Intro to Research Computing with Python: Visualization

Erik Spence

SciNet HPC Consortium

20 November 2014

SciNet compute • calcul CANADA

Today's class

Today 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.
- Animations.

SciNet compute • calcul CANADA

Maplotlib's pyplot

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

- matplotlib is an add-on package to Python (comes with Canopy).
- 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.

SciNet compute • calcul CANADA

Plotting in Python

By default, plotting in Python is a little wonky. Note that the following will NOT work in Canopy.

```
ejspence@mycomp ~>
ejspence@mycomp ~> python
>>>
>>> import matplotlib.pyplot as plt
>>> plt.figure()
<matplotlib.figure.Figure object at 0x7fdbfec5ff90>
>>>
>>> plt.show()
>>>
```

As you can see, the default behaviour of matplotlib is to delay drawing the figure until it is told to do so.

Plotting in Python, continued

Using ipython instead of python will get you around this, but only if you use the --pylab flag.

```
ejspence@mycomp ~>
ejspence@mycomp ~> ipython
In [1]:
In [2]: import matplotlib.pyplot as plt
In [3]: plt.figure()
<matplotlib.figure.Figure object at 0x7fdbfec5ff90>
In [4]: exit
ejspence@mycomp ~>
ejspence@mycomp ~> ipython --pylab
In [1]: import matplotlib.pyplot as plt
In [2]: plt.figure()
```

Interactive mode will only work with pyplot commands, however, not matplotlib objects.

Plotting

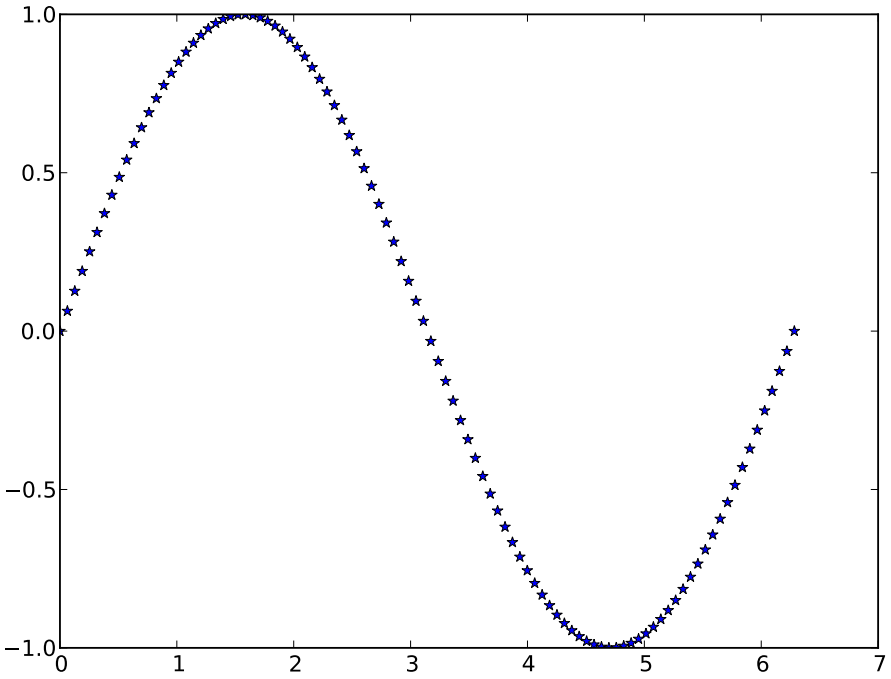
```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]:
```

Plotting

```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]: x =
  np.linspace(0, 2 * np.pi, 100)

In [4]:
```

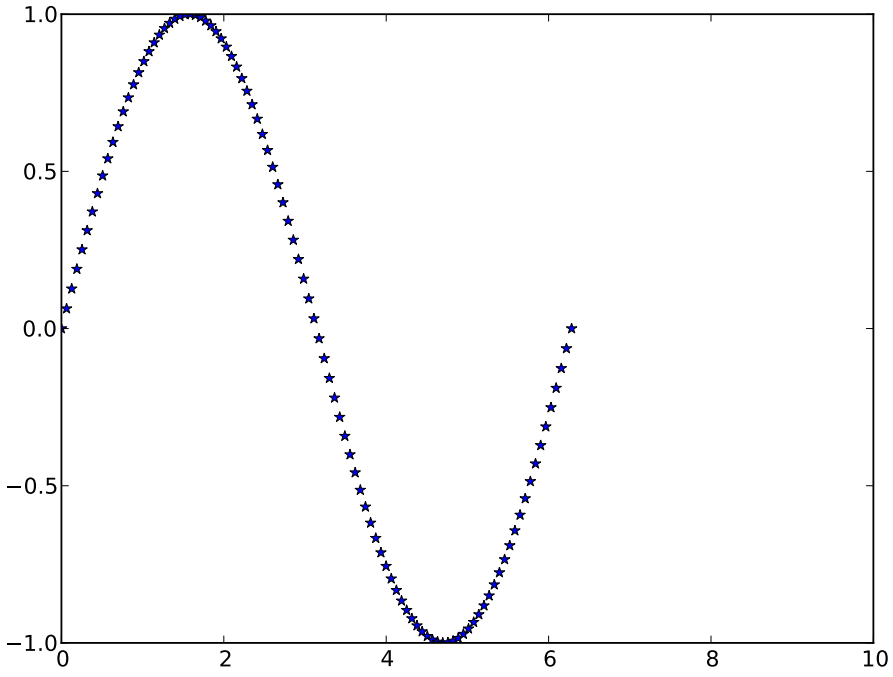
Plotting

```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]: x =
  np.linspace(0, 2 * np.pi, 100)

In [4]: plt.plot(x, np.sin(x), '*')

In [5]:
```



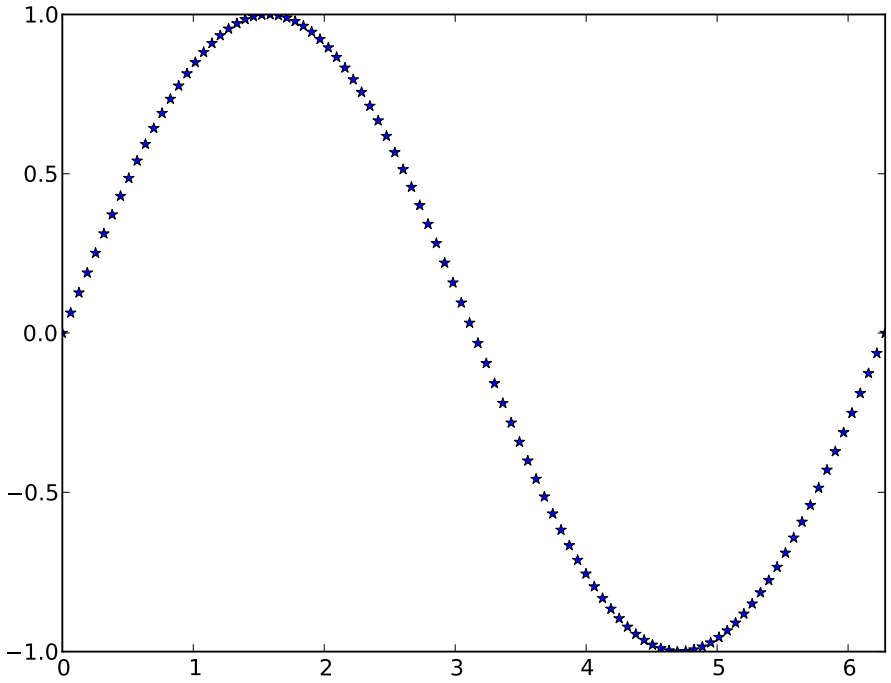
Plotting

```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]: x =
  np.linspace(0, 2 * np.pi, 100)

In [4]: plt.plot(x, np.sin(x), '*')

In [5]: plt.xlim(0, 10)
Out[5]: (0, 10)

In [6]:
```



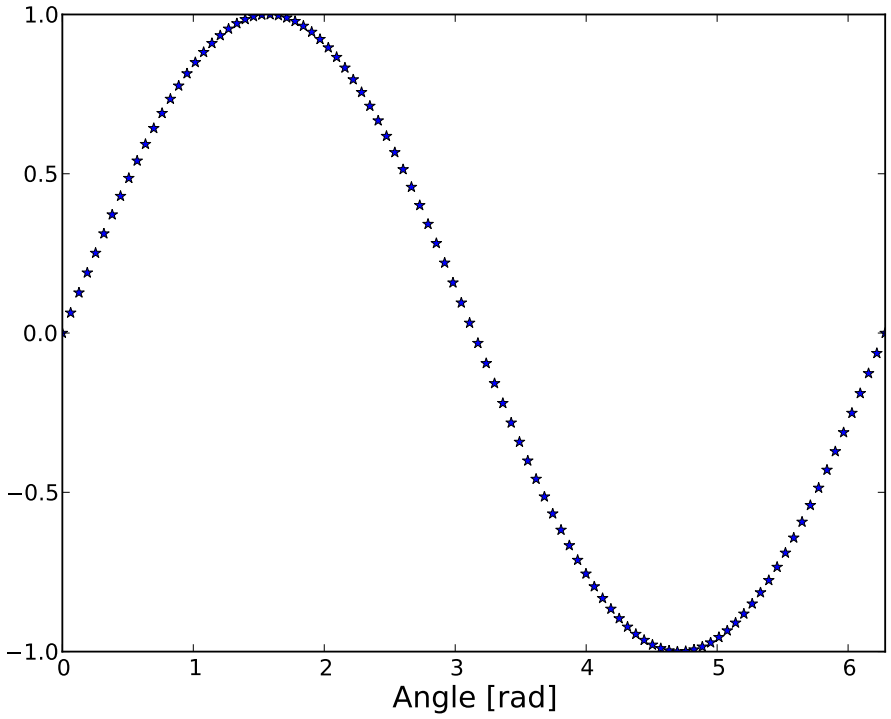
Plotting

```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]: x =
  np.linspace(0, 2 * np.pi, 100)

In [4]: plt.plot(x, np.sin(x), '*')

In [5]: plt.xlim(0, 10)
Out[5]: (0, 10)

In [6]: plt.xlim(0, 2 * np.pi)
Out[6]: (0, 6.2831853071795862)

In [7]:
```



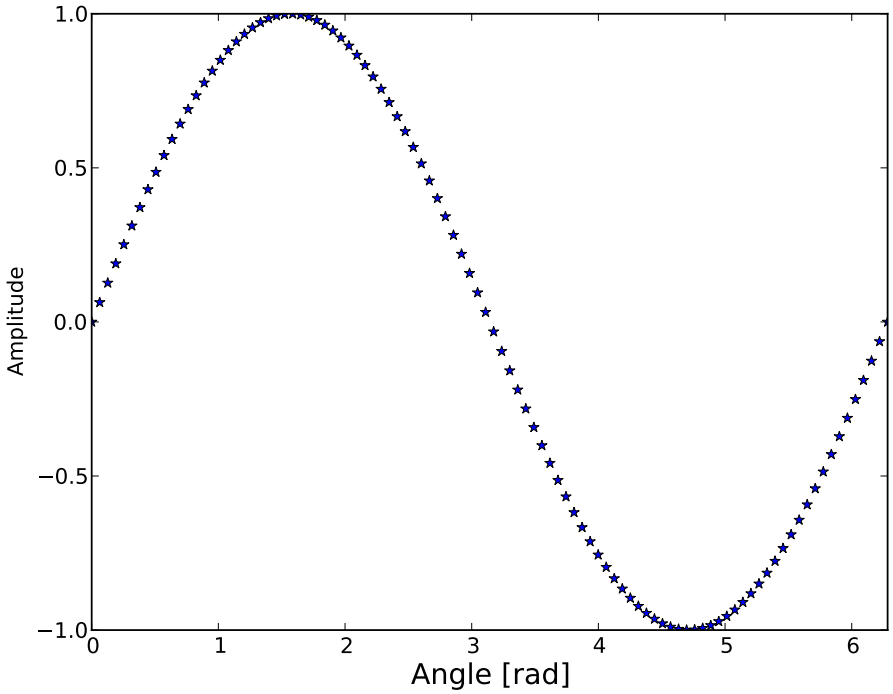
Plotting

```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]: x =
  np.linspace(0, 2 * np.pi, 100)

In [4]: plt.plot(x, np.sin(x), '*')

In [5]: plt.xlim(0, 10)
Out[5]: (0, 10)

In [6]: plt.xlim(0, 2 * np.pi)
Out[6]: (0, 6.2831853071795862)

In [7]: plt.xlabel('Angle [rad]',
  fontsize = 16)

In [8]:
```



Plotting

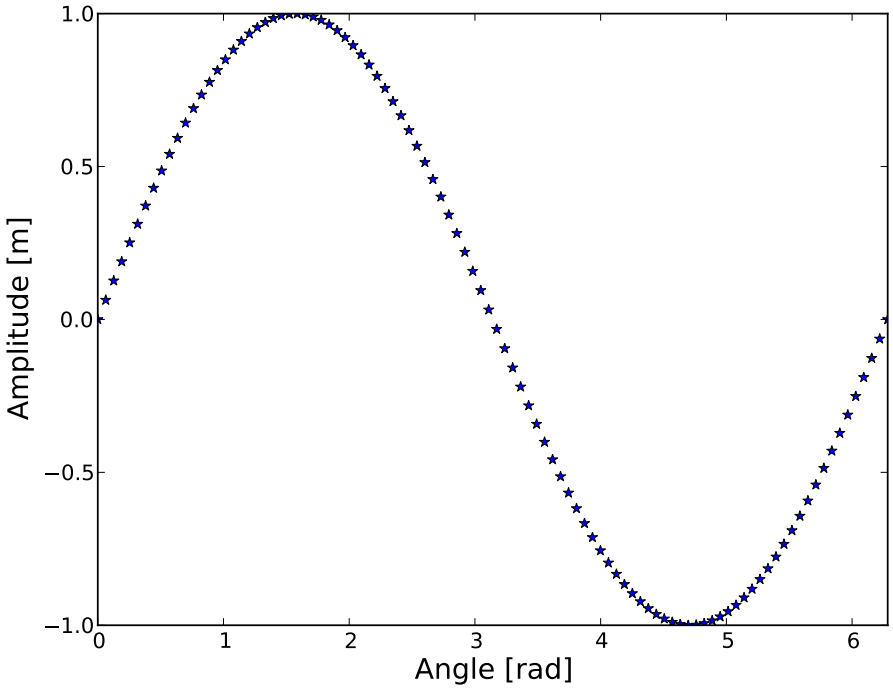
```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]: x =
  np.linspace(0, 2 * np.pi, 100)

In [4]: plt.plot(x, np.sin(x), '*')

In [5]: plt.xlim(0, 10)
Out[5]: (0, 10)

In [6]: plt.xlim(0, 2 * np.pi)
Out[6]: (0, 6.2831853071795862)

In [7]: plt.xlabel('Angle [rad]',
  fontsize = 16)

In [8]: plt.ylabel('Amplitude')
```




```
In [9]:
```

