(x, np.sin(x), 'v')

>>>
```



Playing with subplots

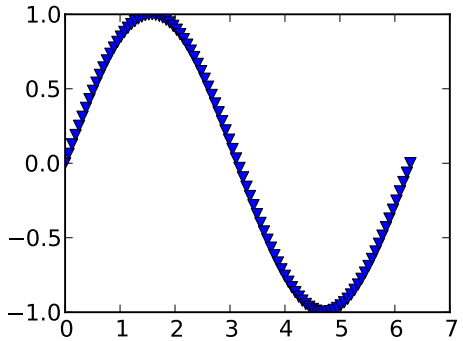
```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>> plt.subplot(2,2,1)

>>> plt.plot(x, np.sin(x), 'v')

>>> plt.subplot(2,2,4)

>>>
```

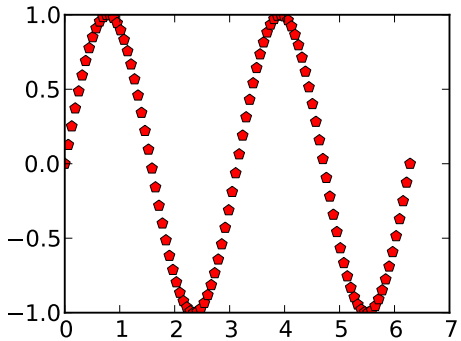




Playing with subplots

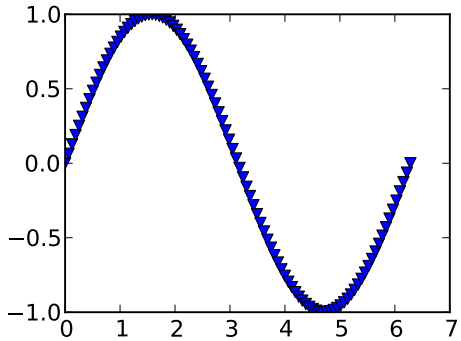
```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>> plt.subplot(2,2,1)

>>> plt.plot(x, np.sin(x), 'v')

>>> plt.subplot(2,2,4)

>>> plt.plot(x, np.sin(2 * x),'rp')

>>>
```

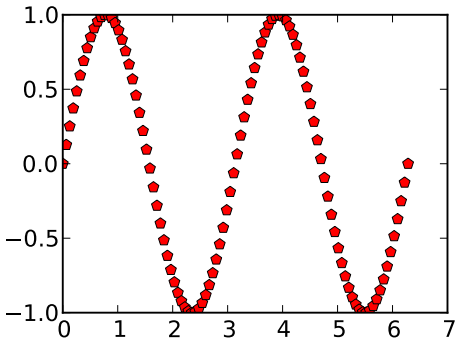




Playing with subplots

```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>> plt.subplot(2,2,1)

>>> plt.plot(x, np.sin(x), 'v')

>>> plt.subplot(2,2,4)

>>> plt.plot(x, np.sin(2 * x),'rp')

>>> plt.subplot(2,2,1)
```

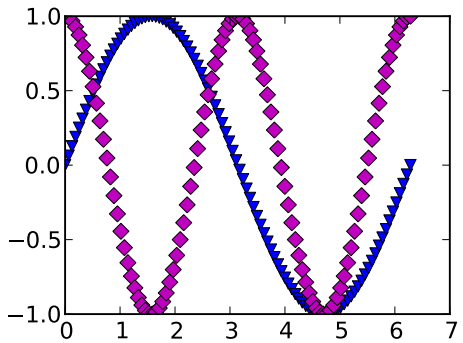


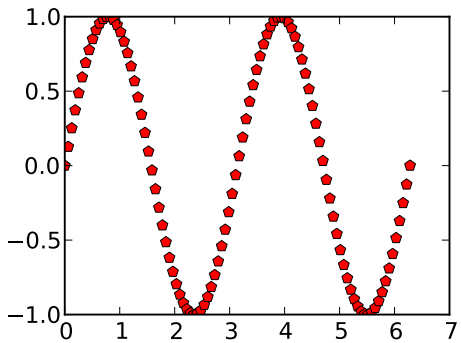



```
>>>
```

Playing with subplots

```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>> plt.subplot(2,2,1)

>>> plt.plot(x, np.sin(x), 'v')

>>> plt.subplot(2,2,4)

>>> plt.plot(x, np.sin(2 * x),'rp')

>>> plt.subplot(2,2,1)
```

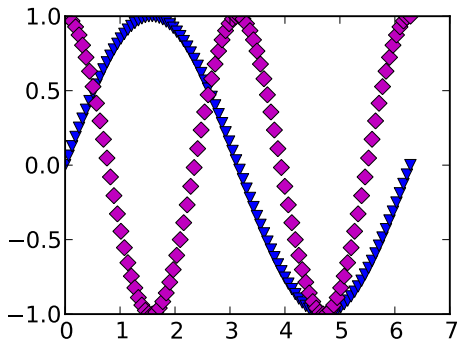


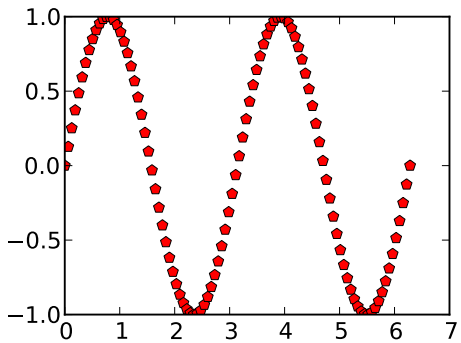



```
>>> plt.plot(x, np.cos(2 * x), 'mD')

>>>
```

Playing with subplots

```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>> plt.subplot(2,2,1)

>>> plt.plot(x, np.sin(x), 'v')

>>> plt.subplot(2,2,4)

>>> plt.plot(x, np.sin(2 * x),'rp')

>>> plt.subplot(2,2,1)
```

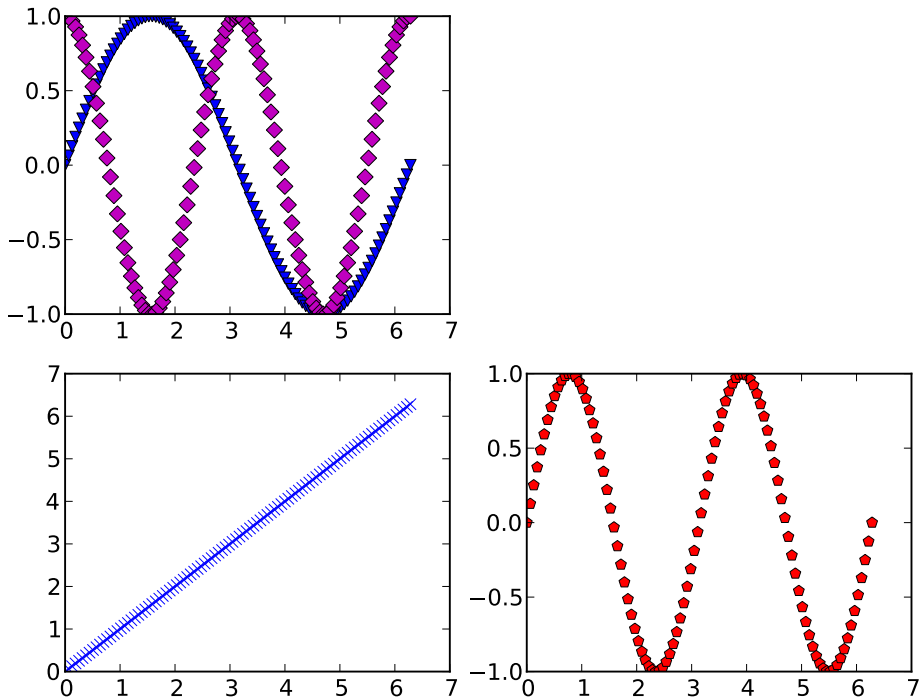





```
>>> plt.plot(x, np.cos(2 * x), 'mD')

>>> plt.subplot(2,2,3)

>>>
```

Playing with subplots

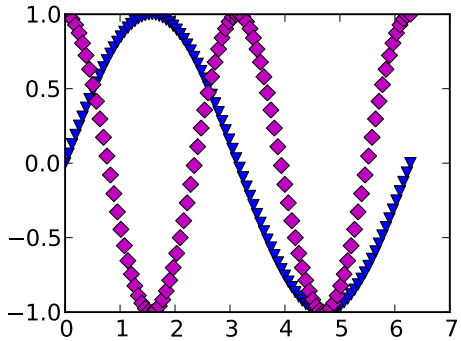
```
>>> plt.clf()

>>> plt.subplot(1,1,1)

>>> plt.subplot(2,1,1)

>>> plt.subplot(1,2,1)

>>> plt.subplot(2,2,1)

>>> plt.plot(x, np.sin(x), 'v')

>>> plt.subplot(2,2,4)

>>> plt.plot(x, np.sin(2 * x),'rp')

>>> plt.subplot(2,2,1)
```

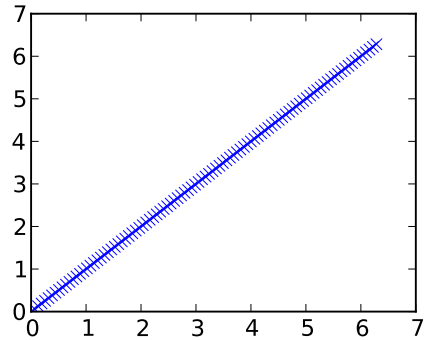


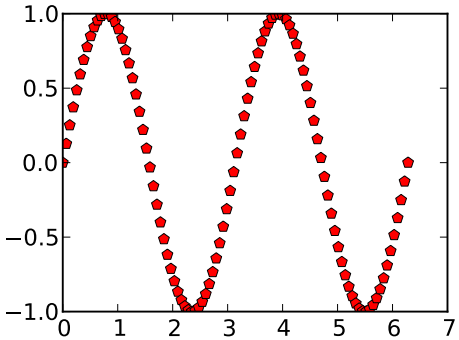
```
>>> plt.plot(x, np.cos(2 * x), 'mD')

>>> plt.subplot(2,2,3)

>>> plt.plot(x, x, 'x')
```

Using error bars

```
>>>
```







Using error bars

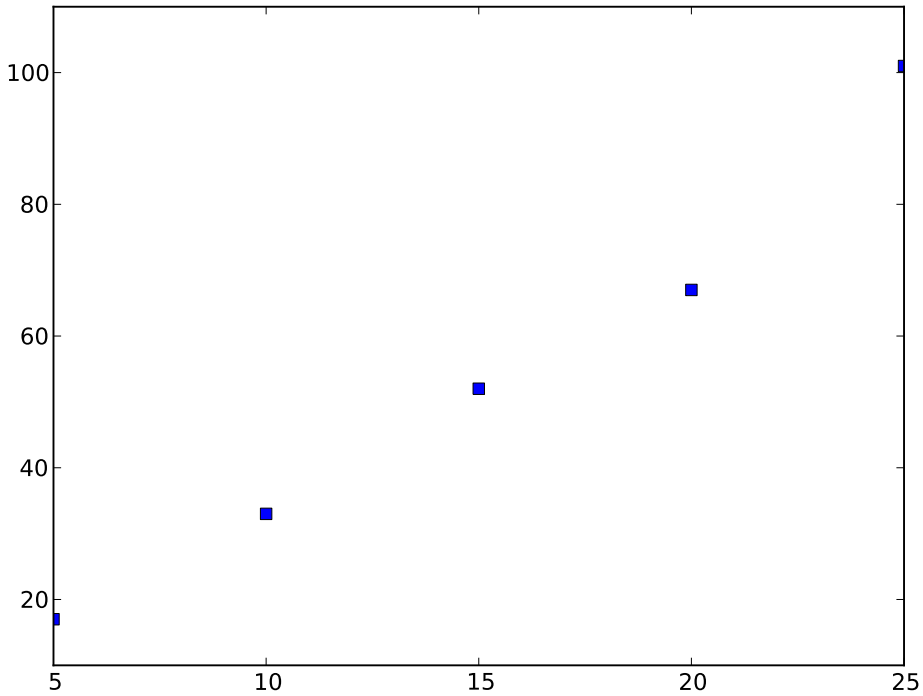
```
>>> plt.clf()