Plotting

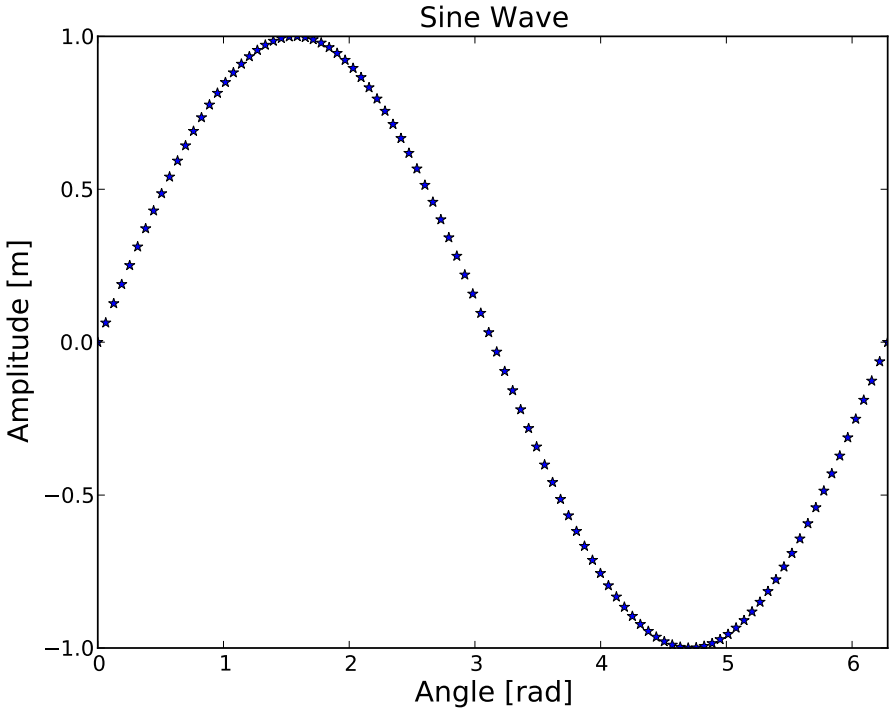
```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]: x =
  np.linspace(0, 2 * np.pi, 100)

In [4]: plt.plot(x, np.sin(x), '*')

In [5]: plt.xlim(0, 10)
Out[5]: (0, 10)

In [6]: plt.xlim(0, 2 * np.pi)
Out[6]: (0, 6.2831853071795862)

In [7]: plt.xlabel('Angle [rad]',
  fontsize = 16)

In [8]: plt.ylabel('Amplitude')
```




```
In [9]: plt.ylabel('Amplitude [m]',
  fontsize = 16)

In [10]:
```

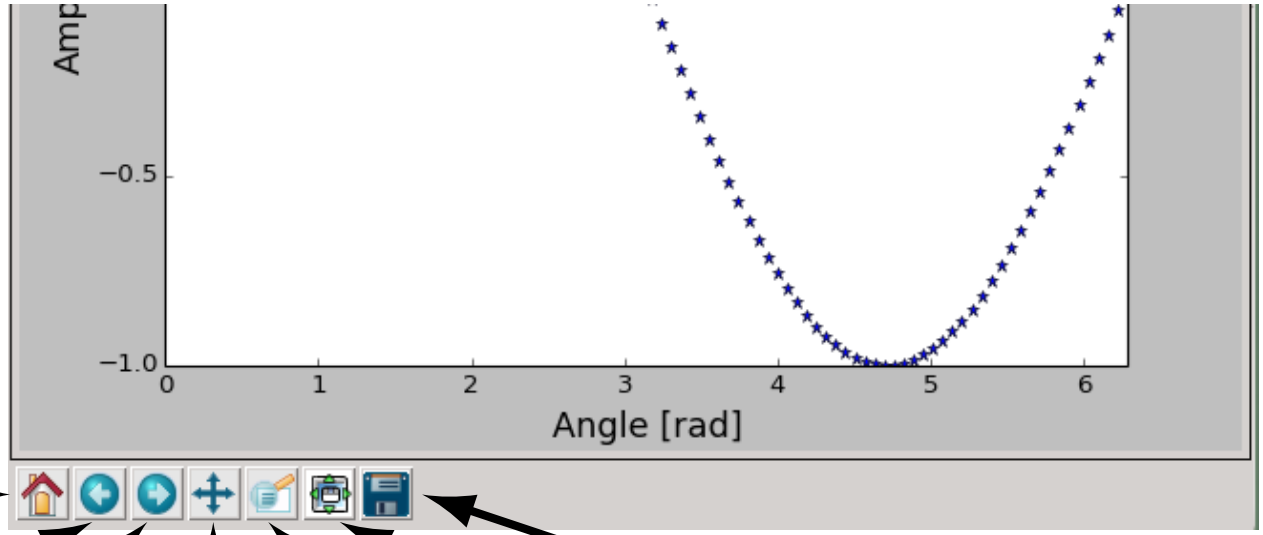
Plotting

```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot
  as plt

In [3]: x =
  np.linspace(0, 2 * np.pi, 100)

In [4]: plt.plot(x, np.sin(x), '*')

In [5]: plt.xlim(0, 10)
Out[5]: (0, 10)

In [6]: plt.xlim(0, 2 * np.pi)
Out[6]: (0, 6.2831853071795862)

In [7]: plt.xlabel('Angle [rad]',
  fontsize = 16)

In [8]: plt.ylabel('Amplitude')
```




```
In [9]: plt.ylabel('Amplitude [m]',
  fontsize = 16)

In [10]: plt.title('Sine Wave',
  fontsize = 16)
```

What are those buttons?



Return to where you started.

Go back a step.

Go forward a step.

Grab the plot and move it around.

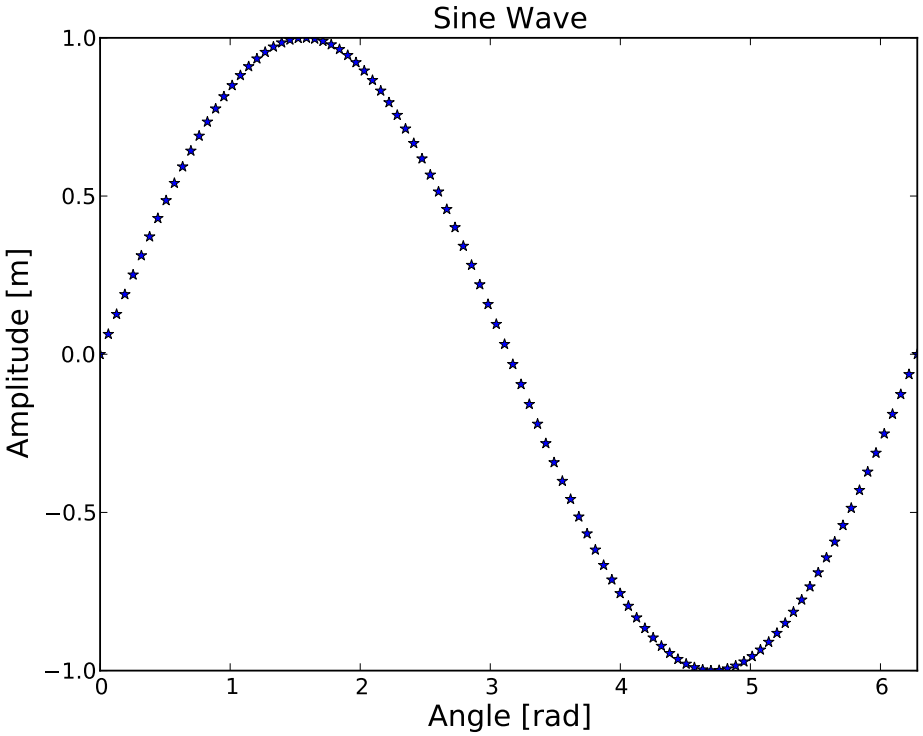
Zoom in on a section.

Adjust spacing between plots.

Save the figure.

Playing with subplots

```
In [11]:
```

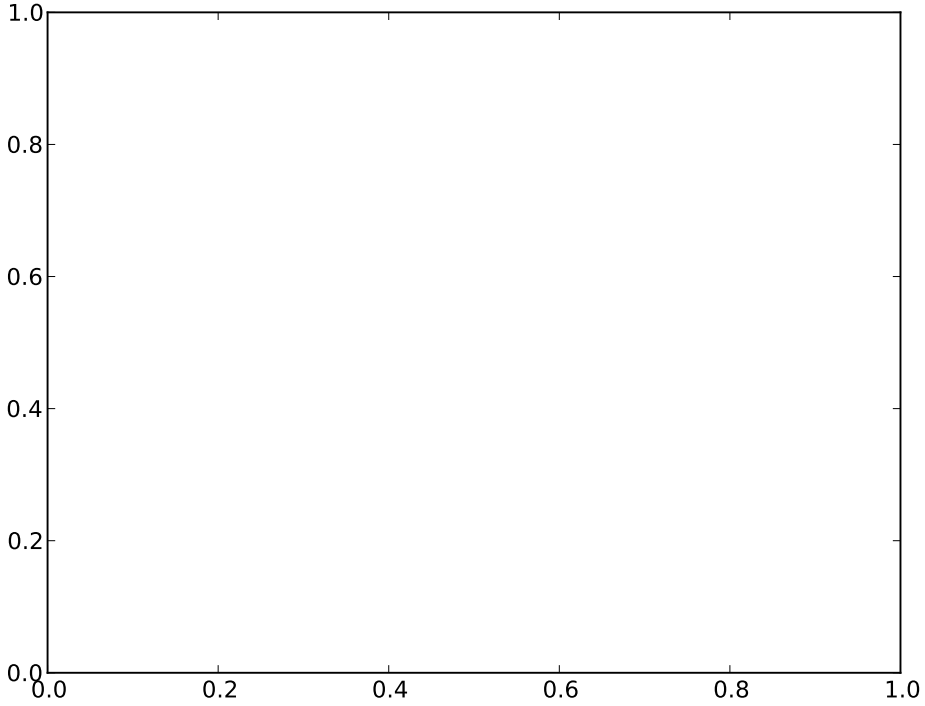


Playing with subplots

```
In [11]: plt.clf()

In [12]:
```

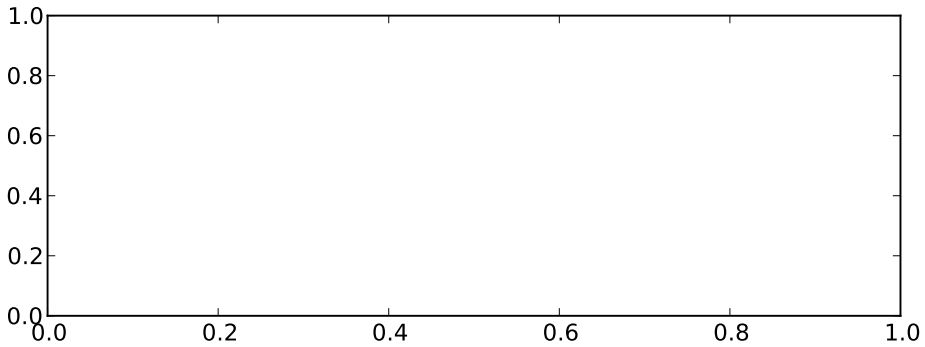
Playing with subplots

```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]:
```



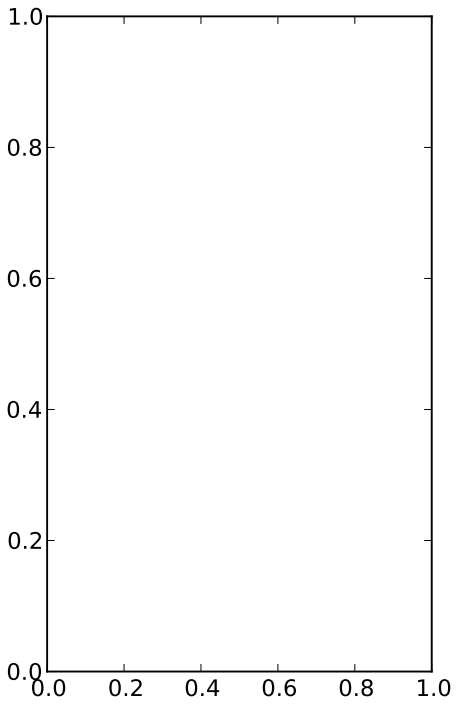
Playing with subplots

```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]:
```



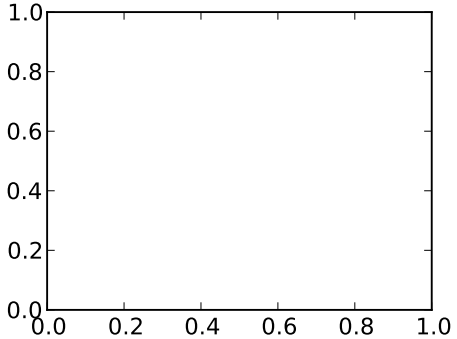
Playing with subplots

```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]:
```