>>>
```

Using error bars

```
>>> plt.clf()

>>> o2 = np.arange(5, 26, 5)
>>> o1 = [17, 33, 52, 67, 101]

>>>
```

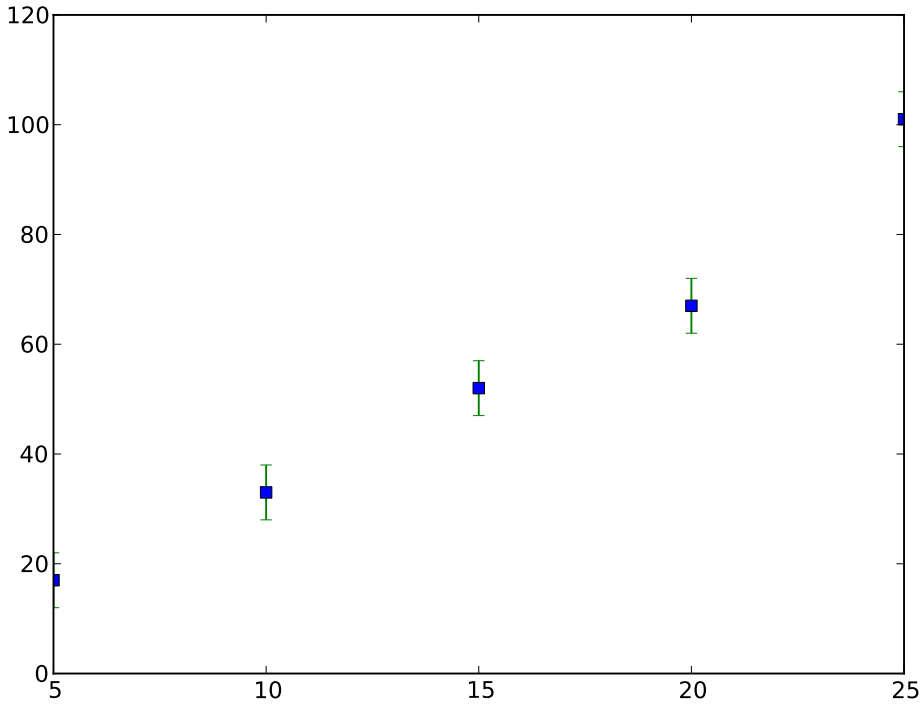
Using error bars

```
>>> plt.clf()

>>> o2 = np.arange(5, 26, 5)
>>> o1 = [17, 33, 52, 67, 101]

>>> plt.plot(o2, o1, 's',
  label = 'Raw Data')

>>>
```



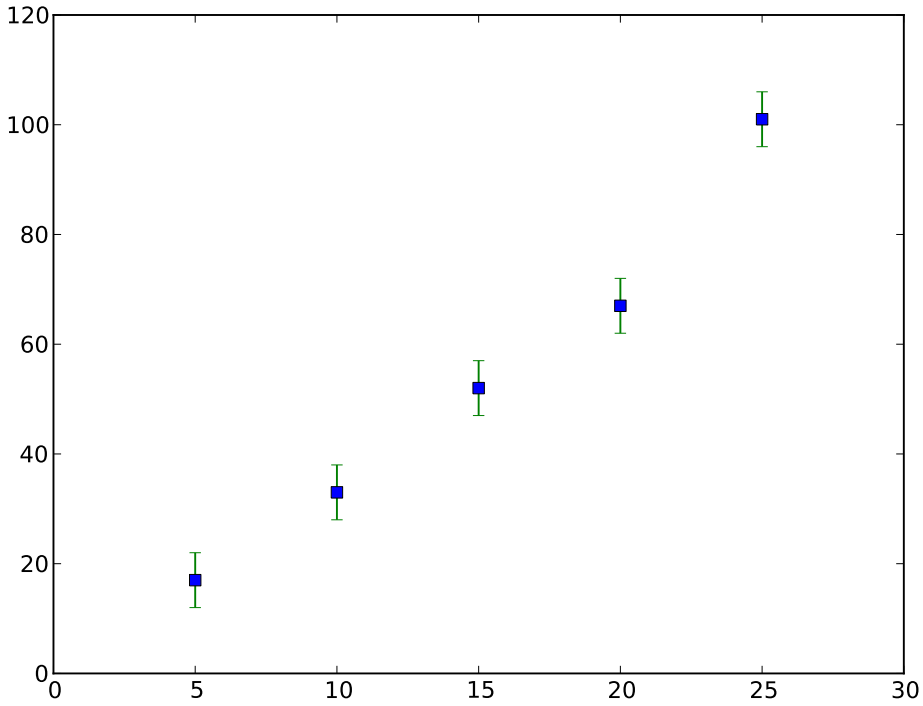
Using error bars

```
>>> plt.clf()

>>> o2 = np.arange(5, 26, 5)
>>> o1 = [17, 33, 52, 67, 101]

>>> plt.plot(o2, o1, 's',
  label = 'Raw Data')

>>> plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

>>>
```



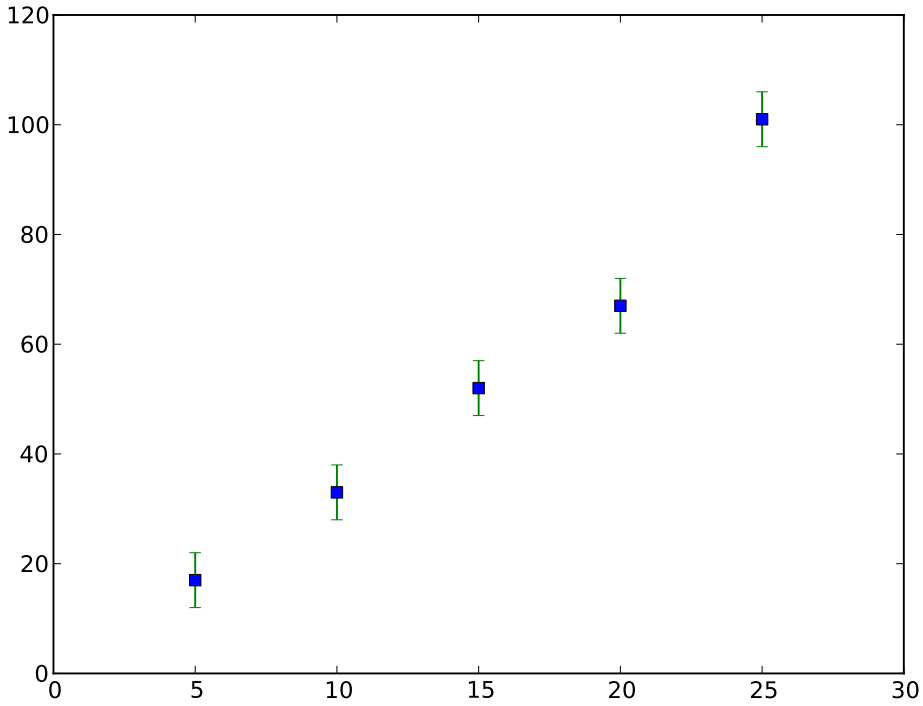
Using error bars

```
>>> plt.clf()

>>> o2 = np.arange(5, 26, 5)
>>> o1 = [17, 33, 52, 67, 101]

>>> plt.plot(o2, o1, 's',
  label = 'Raw Data')

>>> plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

>>> plt.xlim(0, 30)

>>>
```



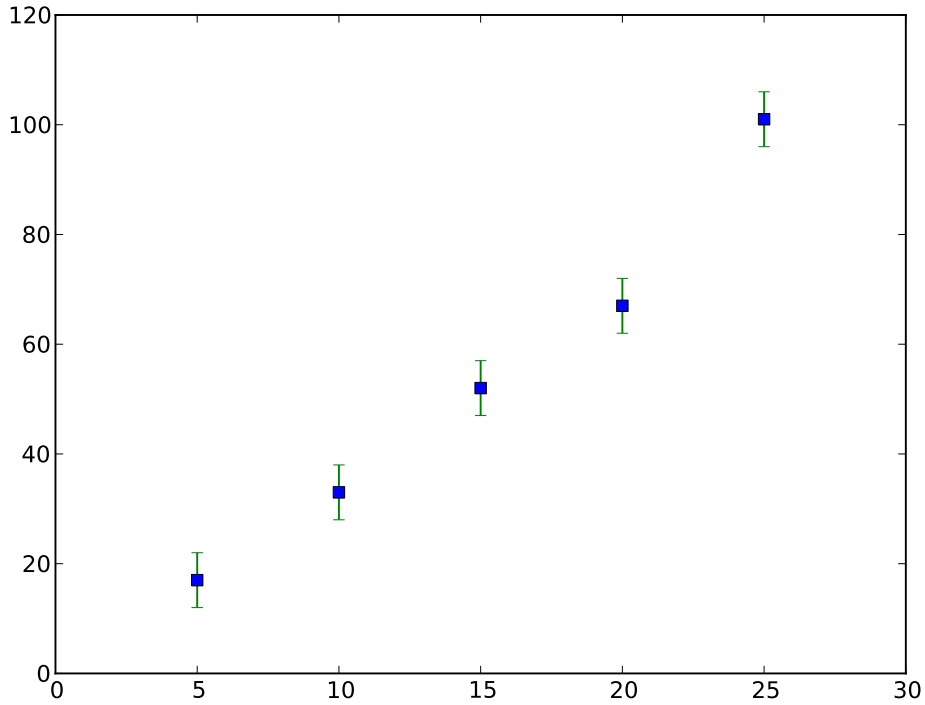
Using error bars

```
>>> plt.clf()

>>> o2 = np.arange(5, 26, 5)
>>> o1 = [17, 33, 52, 67, 101]

>>> plt.plot(o2, o1, 's',
  label = 'Raw Data')

>>> plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

>>> plt.xlim(0, 30)

>>> fit = np.polyfit(o2, o1, 1)

>>>
```



Using error bars

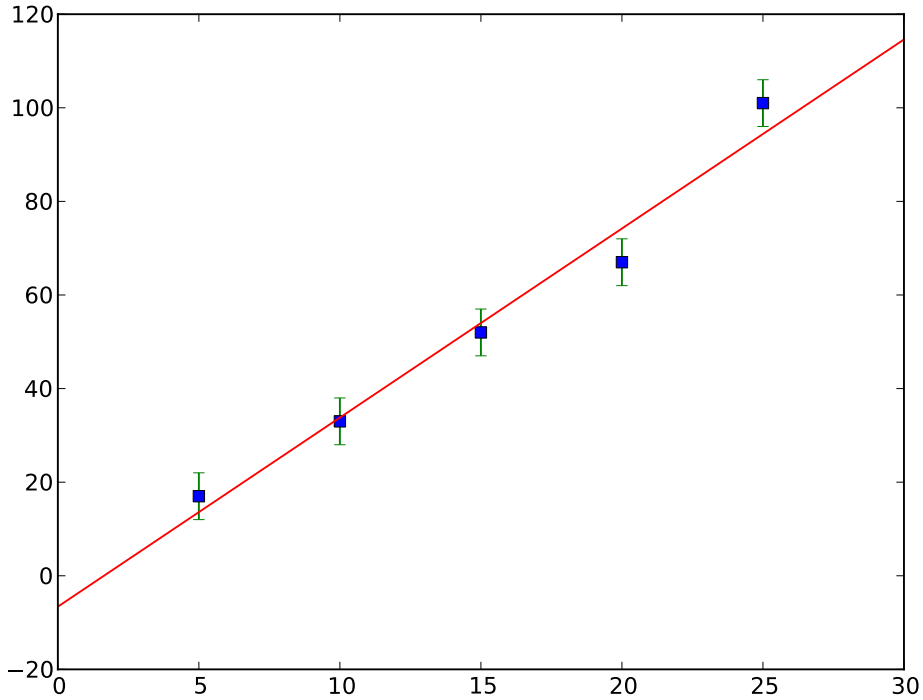
```
>>> plt.clf()

>>> o2 = np.arange(5, 26, 5)
>>> o1 = [17, 33, 52, 67, 101]

>>> plt.plot(o2, o1, 's',
  label = 'Raw Data')

>>> plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

>>> plt.xlim(0, 30)

>>> fit = np.polyfit(o2, o1, 1)

>>> x = np.linspace(0, 30, 100)
```




```
>>>
```

Using error bars

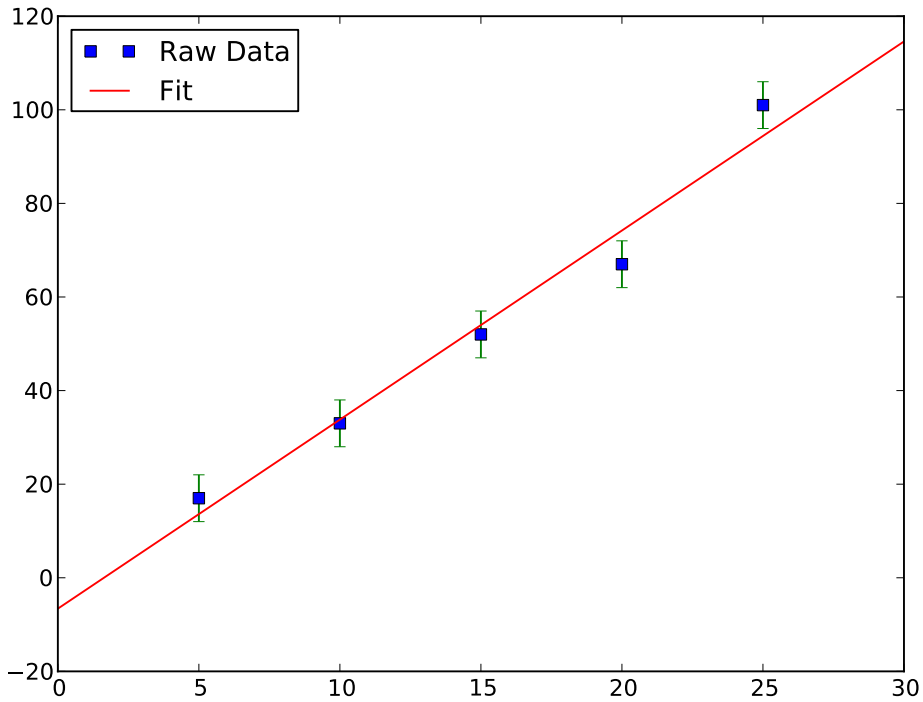
```
>>> plt.clf()

>>> o2 = np.arange(5, 26, 5)
>>> o1 = [17, 33, 52, 67, 101]

>>> plt.plot(o2, o1, 's',
  label = 'Raw Data')

>>> plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

>>> plt.xlim(0, 30)

>>> fit = np.polyfit(o2, o1, 1)

>>> x = np.linspace(0, 30, 100)
```




```
>>> plt.plot(x, np.polyval(fit,x),
  label = 'Fit')

>>>
```

Using error bars

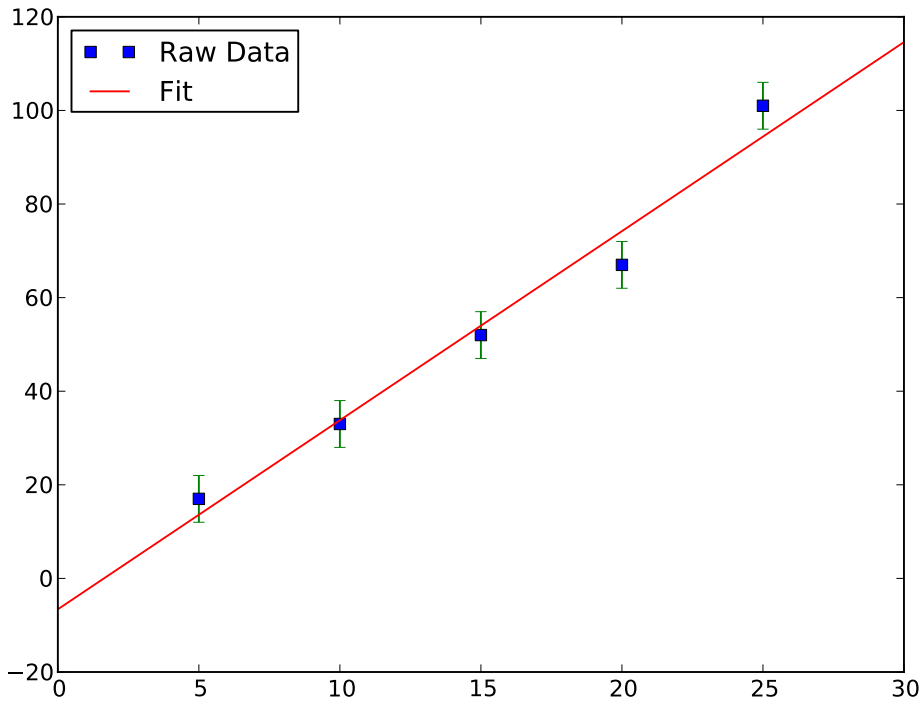
```
>>> plt.clf()

>>> o2 = np.arange(5, 26, 5)
>>> o1 = [17, 33, 52, 67, 101]

>>> plt.plot(o2, o1, 's',
  label = 'Raw Data')

>>> plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

>>> plt.xlim(0, 30)

>>> fit = np.polyfit(o2, o1, 1)

>>> x = np.linspace(0, 30, 100)
```




```
>>> plt.plot(x, np.polyval(fit,x),
  label = 'Fit')

>>> plt.legend(loc = 'best')
```

2D plotting

```
>>>
```



2D plotting

```
>>> plt.clear()

>>>
```

2D plotting

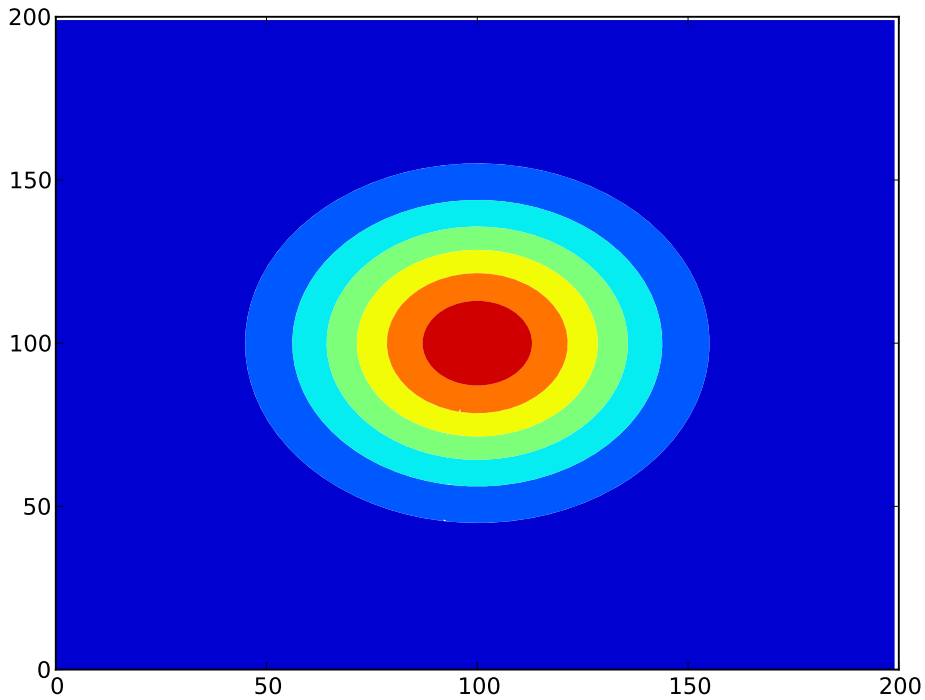
```
>>> plt.clear()

>>> x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

>>>
```

2D plotting

```
>>> plt.clear()

>>> x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

>>> g =
  np.exp(-(x**2 + y**2) / 16)

>>>
```

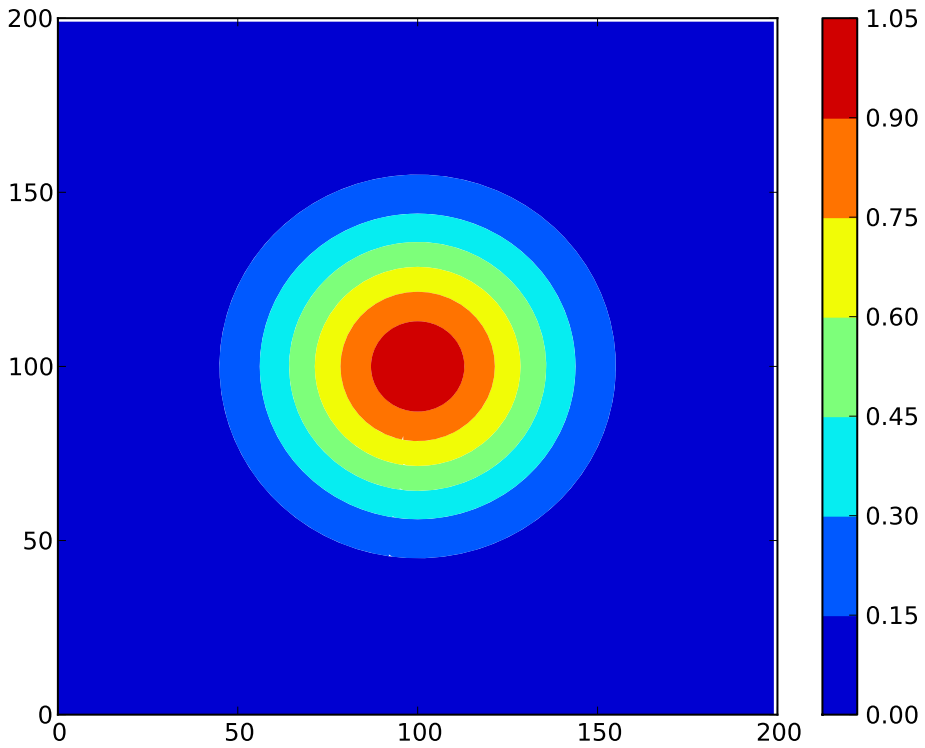
2D plotting

```
>>> plt.clear()

>>> x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

>>> g =
  np.exp(-(x**2 + y**2) / 16)

>>> plt.contourf(g)

>>>
```



2D plotting

```
>>> plt.clear()

>>> x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

>>> g =
  np.exp(-(x**2 + y**2) / 16)

>>> plt.contourf(g)

>>> plt.colorbar()

>>>
```



2D plotting

```
>>> plt.clear()

>>> x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

>>> g =
  np.exp(-(x**2 + y**2) / 16)

>>> plt.contourf(g)

>>> plt.colorbar()

>>> plt.clf()