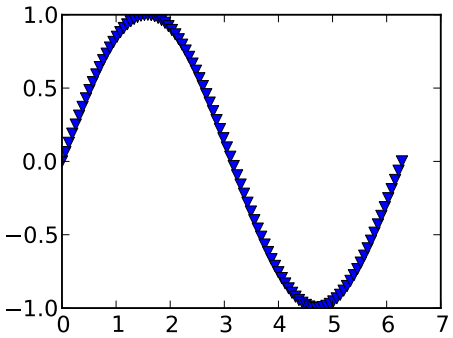
Playing with subplots

```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]: plt.subplot(2,2,1)

In [16]:
```



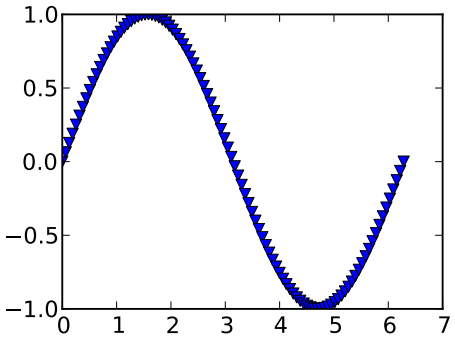
Playing with subplots

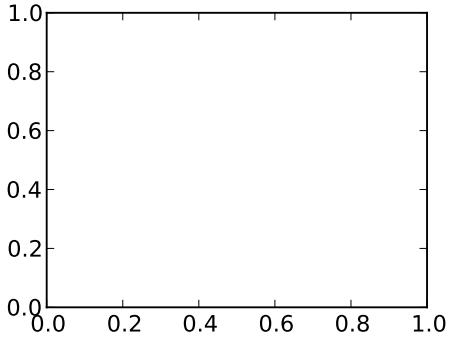
```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]: plt.subplot(2,2,1)

In [16]: plt.plot(x, np.sin(x), 'v')

In [17]:
```



Playing with subplots

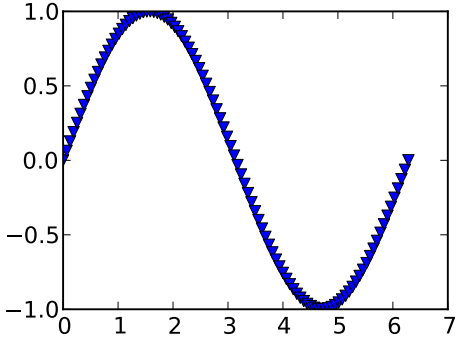
```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]: plt.subplot(2,2,1)

In [16]: plt.plot(x, np.sin(x), 'v')

In [17]: plt.subplot(2,2,4)

In [18]:
```

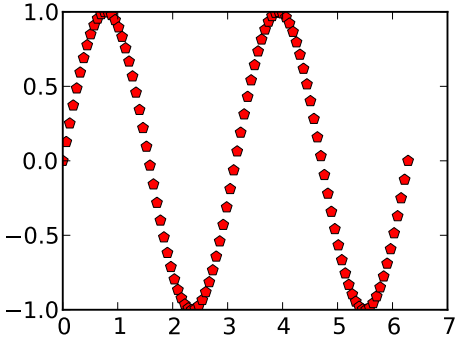




Playing with subplots

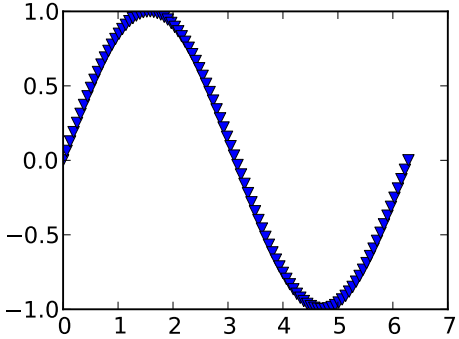
```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]: plt.subplot(2,2,1)

In [16]: plt.plot(x, np.sin(x), 'v')

In [17]: plt.subplot(2,2,4)

In [18]: plt.plot(x, np.sin(2 * x),
  'rp')

In [19]:
```

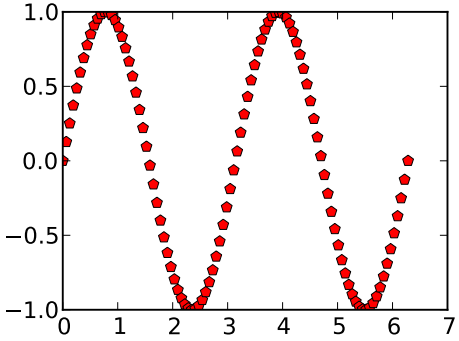




Playing with subplots

```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]: plt.subplot(2,2,1)

In [16]: plt.plot(x, np.sin(x), 'v')

In [17]: plt.subplot(2,2,4)

In [18]: plt.plot(x, np.sin(2 * x),
  'rp')

In [19]: plt.subplot(2,2,1)
```

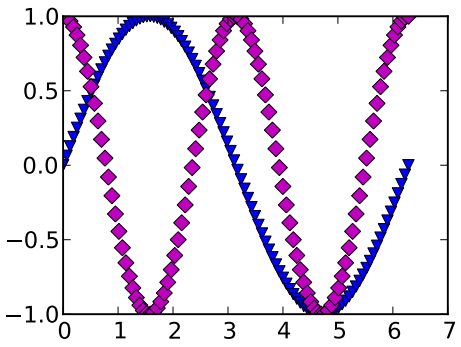


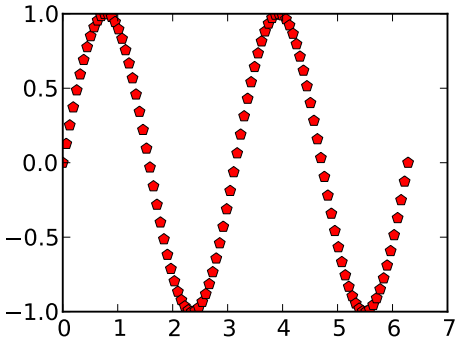



```
In [20]:
```

Playing with subplots

```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]: plt.subplot(2,2,1)

In [16]: plt.plot(x, np.sin(x), 'v')

In [17]: plt.subplot(2,2,4)

In [18]: plt.plot(x, np.sin(2 * x),
  'rp')

In [19]: plt.subplot(2,2,1)
```

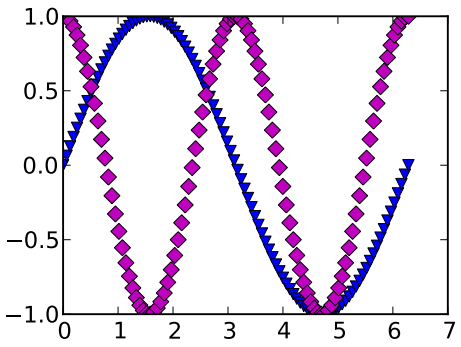


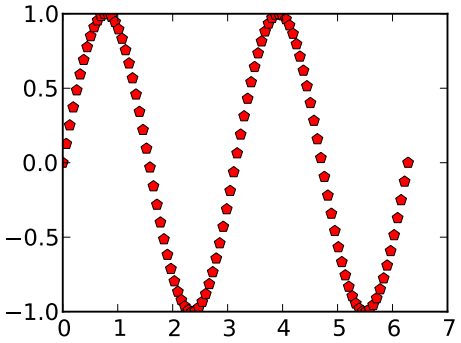



```
In [20]: plt.plot(x, np.cos(2 * x),
  'mD')

In [21]:
```

Playing with subplots

```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]: plt.subplot(2,2,1)

In [16]: plt.plot(x, np.sin(x), 'v')

In [17]: plt.subplot(2,2,4)

In [18]: plt.plot(x, np.sin(2 * x),
  'rp')

In [19]: plt.subplot(2,2,1)
```

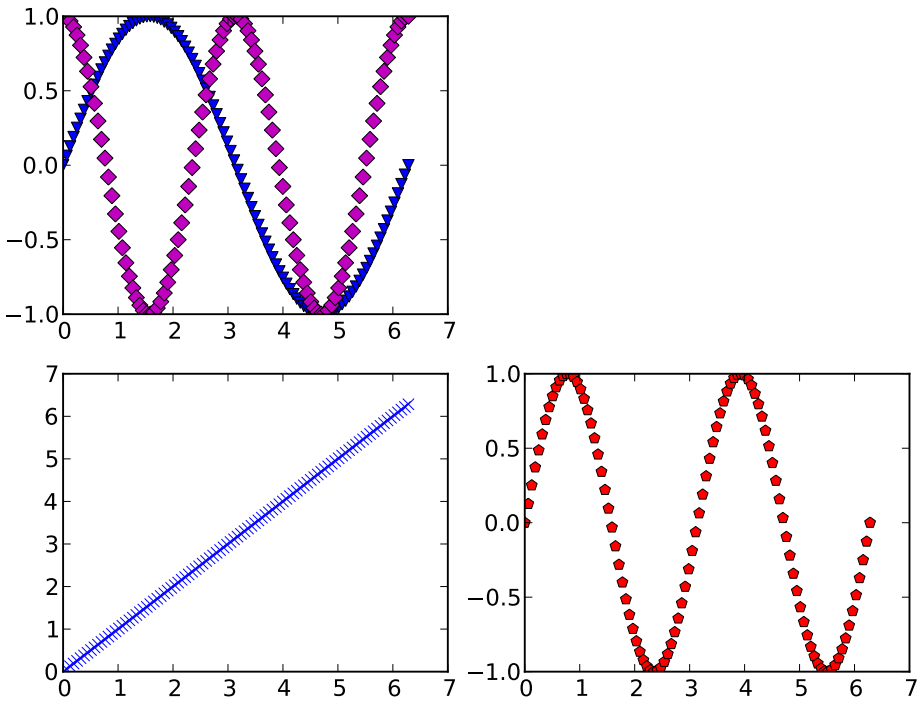





```
In [20]: plt.plot(x, np.cos(2 * x),
  'mD')

In [21]: plt.subplot(2,2,3)

In [22]:
```

Playing with subplots

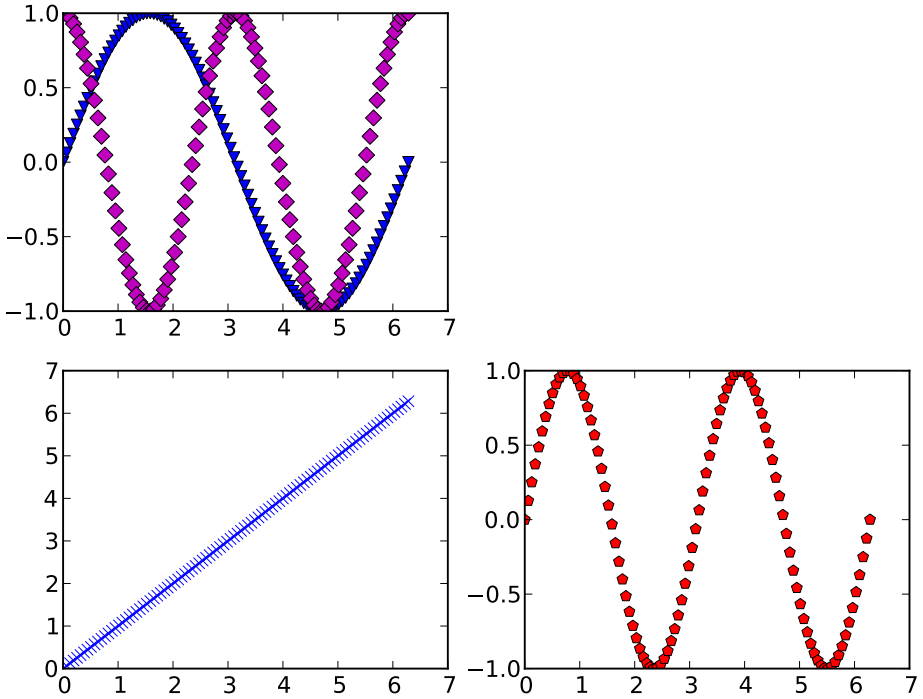
```
In [11]: plt.clf()

In [12]: plt.subplot(1,1,1)

In [13]: plt.subplot(2,1,1)

In [14]: plt.subplot(1,2,1)

In [15]: plt.subplot(2,2,1)

In [16]: plt.plot(x, np.sin(x), 'v')

In [17]: plt.subplot(2,2,4)

In [18]: plt.plot(x, np.sin(2 * x),
  'rp')

In [19]: plt.subplot(2,2,1)
```




```
In [20]: plt.plot(x, np.cos(2 * x),
  'mD')

In [21]: plt.subplot(2,2,3)

In [22]: plt.plot(x, x, 'x')
```

Using error bars

```
In [23]:
```



Using error bars

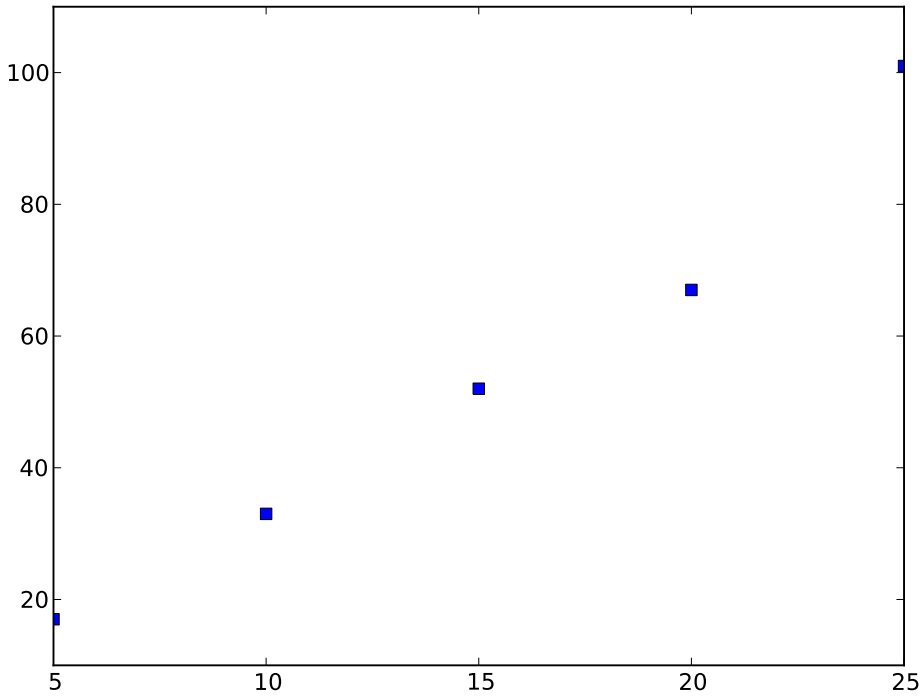
```
In [23]: plt.clf()

In [24]:
```

Using error bars

```
In [23]: plt.clf()

In [24]: o2 = np.arange(5, 26, 5)
In [25]: o1 = [17, 33, 52, 67, 101]

In [26]:
```