>>>
```

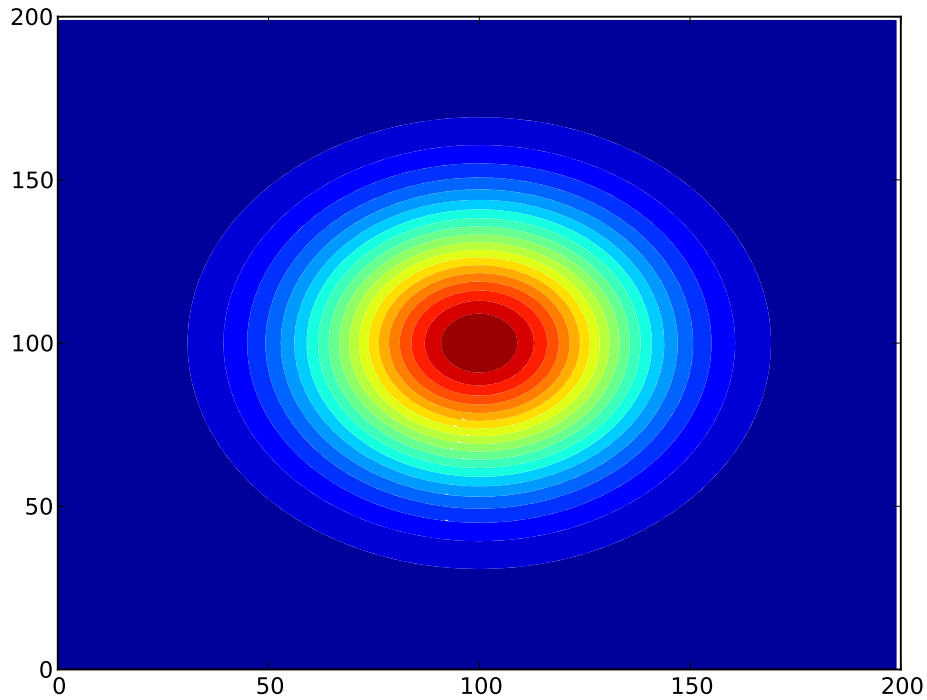
2D plotting

```
>>> plt.clear()

>>> x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

>>> g =
  np.exp(-(x**2 + y**2) / 16)

>>> plt.contourf(g)

>>> plt.colorbar()

>>> plt.clf()

>>> V = np.linspace(g.min(),
  g.max(), 21)
```

```
>>>
```

2D plotting

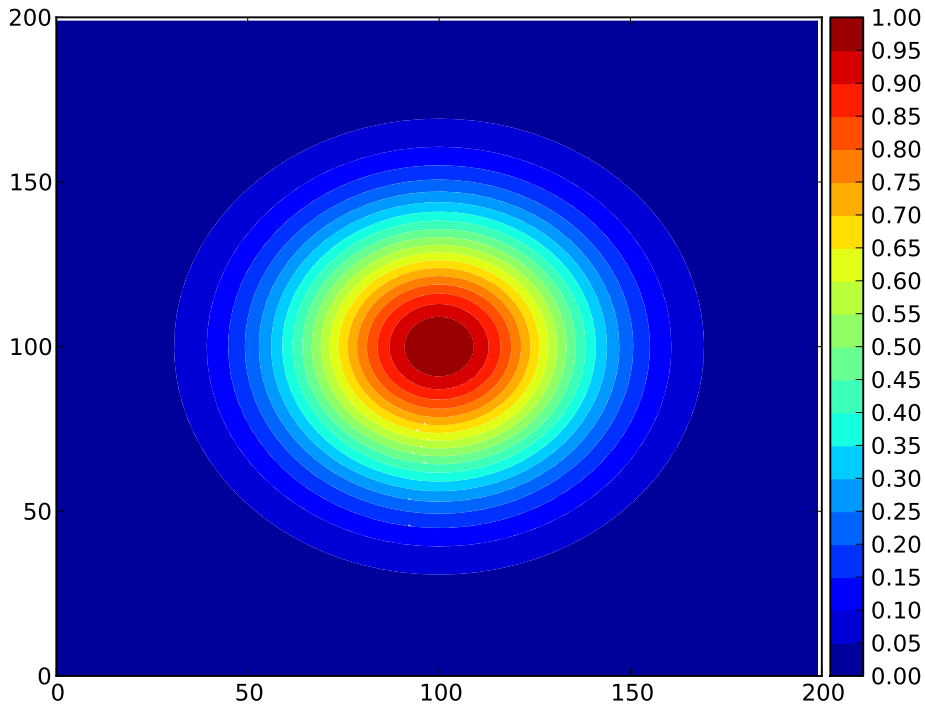
```
>>> plt.clear()

>>> x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

>>> g =
  np.exp(-(x**2 + y**2) / 16)

>>> plt.contourf(g)

>>> plt.colorbar()

>>> plt.clf()

>>> V = np.linspace(g.min(),
  g.max(), 21)
```




```
>>> plt.contourf(g, V)

>>>
```

2D plotting

```
>>> plt.clear()

>>> x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

>>> g =
  np.exp(-(x**2 + y**2) / 16)

>>> plt.contourf(g)

>>> plt.colorbar()

>>> plt.clf()

>>> V = np.linspace(g.min(),
  g.max(), 21)
```




```
>>> plt.contourf(g, V)

>>> plt.colorbar(format = '%.2f',
  pad = 0.01, fraction = 0.09,
  ticks = V)
```

SciNet compute • calcul CANADA

pyplot.figure() arguments

The pyplot.figure() object is the plotting canvas in Python that everyone tends to use. It has a couple of optional arguments of note:

- dpi: the figure resolution (dots per inch).
- figsize: a tuple, which indicates the dimensions of the figure, in inches.

Use these arguments when you are creating figures for your papers, and when you do the homework assignment.

SciNet compute • calcul CANADA

Figure confusion

There is a source of confusion online in the way figure() can be adjusted:

- pyplot.figure() is actually an interface to the matplotlib.figure.Figure() object.
- As mentioned earlier, interactive mode (which is what ipython provides) only works on pyplot commands, not matplotlib Figure-object functions.
- Consequently many of these object functions don't appear to work, since nothing seems to happen.
- If you invoke a command on a matplotlib object, you must still use show() to update the plot.

The code presented here should work, but be aware of this source of confusion when searching for more functionality.

Figure object functions

The following functions are part of the matplotlib.figure.Figure object:

- gca: returns the axes, which refers to the plotting environment itself, where the data lives. Will create an axes if one does not already exist.
- add_subplot: adds a subplot to an existing figure object.
- set_dpi: set the figure resolution.
- set_figwidth, set_figheight: set the dimensions, in inches.
- subplots_adjust: change the positioning of the subplots.

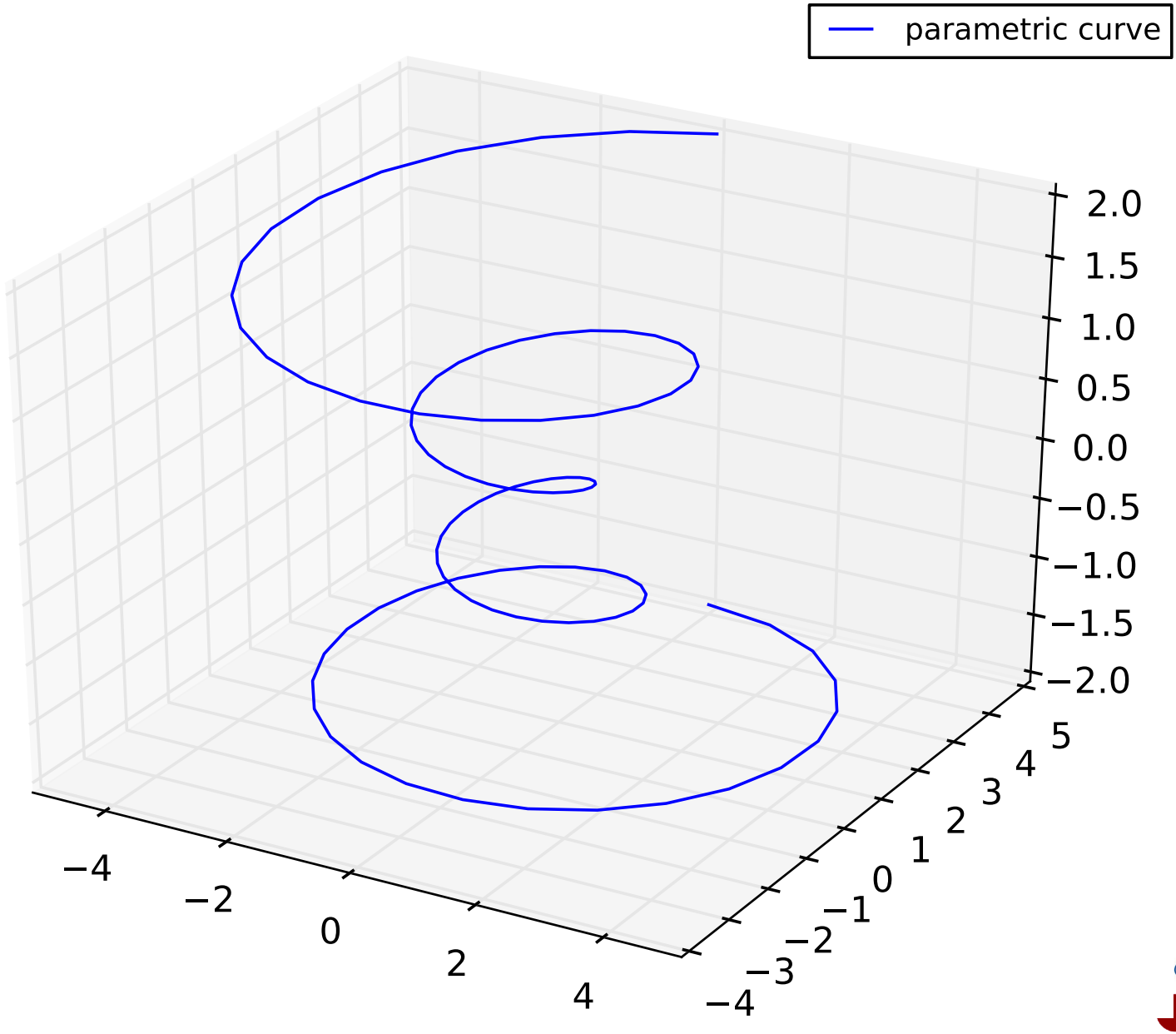
If you're writing code which will be in a script, test it against the standard Python prompt to make sure that it will work.

3D plotting

Plotting in 3 dimensions is a little more complicated than in 2.

```
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np, matplotlib.pyplot as plt

mpl.rcParams['legend.fontsize'] = 10

fig = plt.figure()                      # Open a figure.
ax = fig.gca(projection = '3d')         # Create an axis, of 3d projection.

theta = np.linspace(-4 * np.pi, 4 * np.pi, 100)
z = np.linspace(-2, 2, 100);   r = z**2 + 1
x = r * np.sin(theta);   y = r * np.cos(theta)

ax.plot(x, y, z, label='parametric curve')
ax.legend()

plt.savefig('curve.eps')
plt.show()
```

3D plot



matplotlib.rcParams

You may have noticed the use of the matplotlib.rcParams variable in the last block of code. What is that?

- rcParams is a dictionary which controls the default values of all of the matplotlib parameters.
- If you're curious about what's in there, import matplotlib and print it out.
- If you want to change values of parameters before plotting something, you can do it there.
- However, be aware that as long as the module is in memory the changes will persist.
- It may be best to make a copy of the variable before modifying it, and then reset it when you're done.

SciNet compute • calcul CANADA

Professional plotting

You should make your work presentable. That means making a serious effort to make your results easy for your audience to understand.

Suggestions:

- DO label *everything*: axes, lines, fits, data.
- DO put units on your axis labels, including colourbars.
- DO use legends, where appropriate.
- DO adjust the font size of axis and tick labels so that they can actually be read.
- DO NOT put a title label over your plot (for talks and papers).
- DO NOT use colours that cannot be read on a white background (yellow, orange, light green, cyan). This is especially important for figures used in talks.
- DO make your data fill the plot.

More plotting for professional scientists

It's also important to consider file types and sizes.

More suggestions:

- DO set the image size and resolution to that requested by the journal you're submitting to.
- DO NOT use bitmap or stroke fonts for your plot. These cannot be rescaled properly, which is often needed for publication. Use vector fonts (the default for Python).
- If possible, DO NOT use image file types that cannot be scaled (bitmap, jpeg). Use EPS or PDF.
- DO NOT leave a bunch of white space around the outside of your plots.
- DO make a script (in Python or whatever language), whose sole purpose is to make that plot for your paper.

SciNet compute • calcul CANADA

Chart junk!

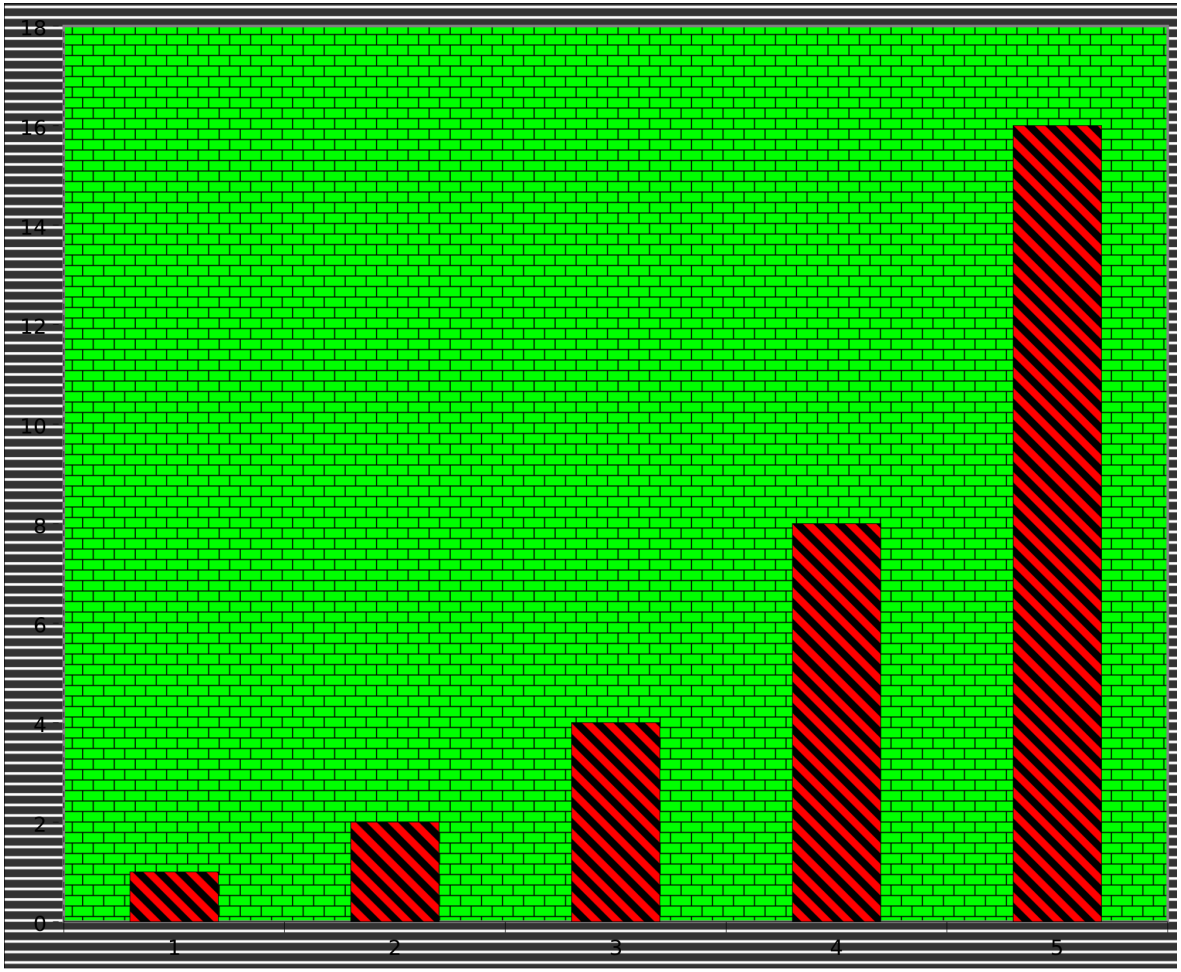


Chart junk is the unnecessary cluttering of your plot. Don't do it!

Image stolen from Wikipedia.

Advanced plotting

```
>>>
```

Advanced plotting

```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>>
```

Advanced plotting

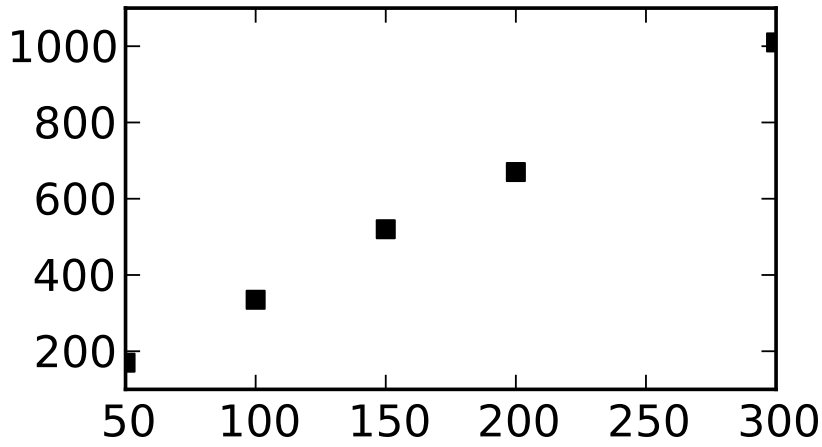
```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>>
```

Advanced plotting

```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>>
```

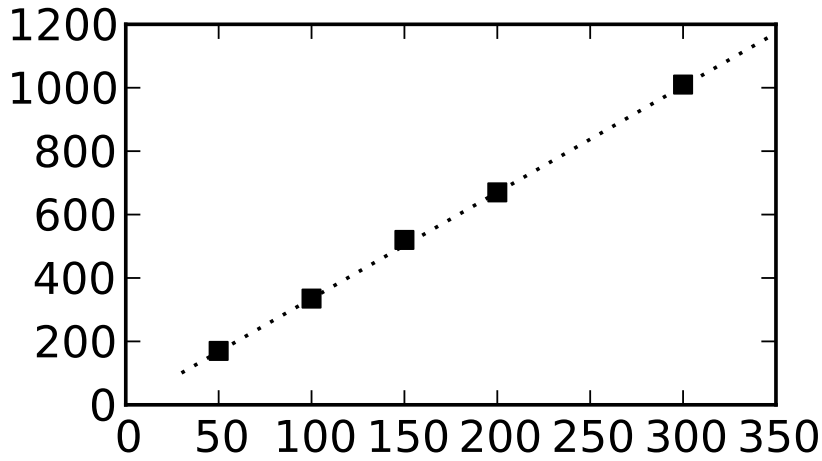
Advanced plotting

```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>>
```



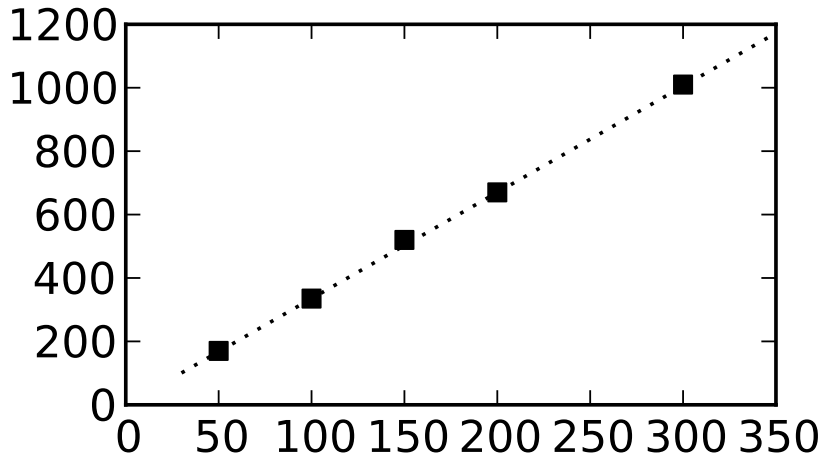
Advanced plotting

```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>>
```



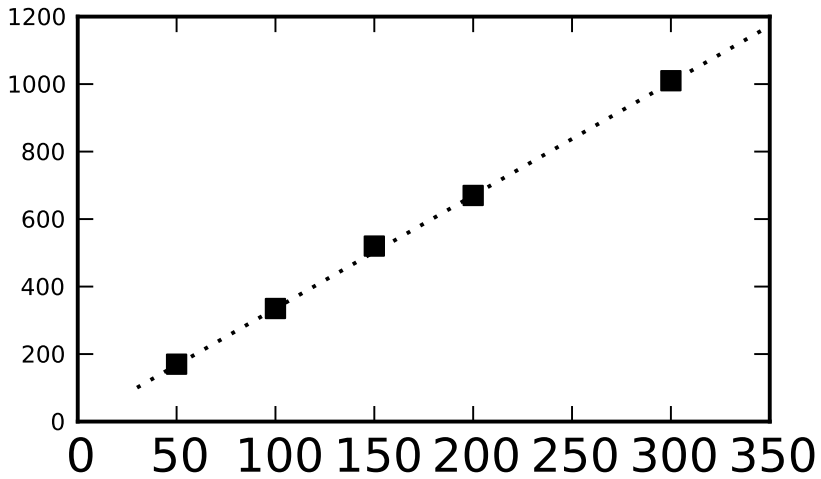
Advanced plotting

```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>> for t in a.yaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>>
```



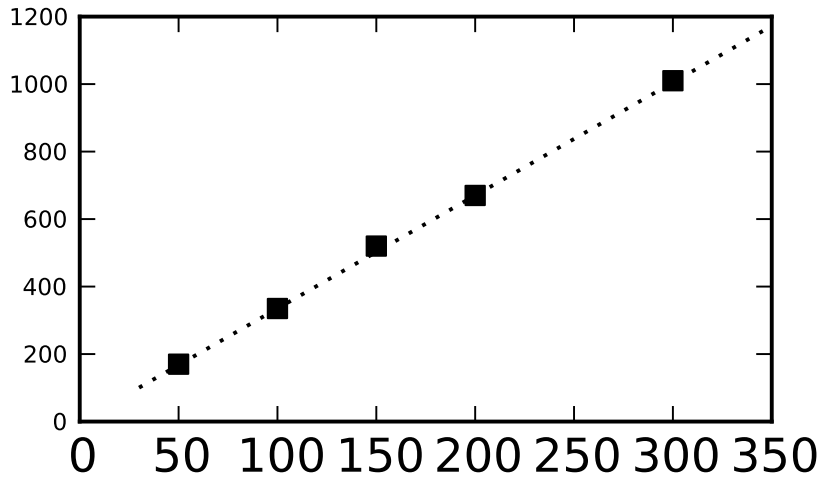
Advanced plotting

```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>> for t in a.yaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
>>>
```



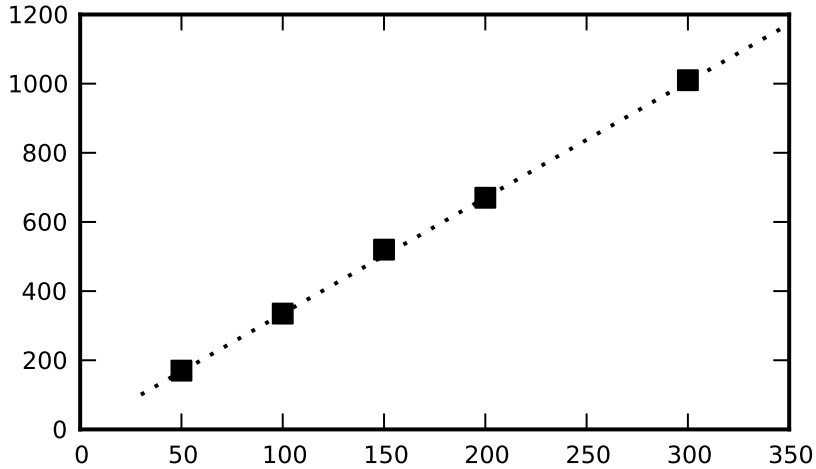
Advanced plotting