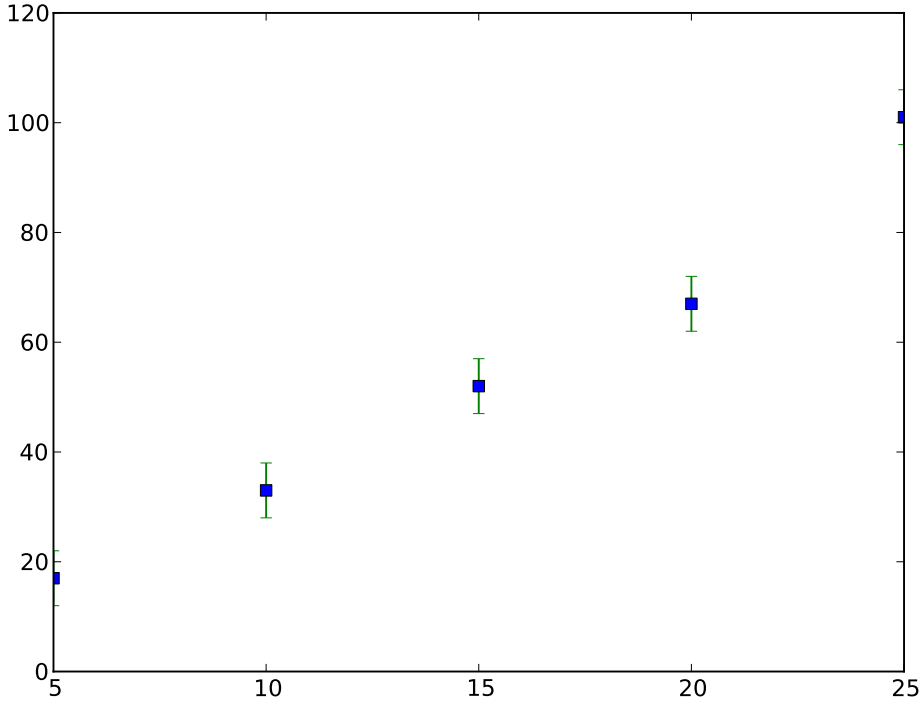
Using error bars

```
In [23]: plt.clf()

In [24]: o2 = np.arange(5, 26, 5)
In [25]: o1 = [17, 33, 52, 67, 101]

In [26]: plt.plot(o2, o1, 's',
  label = 'Raw Data')

In [27]:
```



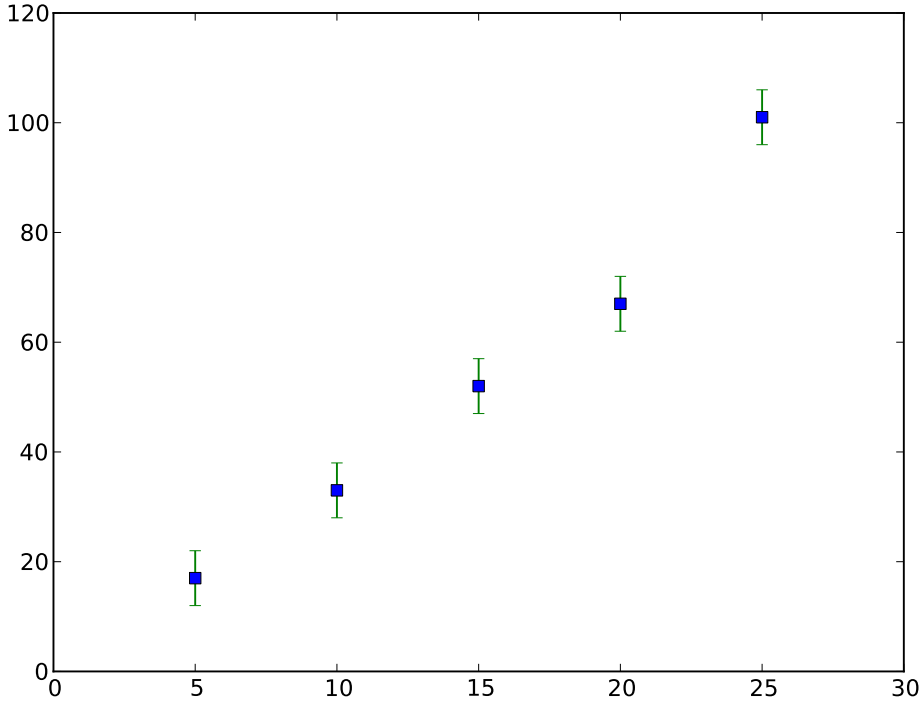
Using error bars

```
In [23]: plt.clf()

In [24]: o2 = np.arange(5, 26, 5)
In [25]: o1 = [17, 33, 52, 67, 101]

In [26]: plt.plot(o2, o1, 's',
  label = 'Raw Data')

In [27]: plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

In [28]:
```



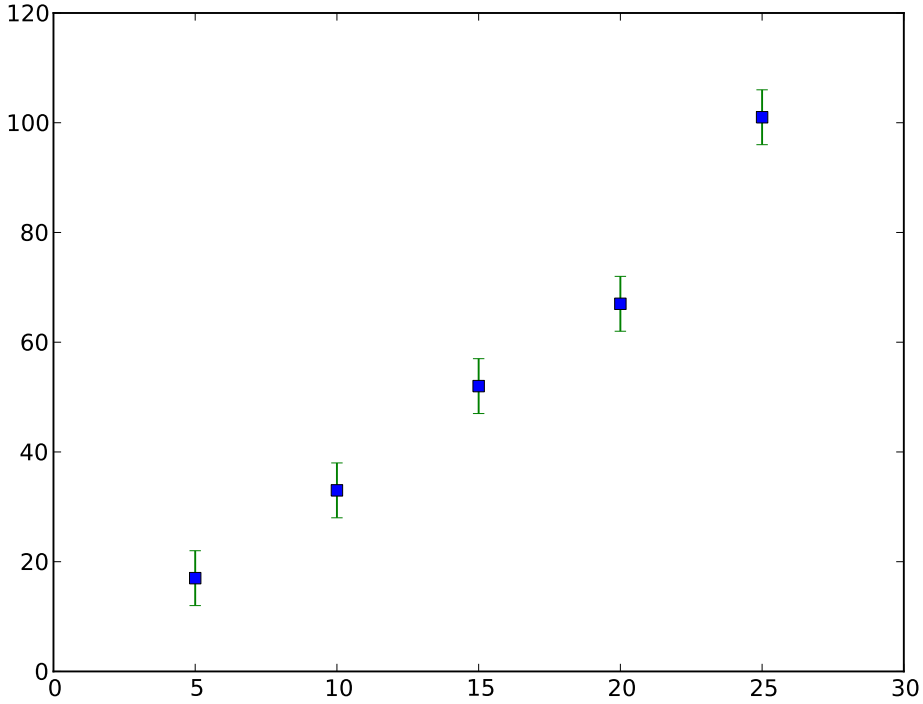
Using error bars

```
In [23]: plt.clf()

In [24]: o2 = np.arange(5, 26, 5)
In [25]: o1 = [17, 33, 52, 67, 101]

In [26]: plt.plot(o2, o1, 's',
  label = 'Raw Data')

In [27]: plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

In [28]: plt.xlim(0, 30)

In [29]:
```



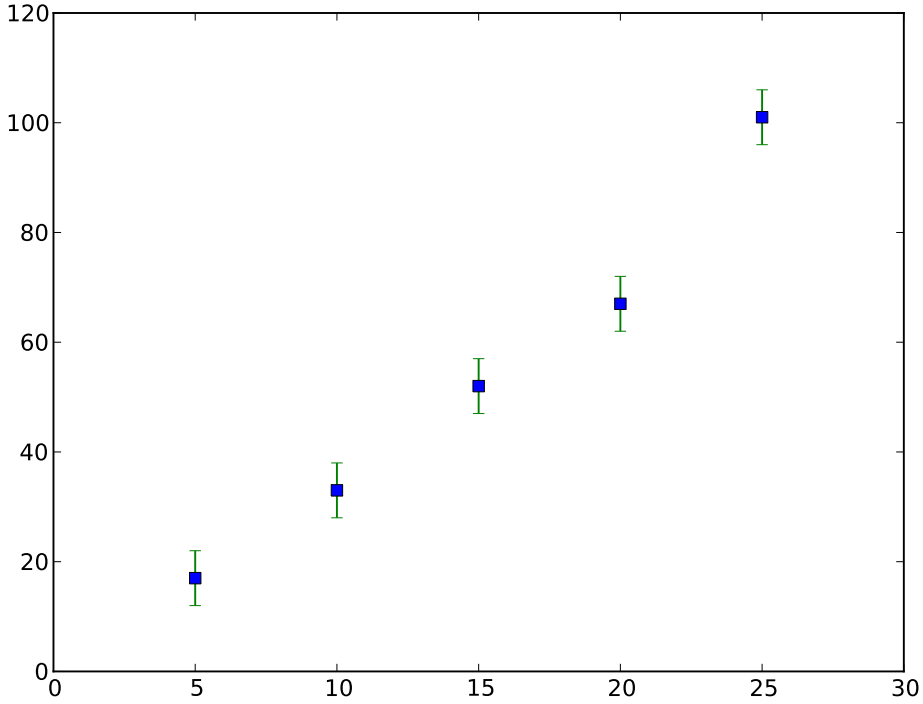
Using error bars

```
In [23]: plt.clf()

In [24]: o2 = np.arange(5, 26, 5)
In [25]: o1 = [17, 33, 52, 67, 101]

In [26]: plt.plot(o2, o1, 's',
  label = 'Raw Data')

In [27]: plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

In [28]: plt.xlim(0, 30)

In [29]: fit = np.polyfit(o2, o1, 1)

In [30]:
```



Using error bars

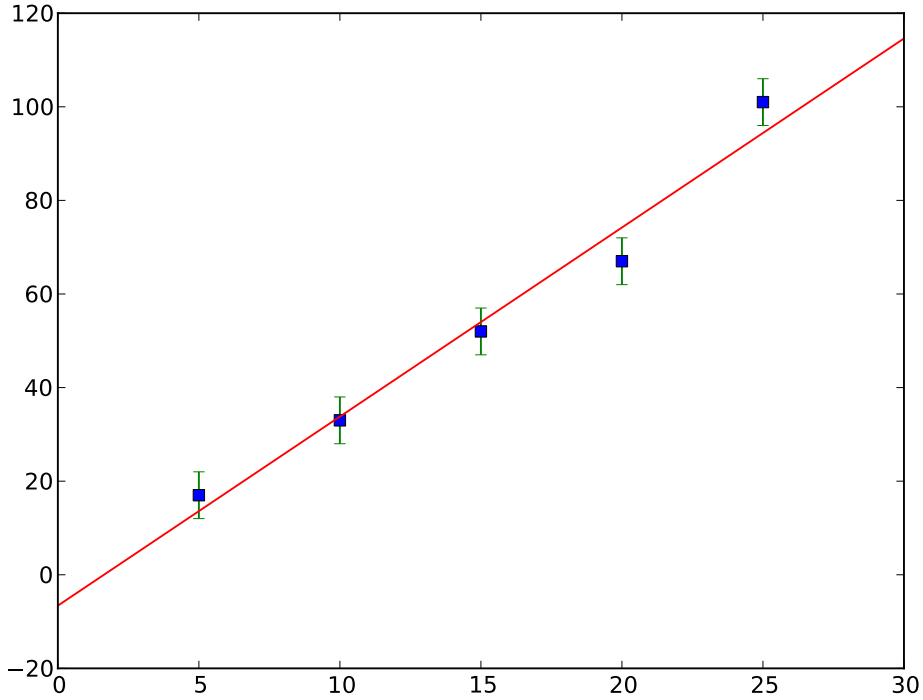
```
In [23]: plt.clf()

In [24]: o2 = np.arange(5, 26, 5)
In [25]: o1 = [17, 33, 52, 67, 101]

In [26]: plt.plot(o2, o1, 's',
  label = 'Raw Data')

In [27]: plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

In [28]: plt.xlim(0, 30)

In [29]: fit = np.polyfit(o2, o1, 1)

In [30]: x = np.linspace(0, 30, 100)
```




```
In [31]:
```

Using error bars

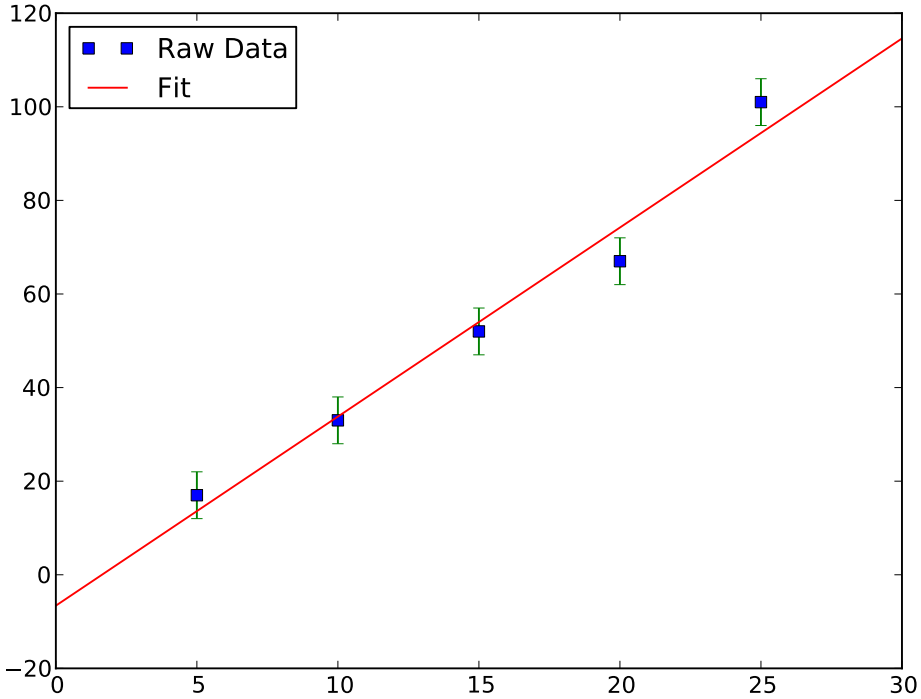
```
In [23]: plt.clf()

In [24]: o2 = np.arange(5, 26, 5)
In [25]: o1 = [17, 33, 52, 67, 101]

In [26]: plt.plot(o2, o1, 's',
  label = 'Raw Data')

In [27]: plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

In [28]: plt.xlim(0, 30)

In [29]: fit = np.polyfit(o2, o1, 1)

In [30]: x = np.linspace(0, 30, 100)
```




```
In [31]: plt.plot(x, np.polyval(fit,x),
  label = 'Fit')

In [32]:
```

Using error bars

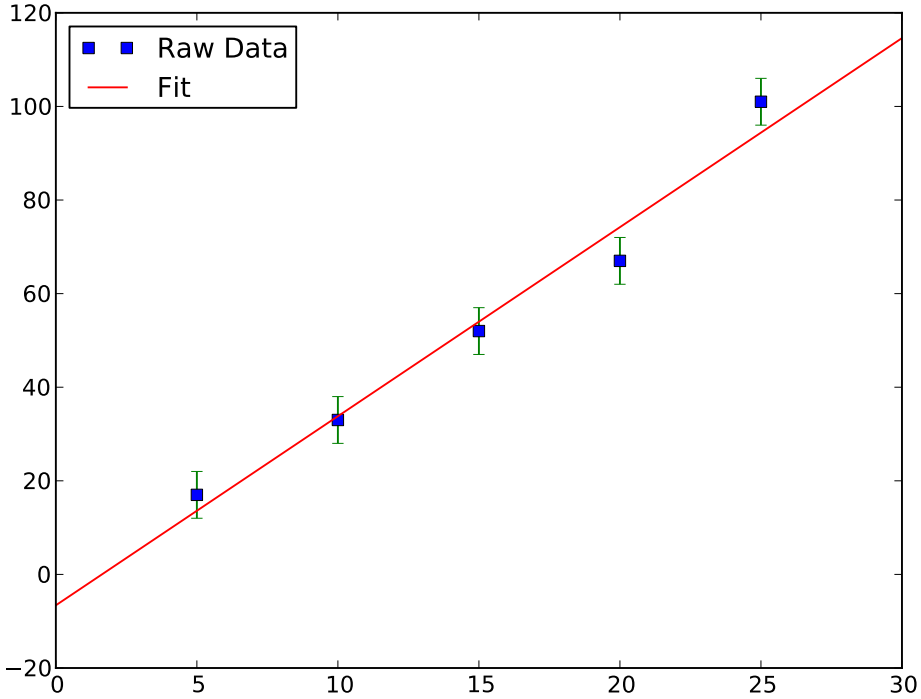
```
In [23]: plt.clf()

In [24]: o2 = np.arange(5, 26, 5)
In [25]: o1 = [17, 33, 52, 67, 101]

In [26]: plt.plot(o2, o1, 's',
  label = 'Raw Data')

In [27]: plt.errorbar(o2, o1,
  yerr = 5, fmt = None)

In [28]: plt.xlim(0, 30)

In [29]: fit = np.polyfit(o2, o1, 1)

In [30]: x = np.linspace(0, 30, 100)
```




```
In [31]: plt.plot(x, np.polyval(fit,x),
  label = 'Fit')

In [32]: plt.legend(loc = 'best')
```

2D plotting

```
In [33]:
```



2D plotting

```
In [33]: plt.clear()

In [34]:
```

2D plotting

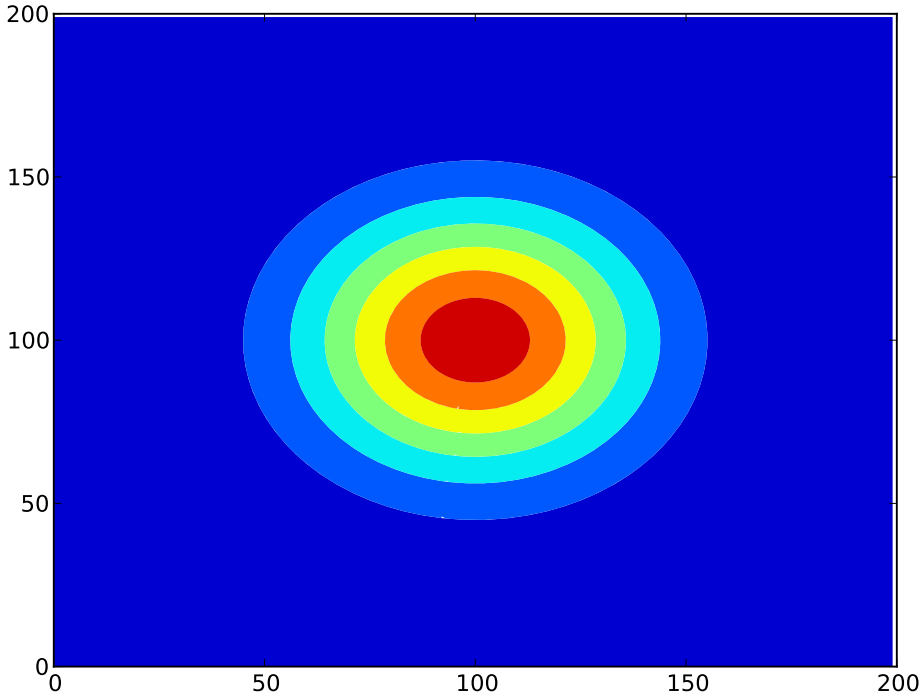
```
In [33]: plt.clear()

In [34]: x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

In [35]:
```

2D plotting

```
In [33]: plt.clear()

In [34]: x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

In [35]: g =
  np.exp(-(x**2 + y**2) / 16)

In [36]:
```

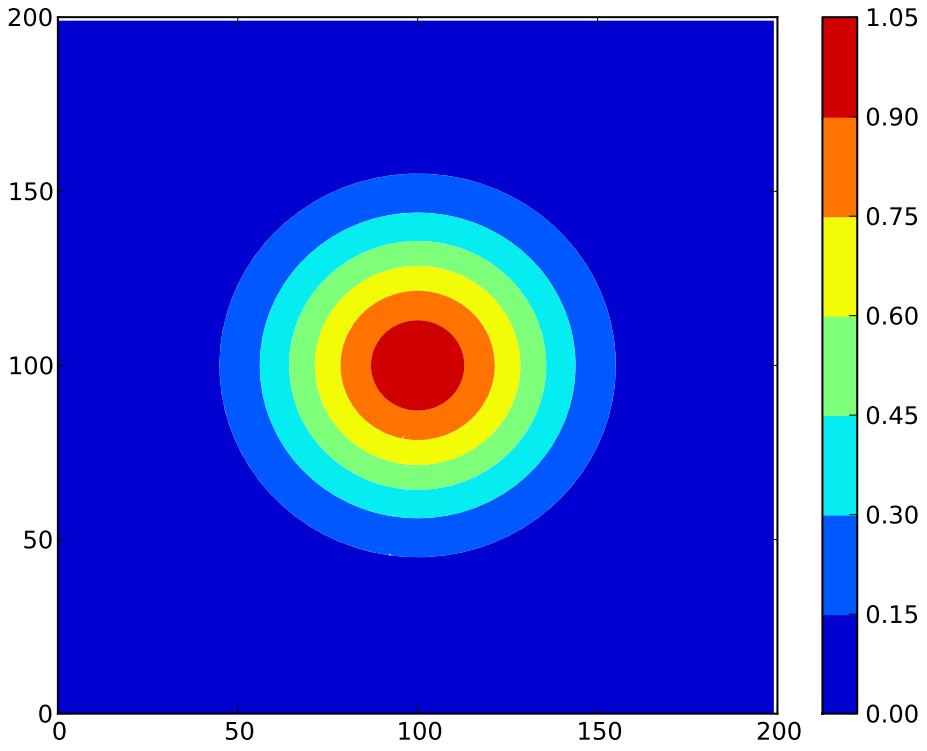
2D plotting

```
In [33]: plt.clear()

In [34]: x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

In [35]: g =
  np.exp(-(x**2 + y**2) / 16)

In [36]: plt.contourf(g)

In [37]:
```



2D plotting

```
In [33]: plt.clear()

In [34]: x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

In [35]: g =
  np.exp(-(x**2 + y**2) / 16)

In [36]: plt.contourf(g)

In [37]: plt.colorbar()

In [38]:
```



2D plotting

```
In [33]: plt.clear()

In [34]: x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

In [35]: g =
  np.exp(-(x**2 + y**2) / 16)

In [36]: plt.contourf(g)

In [37]: plt.colorbar()

In [38]: plt.clf()

In [39]:
```

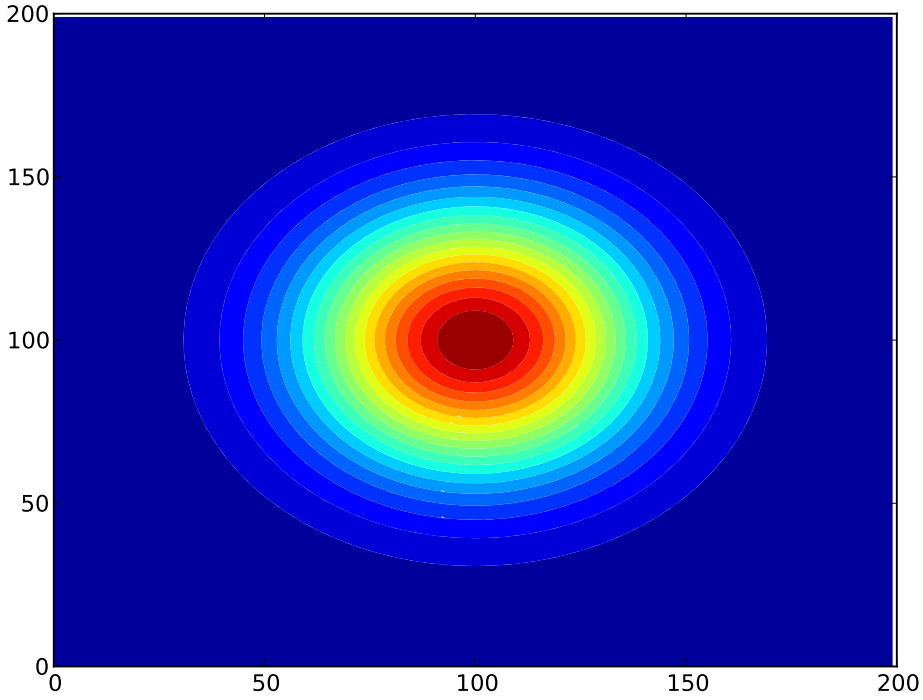
2D plotting

```
In [33]: plt.clear()

In [34]: x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

In [35]: g =
  np.exp(-(x**2 + y**2) / 16)

In [36]: plt.contourf(g)

In [37]: plt.colorbar()

In [38]: plt.clf()

In [39]: V = np.linspace(g.min(),
  g.max(), 21)
```

```
In [40]:
```

2D plotting

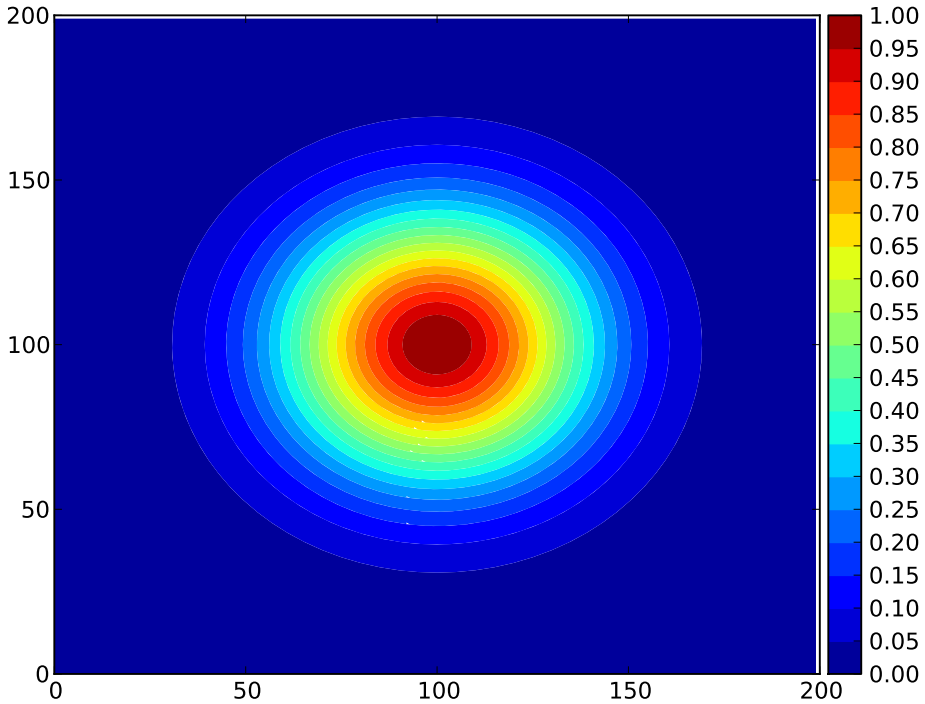
```
In [33]: plt.clear()

In [34]: x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

In [35]: g =
  np.exp(-(x**2 + y**2) / 16)

In [36]: plt.contourf(g)

In [37]: plt.colorbar()

In [38]: plt.clf()

In [39]: V = np.linspace(g.min(),
  g.max(), 21)
```




```
In [40]: plt.contourf(g, V)

In [41]:
```

2D plotting

```
In [33]: plt.clear()

In [34]: x, y = np.mgrid[-10:10:0.1,
  -10:10:0.1]

In [35]: g =
  np.exp(-(x**2 + y**2) / 16)

In [36]: plt.contourf(g)

In [37]: plt.colorbar()

In [38]: plt.clf()

In [39]: V = np.linspace(g.min(),
  g.max(), 21)
```




```
In [40]: plt.contourf(g, V)

In [41]: plt.colorbar(format = '%.2f',
  pad = 0.01, fraction = 0.09,
  ticks = V)
```

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.