```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>> for t in a.yaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
>>> for t in a.xaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>>
```



Advanced plotting

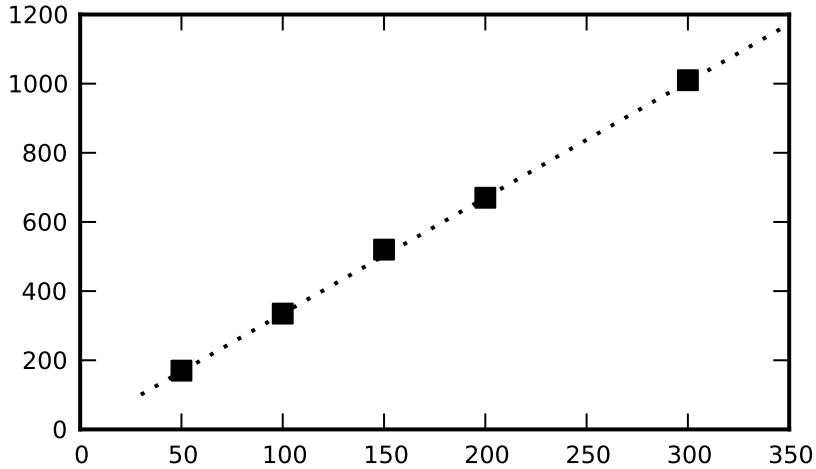
```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>> for t in a.yaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
>>> for t in a.xaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
```




```
>>>
```

Advanced plotting

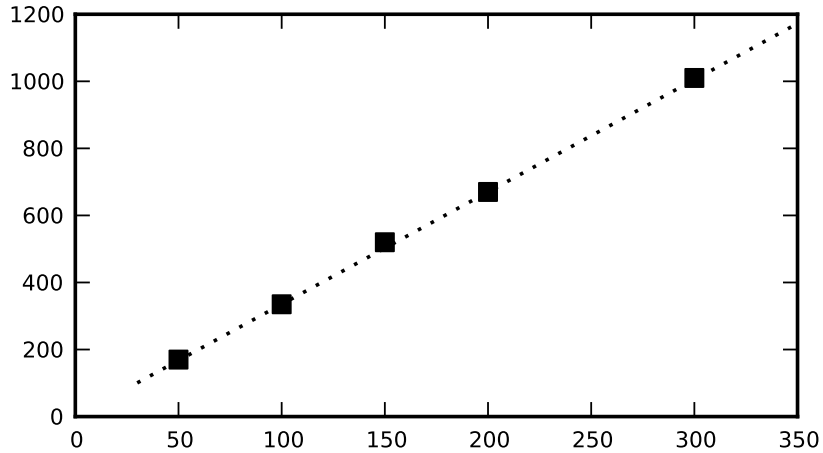
```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>> for t in a.yaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
>>> for t in a.xaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
```




```
>>> fig.subplots_adjust(
  top = 0.97, right = 0.98,
  left = 0.15, bottom = 0.15)

>>>
```

Advanced plotting

```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>> for t in a.yaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
>>> for t in a.xaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
```

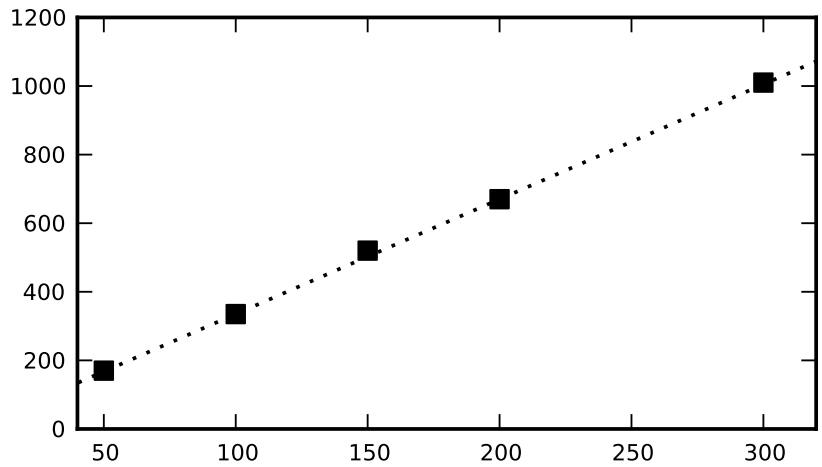



```
>>> fig.subplots_adjust(
  top = 0.97, right = 0.98,
  left = 0.15, bottom = 0.15)

>>> plt.show()
>>>
```

compute • calcul CANADA

Advanced plotting

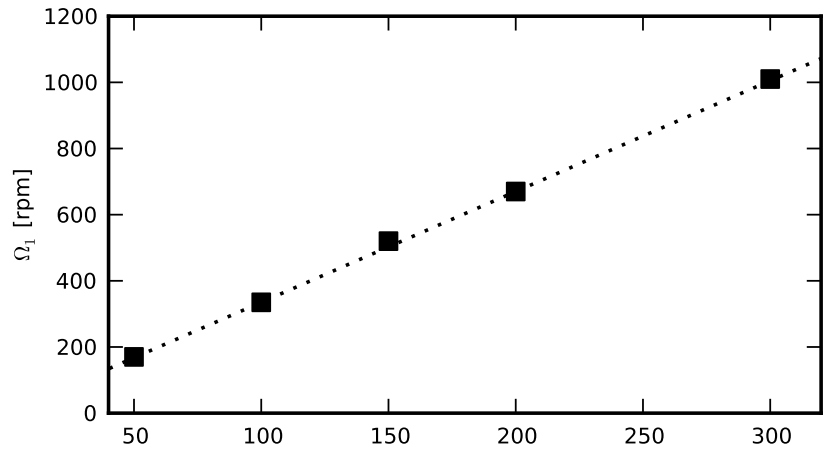
```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>> for t in a.yaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
>>> for t in a.xaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
```




```
>>> fig.subplots_adjust(
  top = 0.97, right = 0.98,
  left = 0.15, bottom = 0.15)

>>> plt.show()
>>> plt.xlim(40, 320)
(40, 320)
>>>
```

Advanced plotting

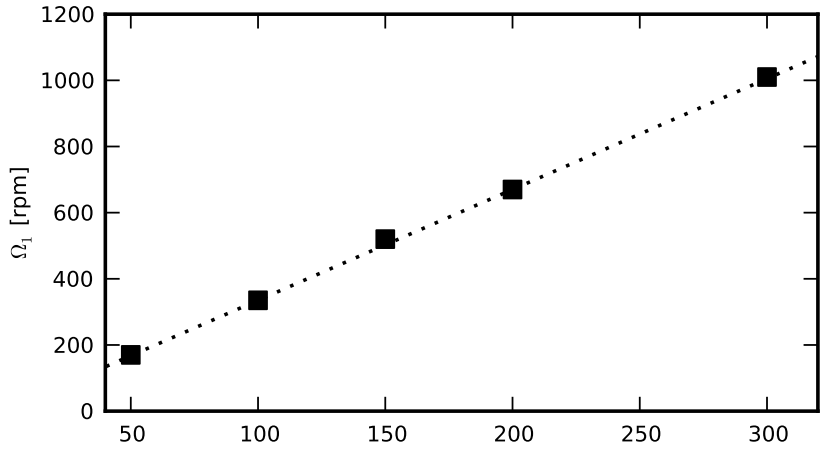
```
>>> o2 = [50, 100, 150, 200, 300]
>>> o1 = [170, 335, 520, 670, 1010]
>>> o = linspace(30, 350, 100)
>>> fig = plt.figure(
...  figsize = (3.375,2), dpi = 600)
...
>>> a = fig.add_subplot(1,1,1)

>>> plt.plot(o2, o1, 'ks', markersize=5)
>>> plt.plot(o, 3.35 * o, ':',color='k')

>>> for t in a.yaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
>>> for t in a.xaxis.get_major_ticks():
...  t.label.set_fontsize(6)
...
>>> plt.show()
```




```
>>> fig.subplots_adjust(
  top = 0.97, right = 0.98,
  left = 0.15, bottom = 0.15)

>>> plt.show()
>>> plt.xlim(40, 320)
(40, 320)
>>> plt.ylabel(
  r'$\Omega_1$ [rpm]', fontsize = 6,
  verticalalignment = 'center')
```

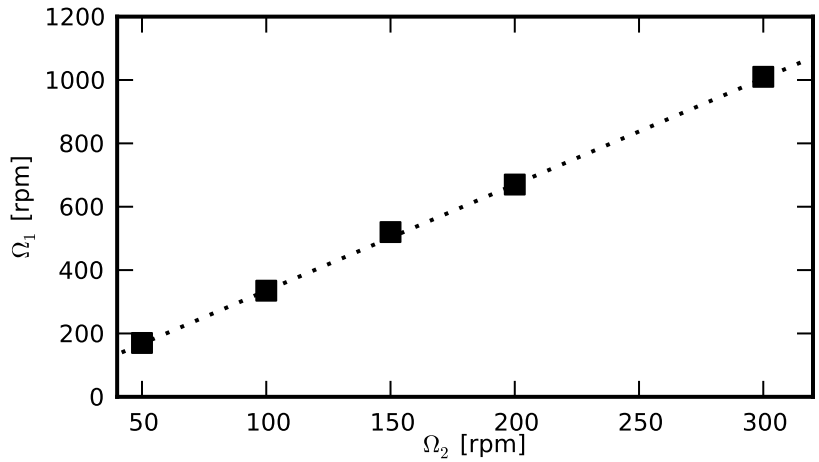
Advanced plotting, continued

```
>>>
```



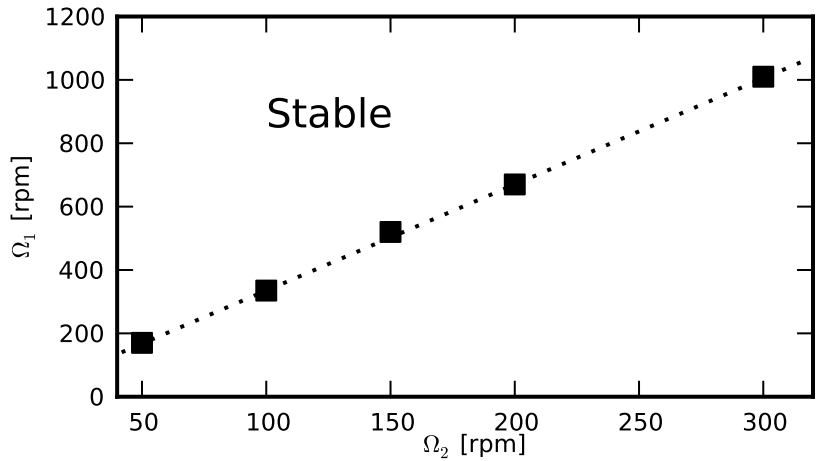
Advanced plotting, continued

```
>>> plt.text(163, -180.,
  r'$\Omega_2$ [rpm]', fontsize = 6)

>>>
```



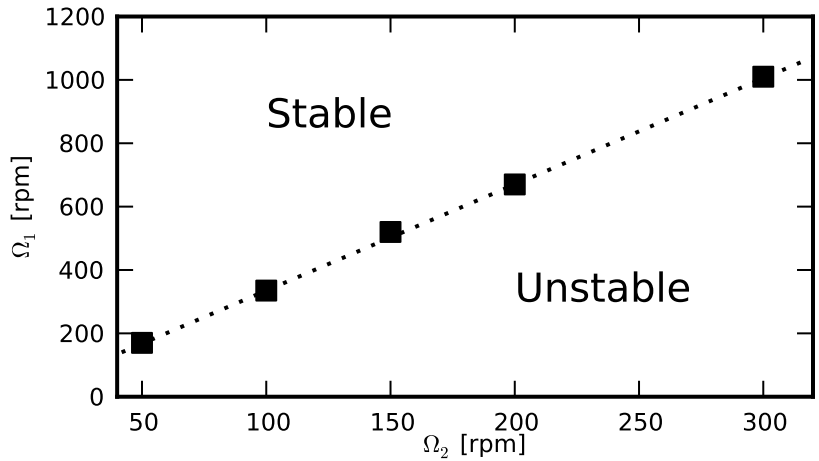
Advanced plotting, continued

```
>>> plt.text(163, -180.,
  r'$\Omega_2$ [rpm]', fontsize = 6)