SciNet compute • calcul CANADA

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.

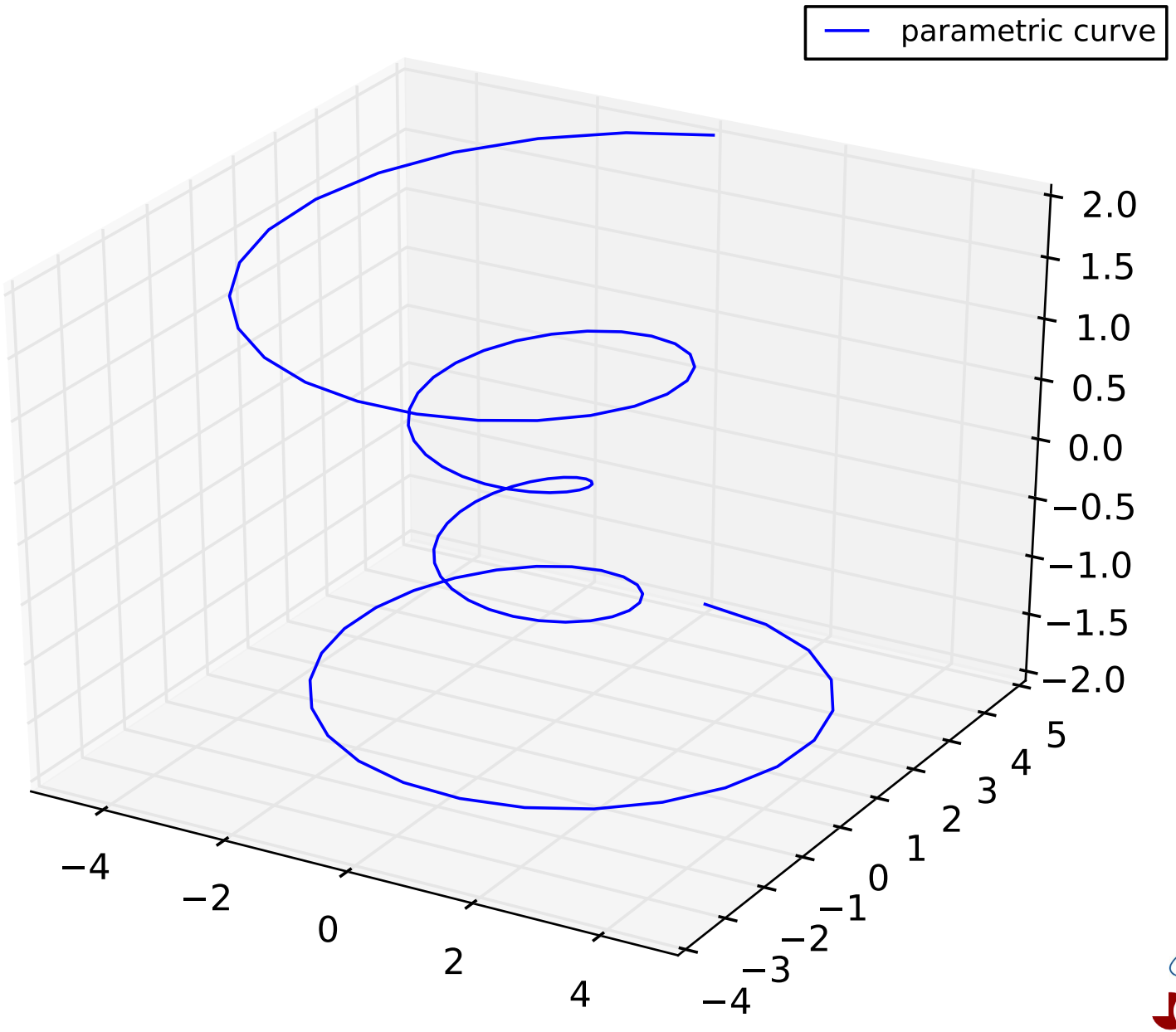
SciNet compute • calcul CANADA

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

Plotting for professional scientists

Scientists must make their 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.

SciNet compute • calcul CANADA

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!

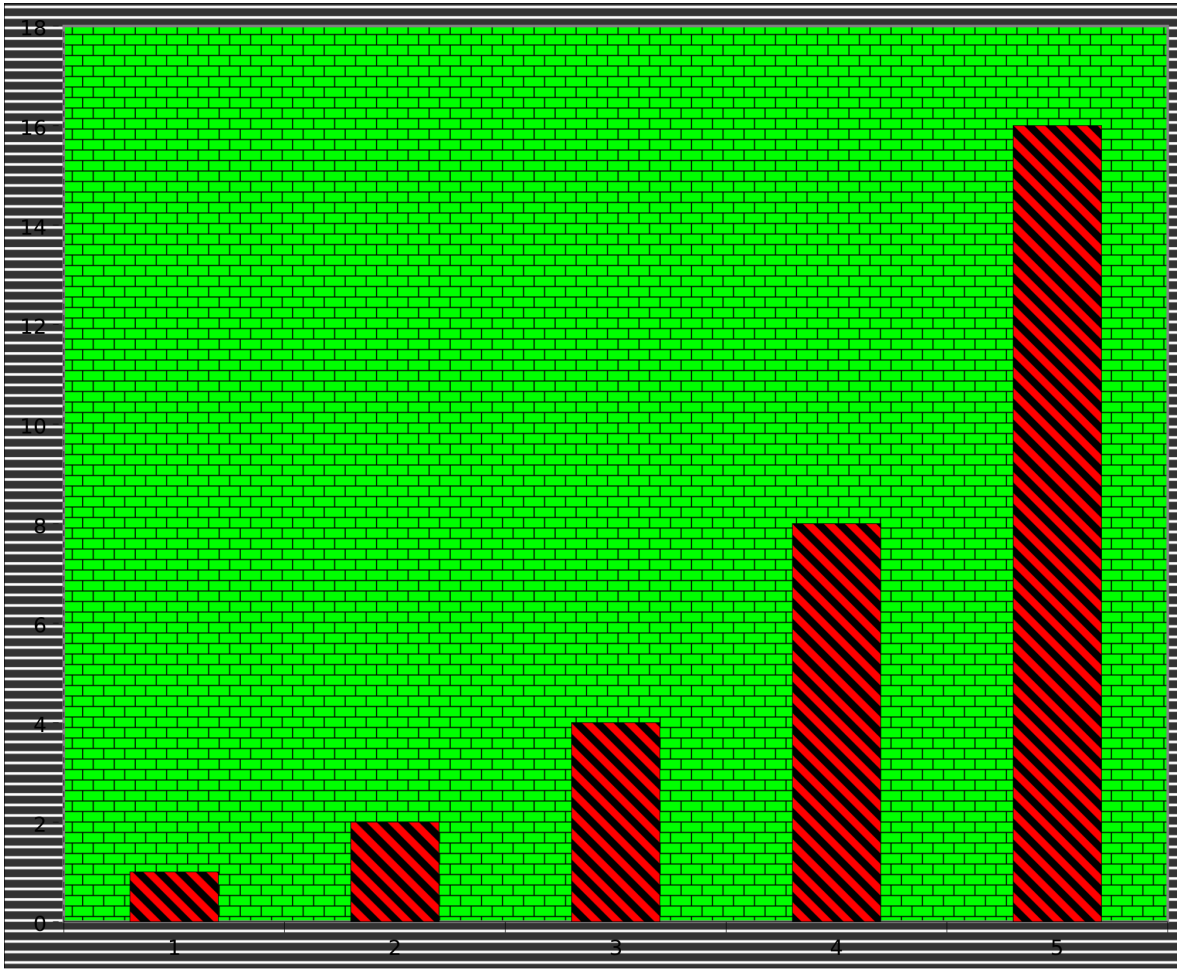


Chart junk is the unnecessary cluttering of your plot. Don't do it!

Image stolen from Wikipedia.

Advanced plotting

```
In [42]:
```

Advanced plotting

```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]:
```

Advanced plotting

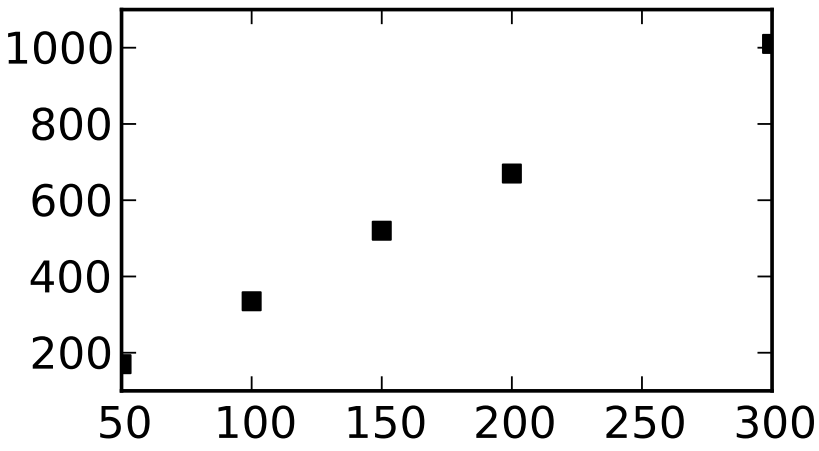
```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]:
```

Advanced plotting

```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]:
```

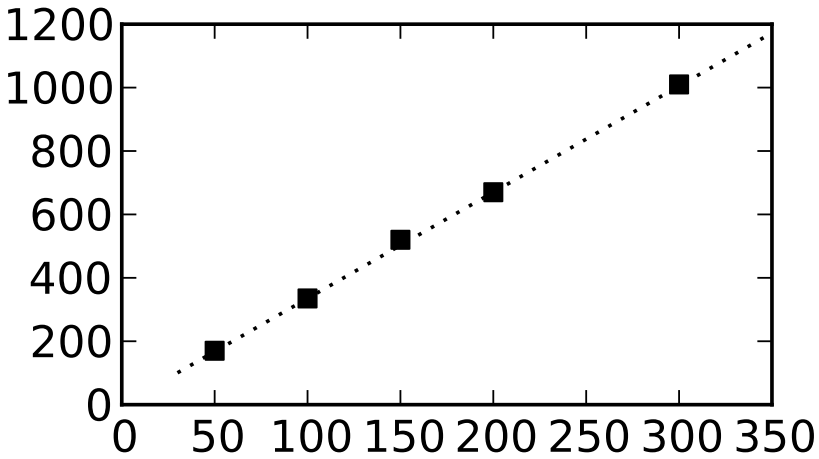
Advanced plotting

```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]:
```



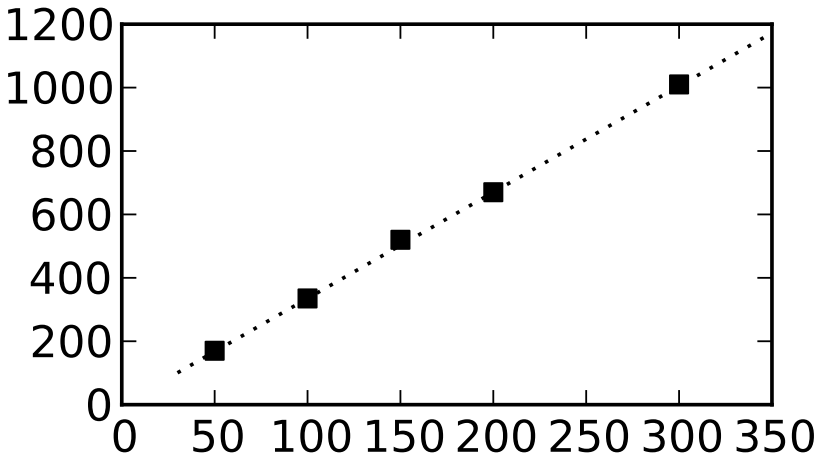
Advanced plotting

```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]:
```



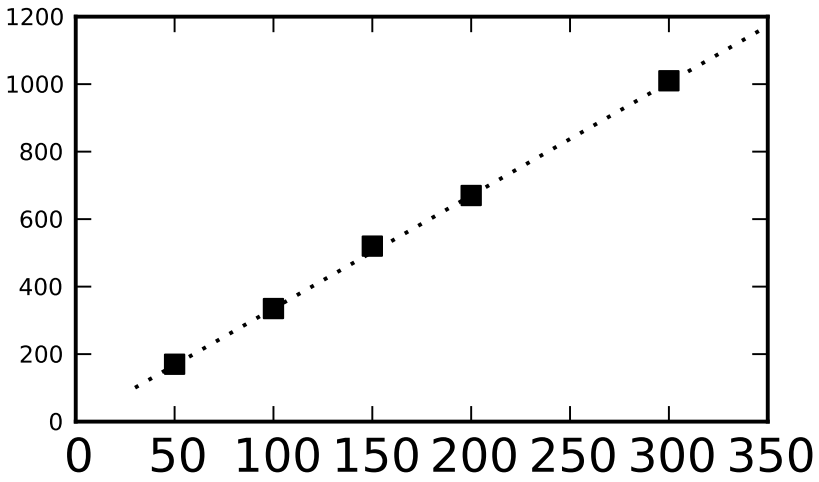
Advanced plotting

```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]: for t in
  a.yaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [50]:
```



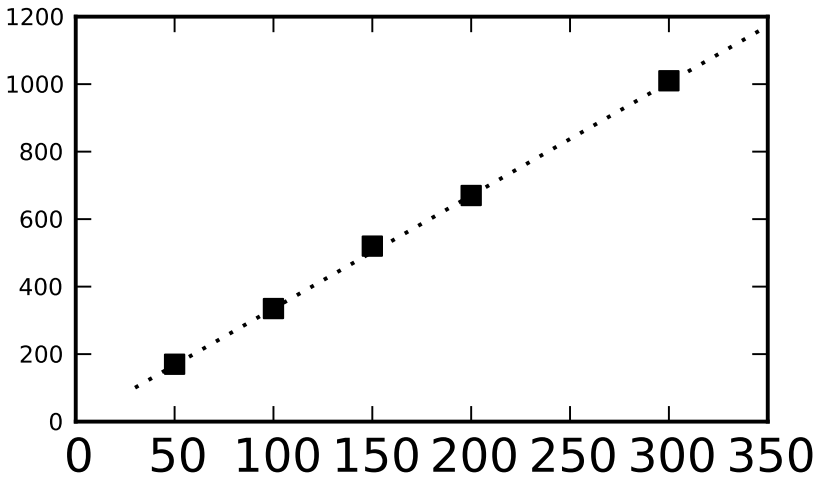
Advanced plotting

```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]: for t in
  a.yaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [50]: plt.show()
In [51]:
```



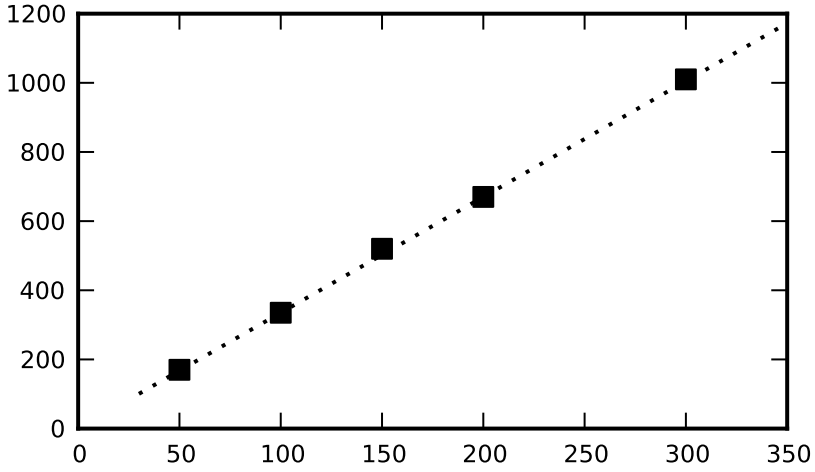
Advanced plotting

```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]: for t in
  a.yaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [50]: plt.show()
In [51]: for t in
  a.xaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [52]:
```



Advanced plotting

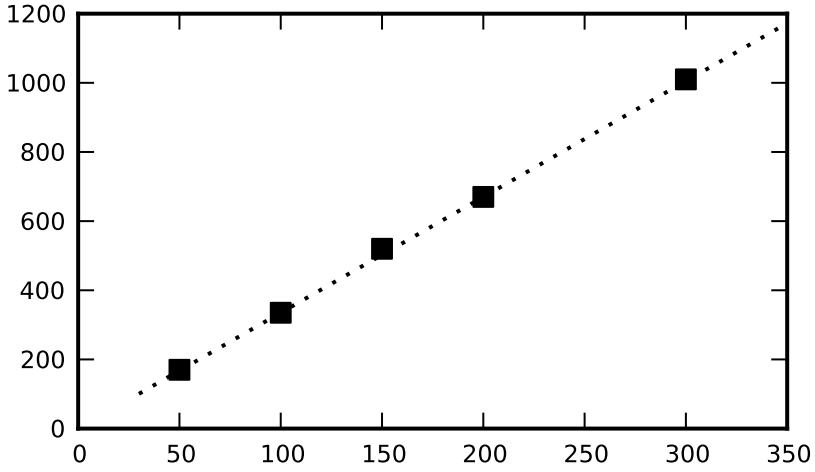
```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]: for t in
  a.yaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [50]: plt.show()
In [51]: for t in
  a.xaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [52]: plt.show()
```




```
In [53]:
```

Advanced plotting

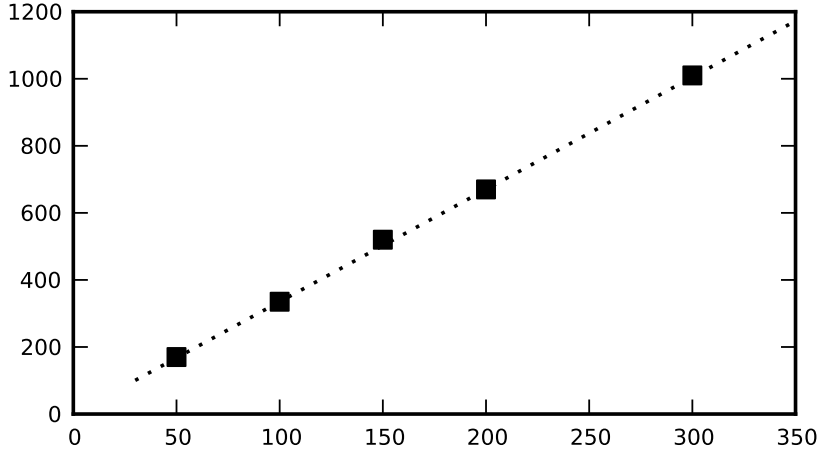
```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]: for t in
  a.yaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [50]: plt.show()
In [51]: for t in
  a.xaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [52]: plt.show()
```




```
In [53]: fig.subplots_adjust(
  top = 0.97, right = 0.98,
  left = 0.15, bottom = 0.15)
In [54]:
```

Advanced plotting

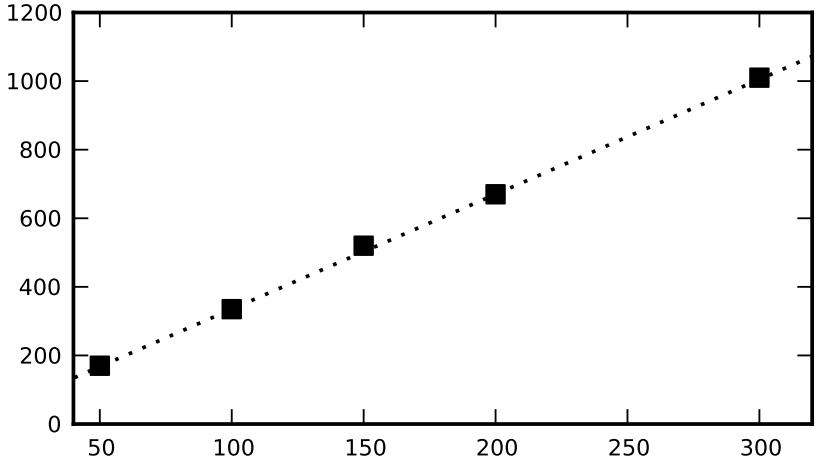
```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]: for t in
  a.yaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [50]: plt.show()
In [51]: for t in
  a.xaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [52]: plt.show()
```




```
In [53]: fig.subplots_adjust(
  top = 0.97, right = 0.98,
  left = 0.15, bottom = 0.15)
In [54]: plt.show()
In [55]:
```

Advanced plotting

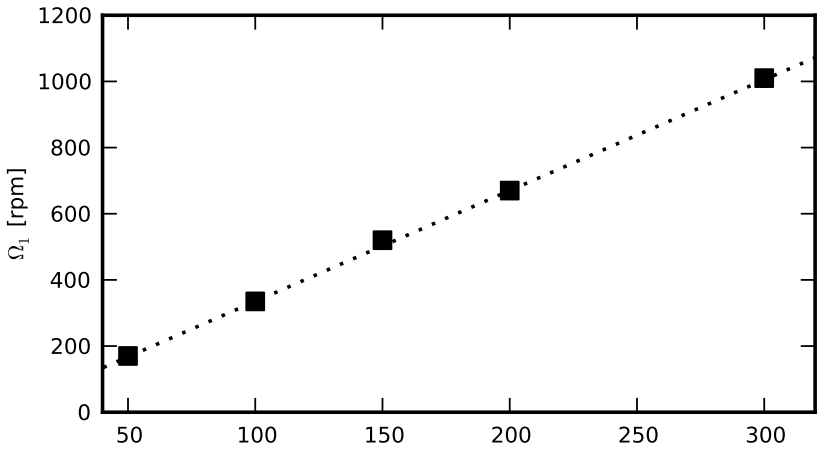
```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]: for t in
  a.yaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [50]: plt.show()
In [51]: for t in
  a.xaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [52]: plt.show()
```




```
In [53]: fig.subplots_adjust(
  top = 0.97, right = 0.98,
  left = 0.15, bottom = 0.15)
In [54]: plt.show()
In [55]: plt.xlim(40, 320)
Out[56]: (40, 320)
In [57]:
```

Advanced plotting

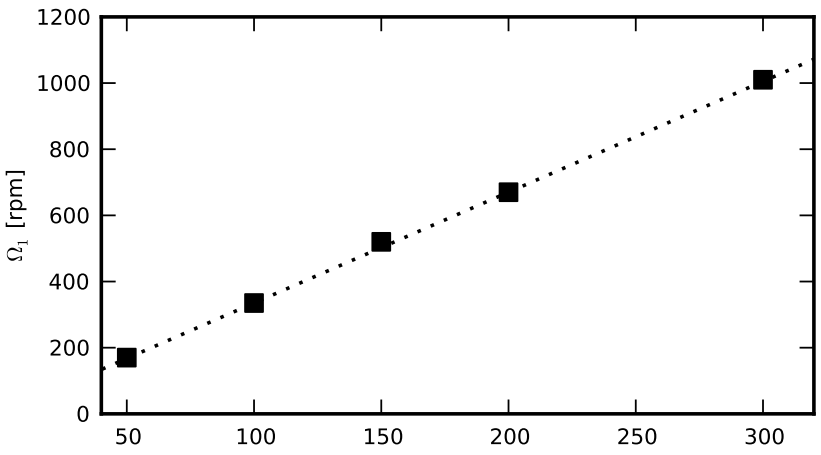
```
In [42]: o2 = [50, 100, 150, 200, 300]
In [43]: o1 = [170, 335, 520, 670, 1010]
In [44]: o = linspace(30, 350, 100)
In [45]: fig = plt.figure(
  figsize = (3.375,2), dpi = 600)
In [46]: a = fig.add_subplot(1,1,1)
In [47]: plt.plot(o2, o1, 'ks',
  markersize = 5)
In [48]: plt.plot(o, 3.35 * o, ':',
  color = 'k')
In [49]: for t in
  a.yaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [50]: plt.show()
In [51]: for t in
  a.xaxis.get_major_ticks():
  t.label.set_fontsize(6)
In [52]: plt.show()
```




```
In [53]: fig.subplots_adjust(
  top = 0.97, right = 0.98,
  left = 0.15, bottom = 0.15)
In [54]: plt.show()
In [55]: plt.xlim(40, 320)
Out[56]: (40, 320)
In [57]: plt.ylabel(
  r'$\Omega_1$ [rpm]', fontsize = 6,
  verticalalignment = 'center')
```

Advanced plotting, continued

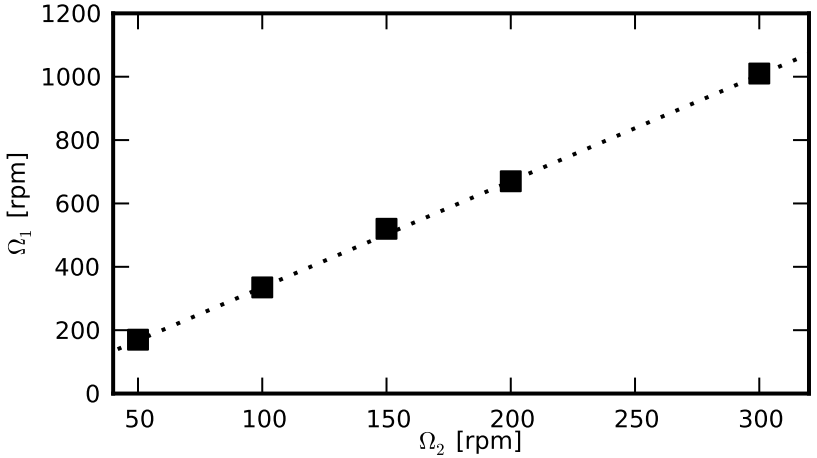
```
In [58]:
```



Advanced plotting, continued

```
In [58]: plt.text(163, -180.,
  r'$\Omega_2$ [rpm]', fontsize = 6)
```

```
In [59]:
```

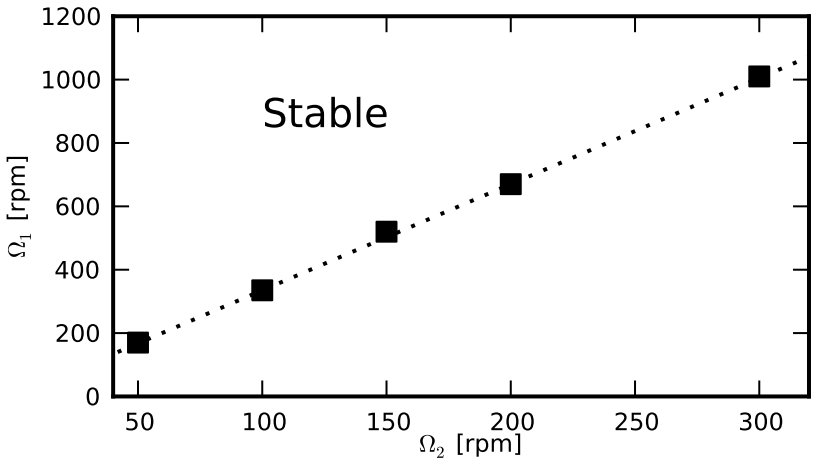


Advanced plotting, continued

```
In [58]: plt.text(163, -180.,
  r'$\Omega_2$ [rpm]', fontsize = 6)
```