>>> plt.text(100, 850., 'Stable',
  fontsize = 10)

>>>
```



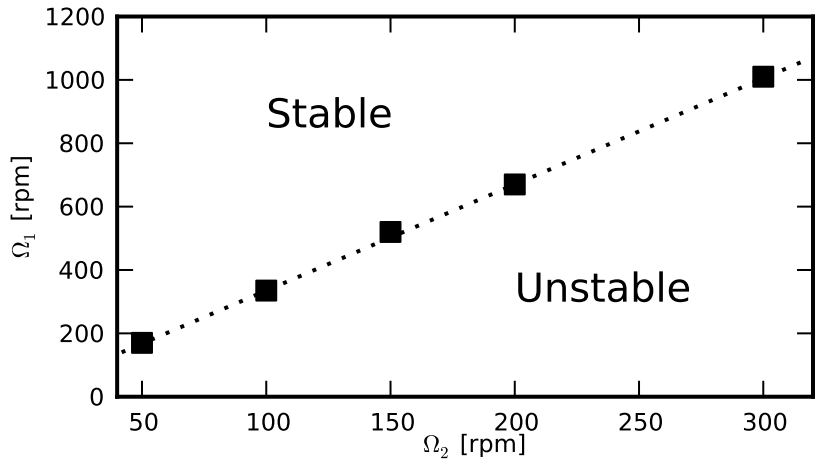
Advanced plotting, continued

```
>>> plt.text(163, -180.,
  r'$\Omega_2$ [rpm]', fontsize = 6)

>>> plt.text(100, 850., 'Stable',
  fontsize = 10)

>>> plt.text(200, 300., 'Unstable',
  fontsize = 10)

>>>
```



Advanced plotting, continued

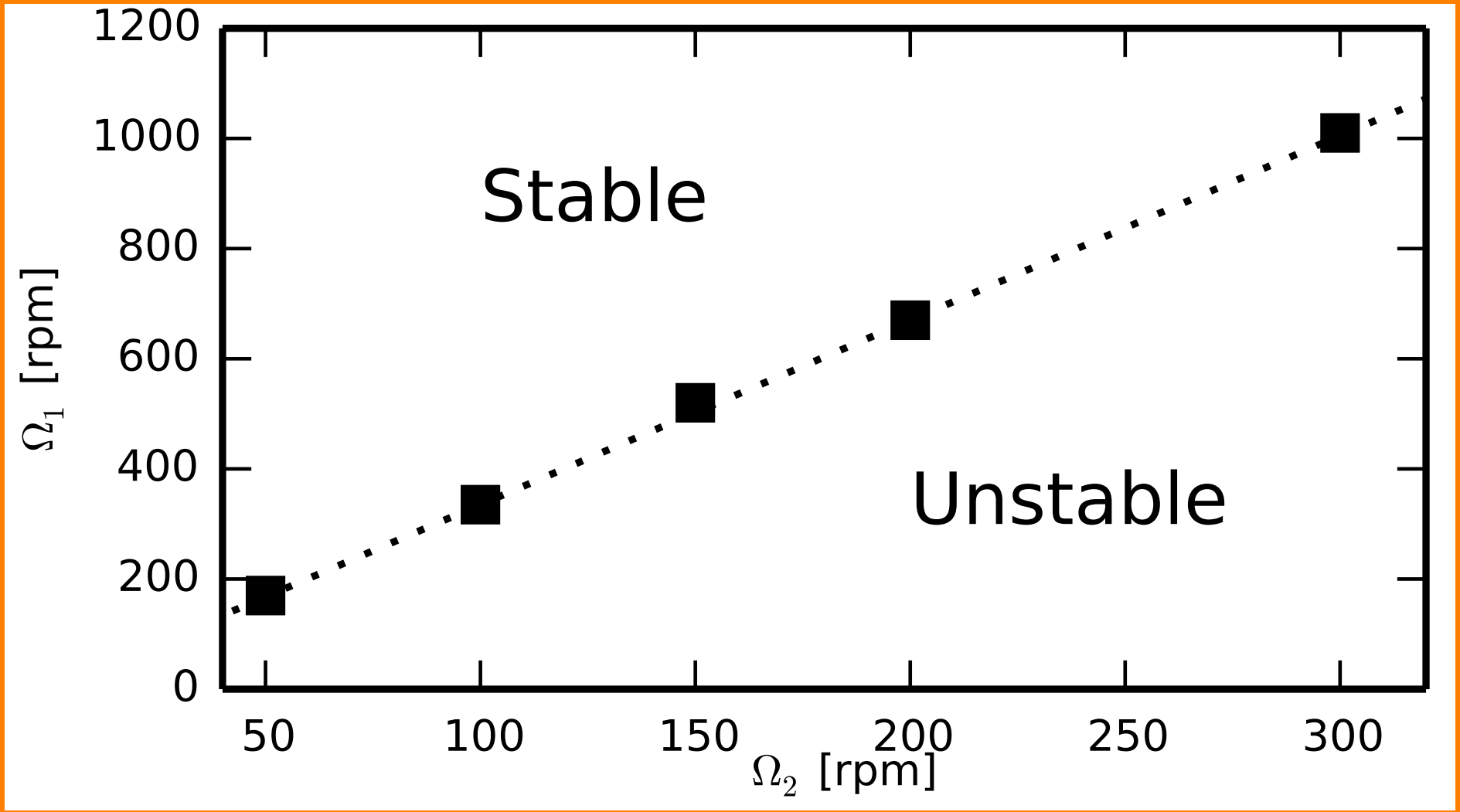
```
>>> plt.text(163, -180.,
  r'$\Omega_2$ [rpm]', fontsize = 6)

>>> plt.text(100, 850., 'Stable',
  fontsize = 10)

>>> plt.text(200, 300., 'Unstable',
  fontsize = 10)

>>> plt.savefig(
  'critical_rossby.eps', dpi = 600)
```



Use savefig to save your figures.

- Possible file types include EPS, PDF, PNG, and others.
- JPEG is not a valid type.

Final product



Another plotting example

```
>>>
```

Another plotting example

```
>>> r,z,s,psi = calc_psi_pop2011()
>>>
```

Another plotting example

```
>>> r,z,s,psi = calc_psi_pop2011()
>>> maxshear = 10.0; maxpsi = 42.0
>>>
```

Another plotting example

```
>>> r,z,s,psi = calc_psi_pop2011()
>>> maxshear = 10.0; maxpsi = 42.0
>>> V = np.linspace(-maxshear,
  0.0, 75)
>>>
```

Another plotting example

```
>>> r,z,s,psi = calc_psi_pop2011()
>>> maxshear = 10.0; maxpsi = 42.0
>>> V = np.linspace(-maxshear,
  0.0, 75)
>>> import matplotlib as mpl
>>> mpl.rcParams['ytick.labelsize'] = 6
>>> mpl.rcParams['xtick.labelsize'] = 6

>>> fig = figure(
  figsize = (3.375,2), dpi = 600)
>>>
```

Another plotting example

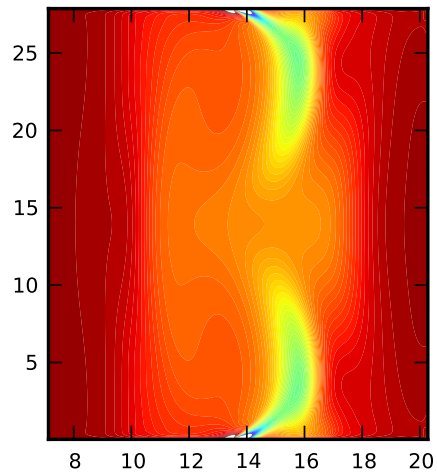
```
>>> r,z,s,psi = calc_psi_pop2011()
>>> maxshear = 10.0; maxpsi = 42.0
>>> V = np.linspace(-maxshear,
  0.0, 75)
>>> import matplotlib as mpl
>>> mpl.rcParams['ytick.labelsize'] = 6
>>> mpl.rcParams['xtick.labelsize'] = 6

>>> fig = figure(
  figsize = (3.375,2), dpi = 600)
>>> fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
>>>
```

Another plotting example

```
>>> r,z,s,psi = calc_psi_pop2011()
>>> maxshear = 10.0; maxpsi = 42.0
>>> V = np.linspace(-maxshear,
  0.0, 75)
>>> import matplotlib as mpl
>>> mpl.rcParams['ytick.labelsize'] = 6
>>> mpl.rcParams['xtick.labelsize'] = 6

>>> fig = figure(
  figsize = (3.375,2), dpi = 600)
>>> fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
>>> a = fig.add_subplot(1,2,1)
>>>
```

Another plotting example

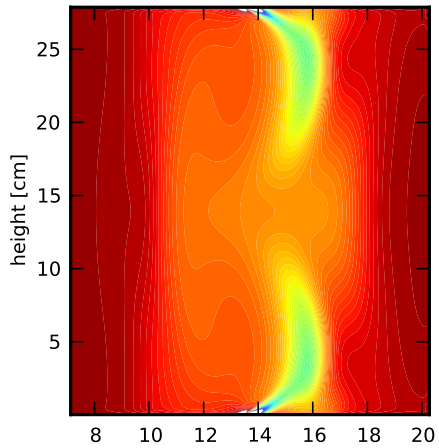
```
>>> r,z,s,psi = calc_psi_pop2011()
>>> maxshear = 10.0; maxpsi = 42.0
>>> V = np.linspace(-maxshear,
  0.0, 75)
>>> import matplotlib as mpl
>>> mpl.rcParams['ytick.labelsize'] = 6
>>> mpl.rcParams['xtick.labelsize'] = 6

>>> fig = figure(
  figsize = (3.375,2), dpi = 600)
>>> fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
>>> a = fig.add_subplot(1,2,1)
>>> plt.contourf(r, z, s, V)
```




```
>>>
```

Another plotting example

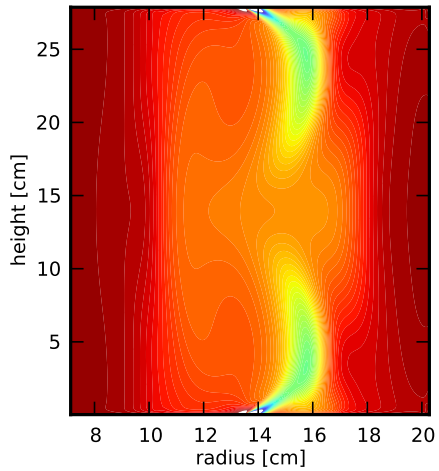
```
>>> r,z,s,psi = calc_psi_pop2011()
>>> maxshear = 10.0; maxpsi = 42.0
>>> V = np.linspace(-maxshear,
  0.0, 75)
>>> import matplotlib as mpl
>>> mpl.rcParams['ytick.labelsize'] = 6
>>> mpl.rcParams['xtick.labelsize'] = 6

>>> fig = figure(
  figsize = (3.375,2), dpi = 600)
>>> fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
>>> a = fig.add_subplot(1,2,1)
>>> plt.contourf(r, z, s, V)
```




```
>>> plt.ylabel('height [cm]',
  fontsize = 6,
  horizontalalignment='center')

>>>
```

Another plotting example

```
>>> r,z,s,psi = calc_psi_pop2011()
>>> maxshear = 10.0; maxpsi = 42.0
>>> V = np.linspace(-maxshear,
  0.0, 75)
>>> import matplotlib as mpl
>>> mpl.rcParams['ytick.labelsize'] = 6
>>> mpl.rcParams['xtick.labelsize'] = 6

>>> fig = figure(
  figsize = (3.375,2), dpi = 600)
>>> fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
>>> a = fig.add_subplot(1,2,1)
>>> plt.contourf(r, z, s, V)
```

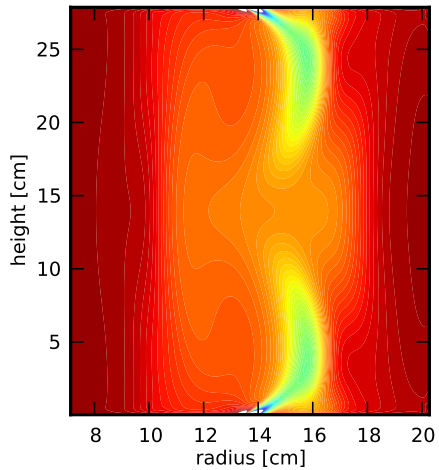



```
>>> plt.ylabel('height [cm]',
  fontsize = 6,
  horizontalalignment='center')

>>> plt.xlabel('radius [cm]',
  fontsize = 6,
  verticalalignment='center')
```

SciNet compute • calcul CANADA

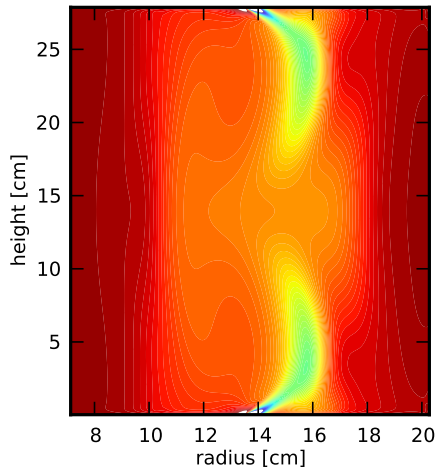
Another plotting example, continued

```
>>>
```



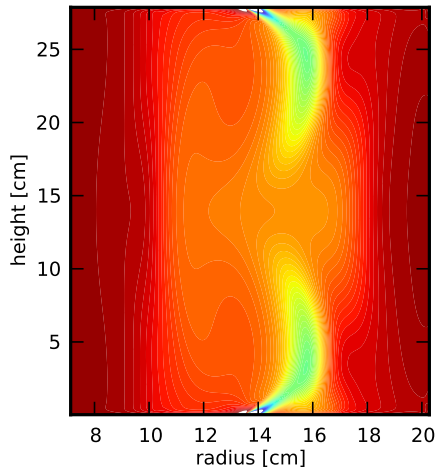
Another plotting example, continued

```
>>> V2 = np.linspace(-maxpsi,
  maxpsi,22)

>>>
```



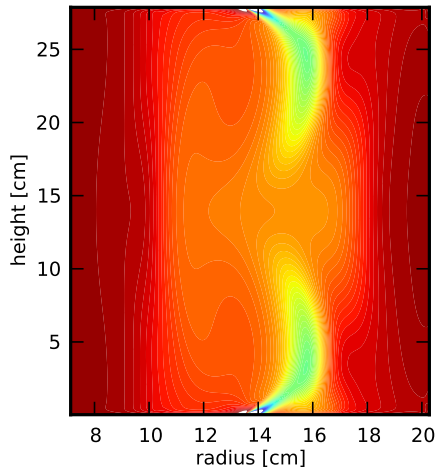
Another plotting example, continued

```
>>> V2 = np.linspace(-maxpsi,
  maxpsi,22)

>>> styles = []
>>>
```



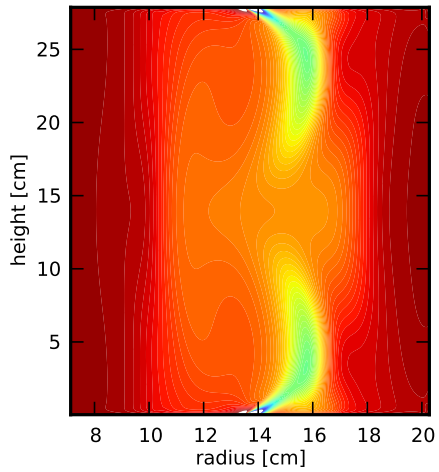
Another plotting example, continued

```
>>> V2 = np.linspace(-maxpsi,
  maxpsi,22)

>>> styles = []
>>> for i in range(11):
...   styles.append('dashed')

>>> for i in range(11):
...   styles.append('solid')

>>>
```



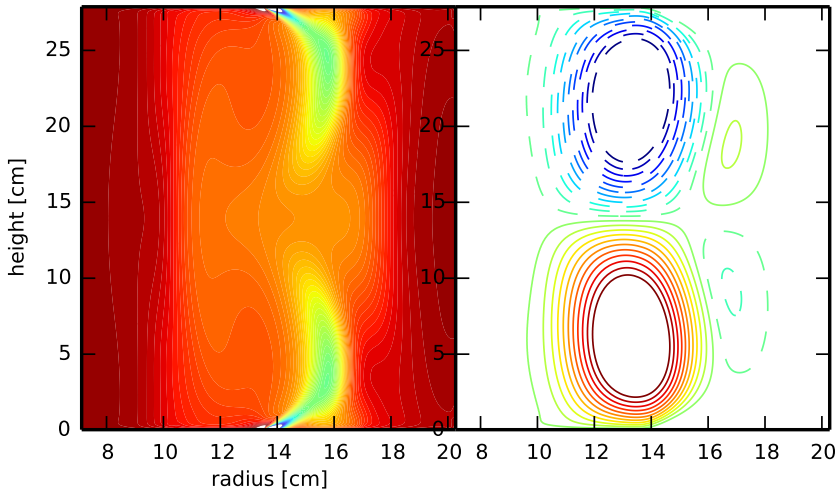
Another plotting example, continued

```
>>> V2 = np.linspace(-maxpsi,
  maxpsi,22)

>>> styles = []
>>> for i in range(11):
...   styles.append('dashed')

>>> for i in range(11):
...   styles.append('solid')

>>> a2 = fig.add_subplot(1,2,2)

>>>
```



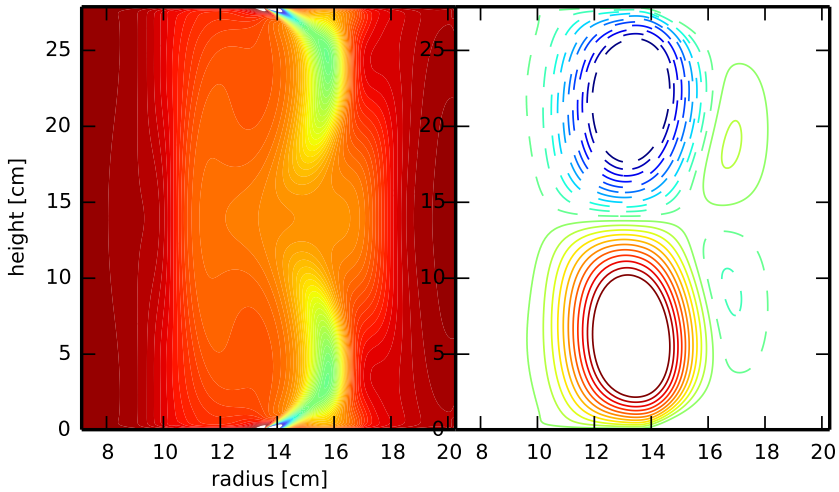
Another plotting example, continued

```
>>> V2 = np.linspace(-maxpsi,
  maxpsi,22)

>>> styles = []
>>> for i in range(11):
...   styles.append('dashed')

>>> for i in range(11):
...   styles.append('solid')

>>> a2 = fig.add_subplot(1,2,2)

>>> contour(r, z, psi, V2,
  linewidths = 0.5, linestyles = styles)

>>>
```



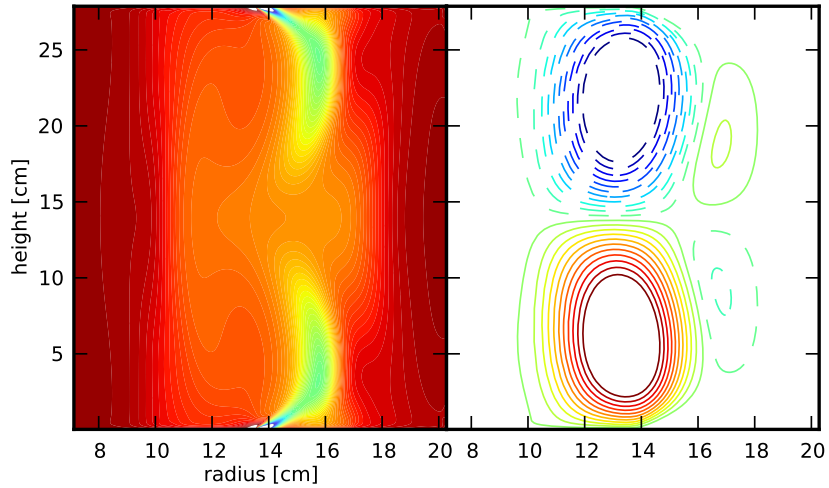
Another plotting example, continued

```
>>> V2 = np.linspace(-maxpsi,
  maxpsi,22)

>>> styles = []
>>> for i in range(11):
...   styles.append('dashed')

>>> for i in range(11):
...   styles.append('solid')

>>> a2 = fig.add_subplot(1,2,2)

>>> contour(r, z, psi, V2,
  linewidths = 0.5, linestyles = styles)

>>> a2.set_yticklabels([])
>>>
```



Another plotting example, continued

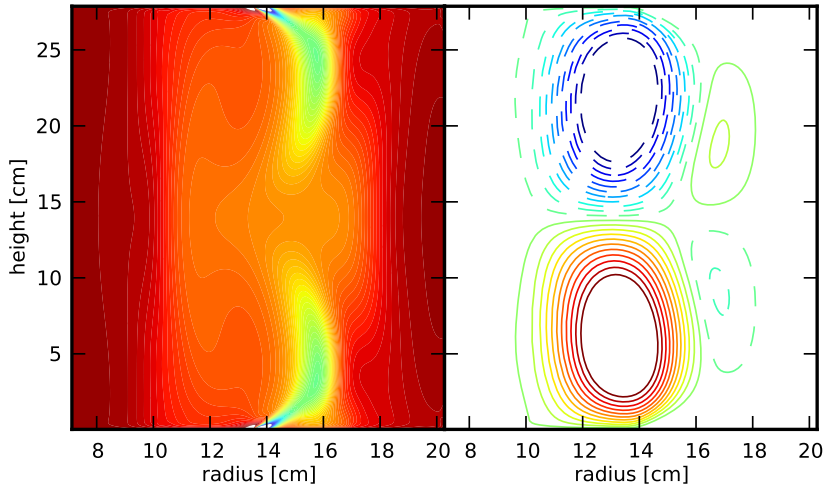
```
>>> V2 = np.linspace(-maxpsi,
  maxpsi,22)

>>> styles = []
>>> for i in range(11):
...   styles.append('dashed')

>>> for i in range(11):
...   styles.append('solid')

>>> a2 = fig.add_subplot(1,2,2)

>>> contour(r, z, psi, V2,
  linewidths = 0.5, linestyles = styles)

>>> a2.set_yticklabels([])
>>> plt.show()
```




```
>>>
```

Another plotting example, continued

```
>>> V2 = np.linspace(-maxpsi,
  maxpsi,22)

>>> styles = []
>>> for i in range(11):
...   styles.append('dashed')

>>> for i in range(11):
...   styles.append('solid')

>>> a2 = fig.add_subplot(1,2,2)

>>> contour(r, z, psi, V2,
  linewidths = 0.5, linestyles = styles)

>>> a2.set_yticklabels([])
>>> plt.show()
```




```
>>> xlabel('radius [cm]',
  fontsize = 6,
  verticalalignment='center')
```

Another final product