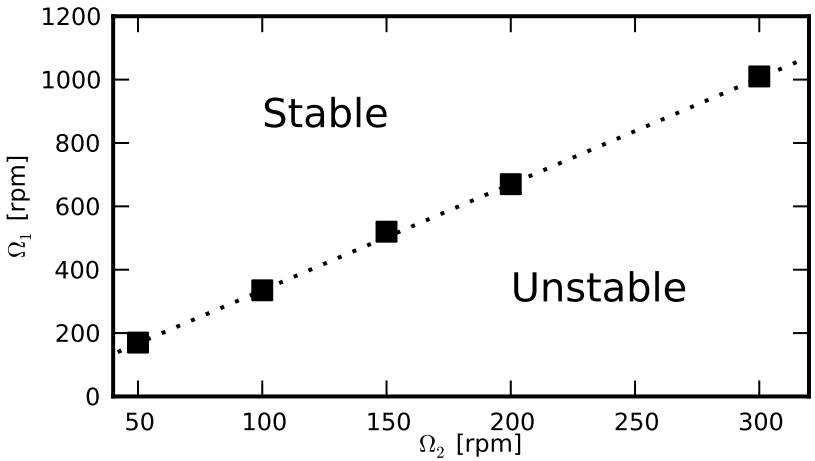
```
In [59]: plt.text(100, 850., 'Stable',
  fontsize = 10)
```

```
In [60]:
```



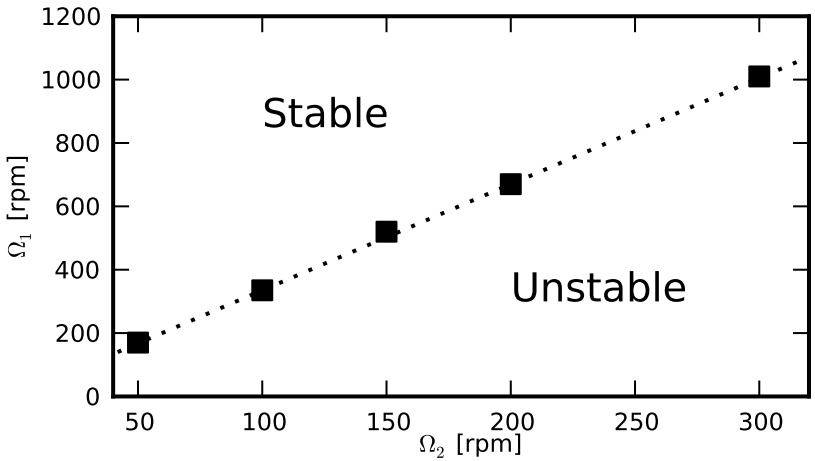
Advanced plotting, continued

```
In [58]: plt.text(163, -180.,
  r'$\Omega_2$ [rpm]', fontsize = 6)
In [59]: plt.text(100, 850., 'Stable',
  fontsize = 10)
In [60]: plt.text(200, 300., 'Unstable',
  fontsize = 10)
In [61]:
```



Advanced plotting, continued

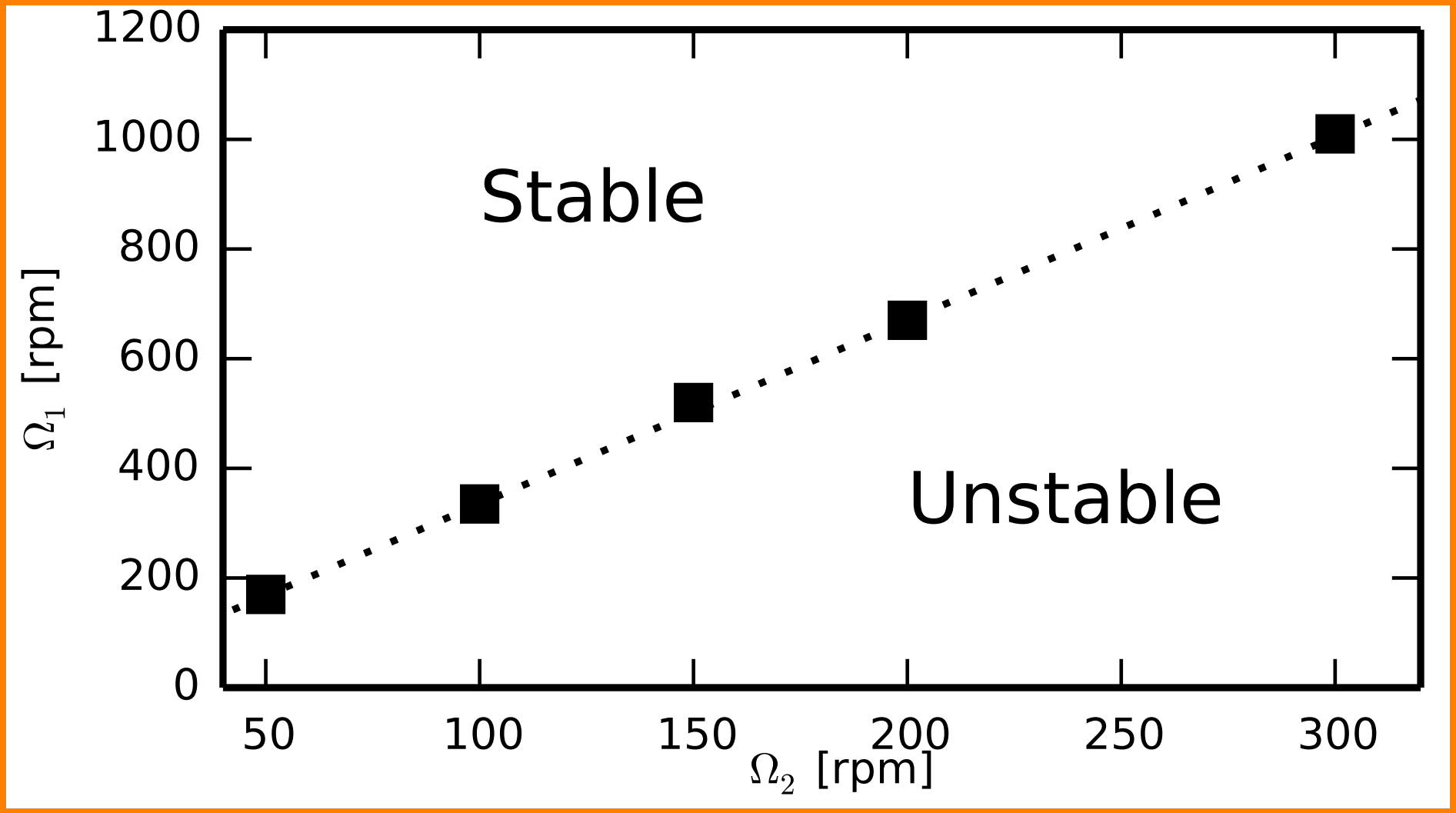
```
In [58]: plt.text(163, -180.,
  r'$\Omega_2$ [rpm]', fontsize = 6)
In [59]: plt.text(100, 850., 'Stable',
  fontsize = 10)
In [60]: plt.text(200, 300., 'Unstable',
  fontsize = 10)
In [61]: plt.savefig(
  'critical_rossby.eps', dpi = 600)
```



Use savefig to save your figures.

- Possible file types include EPS, PDF, PNG, and others.
- JPEG is not a valid type.
- Put this in your program which generates your figure for your paper!

Final product



Another plotting example

```
In [62]:
```

Another plotting example

```
In [62]: r,z,s,psi = calc_psi_pop2011()
```

```
In [63]:
```

Another plotting example

```
In [62]: r,z,s,psi = calc_psi_pop2011()
In [63]: maxshear = 10.0; maxpsi = 42.0
In [64]:
```

Another plotting example

```
In [62]: r,z,s,psi = calc_psi_pop2011()
In [63]: maxshear = 10.0; maxpsi = 42.0
In [64]: V = np.linspace(-maxshear,
  0.0, 75)
In [65]:
```

Another plotting example

```
In [62]: r,z,s,psi = calc_psi_pop2011()
In [63]: maxshear = 10.0; maxpsi = 42.0
In [64]: V = np.linspace(-maxshear,
  0.0, 75)
In [65]: import matplotlib as mpl
In [66]: mpl.rcParams['ytick.labelsize']
  = 6
In [67]: mpl.rcParams['xtick.labelsize']
  = 6
In [68]: fig = figure(
  figsize = (3.375,2), dpi = 600)
In [69]:
```

Another plotting example

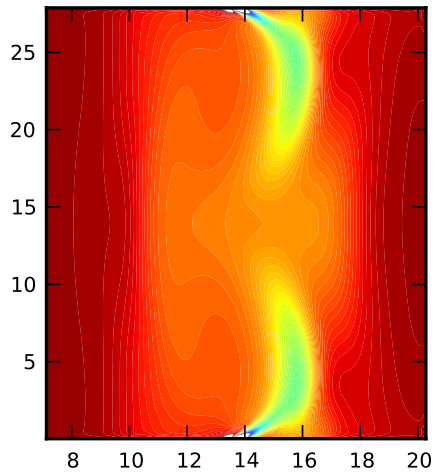
```
In [62]: r,z,s,psi = calc_psi_pop2011()
In [63]: maxshear = 10.0; maxpsi = 42.0
In [64]: V = np.linspace(-maxshear,
  0.0, 75)
In [65]: import matplotlib as mpl
In [66]: mpl.rcParams['ytick.labelsize']
  = 6
In [67]: mpl.rcParams['xtick.labelsize']
  = 6
In [68]: fig = figure(
  figsize = (3.375,2), dpi = 600)
In [69]: fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
In [70]:
```

Another plotting example

```
In [62]: r,z,s,psi = calc_psi_pop2011()
In [63]: maxshear = 10.0; maxpsi = 42.0
In [64]: V = np.linspace(-maxshear,
  0.0, 75)
In [65]: import matplotlib as mpl
In [66]: mpl.rcParams['ytick.labelsize']
  = 6
In [67]: mpl.rcParams['xtick.labelsize']
  = 6
In [68]: fig = figure(
  figsize = (3.375,2), dpi = 600)
In [69]: fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
In [70]: a = fig.add_subplot(1,2,1)
In [71]:
```

Another plotting example

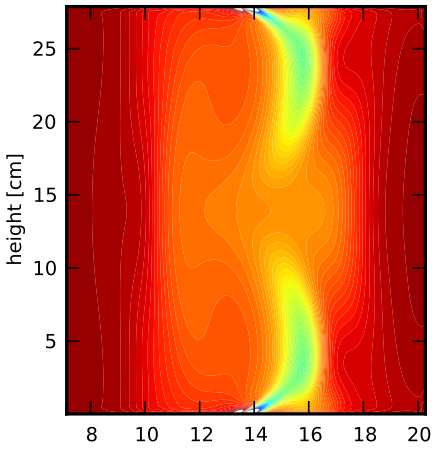
```
In [62]: r,z,s,psi = calc_psi_pop2011()
In [63]: maxshear = 10.0; maxpsi = 42.0
In [64]: V = np.linspace(-maxshear,
  0.0, 75)
In [65]: import matplotlib as mpl
In [66]: mpl.rcParams['ytick.labelsize']
  = 6
In [67]: mpl.rcParams['xtick.labelsize']
  = 6
In [68]: fig = figure(
  figsize = (3.375,2), dpi = 600)
In [69]: fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
In [70]: a = fig.add_subplot(1,2,1)
In [71]: plt.contourf(r, z, s, V)
```




```
In [72]:
```

Another plotting example

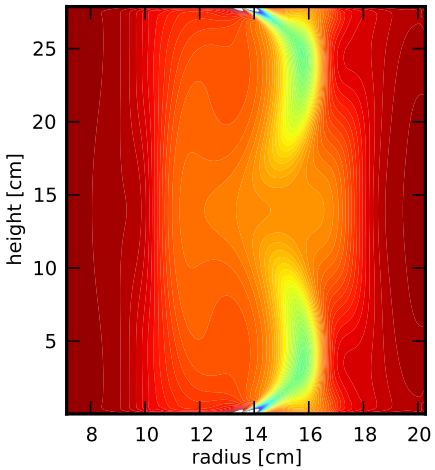
```
In [62]: r,z,s,psi = calc_psi_pop2011()
In [63]: maxshear = 10.0; maxpsi = 42.0
In [64]: V = np.linspace(-maxshear,
  0.0, 75)
In [65]: import matplotlib as mpl
In [66]: mpl.rcParams['ytick.labelsize']
  = 6
In [67]: mpl.rcParams['xtick.labelsize']
  = 6
In [68]: fig = figure(
  figsize = (3.375,2), dpi = 600)
In [69]: fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
In [70]: a = fig.add_subplot(1,2,1)
In [71]: plt.contourf(r, z, s, V)
```




```
In [72]: plt.ylabel('height [cm]',
  fontsize = 6,
  horizontalalignment='center')
In [73]:
```

Another plotting example

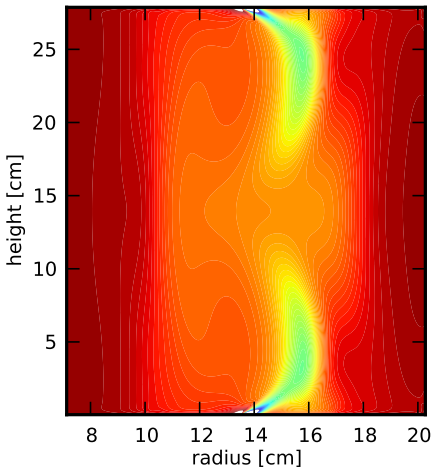
```
In [62]: r,z,s,psi = calc_psi_pop2011()
In [63]: maxshear = 10.0; maxpsi = 42.0
In [64]: V = np.linspace(-maxshear,
  0.0, 75)
In [65]: import matplotlib as mpl
In [66]: mpl.rcParams['ytick.labelsize']
  = 6
In [67]: mpl.rcParams['xtick.labelsize']
  = 6
In [68]: fig = figure(
  figsize = (3.375,2), dpi = 600)
In [69]: fig.subplots_adjust(top = 0.97,
  right = 0.98, left = 0.09,
  bottom = 0.12, wspace = 0.0,
  hspace = 0.0)
In [70]: a = fig.add_subplot(1,2,1)
In [71]: plt.contourf(r, z, s, V)
```




```
In [72]: plt.ylabel('height [cm]',
  fontsize = 6,
  horizontalalignment='center')
In [73]: plt.xlabel('radius [cm]',
  fontsize = 6,
  verticalalignment='center')
```

Another plotting example, continued

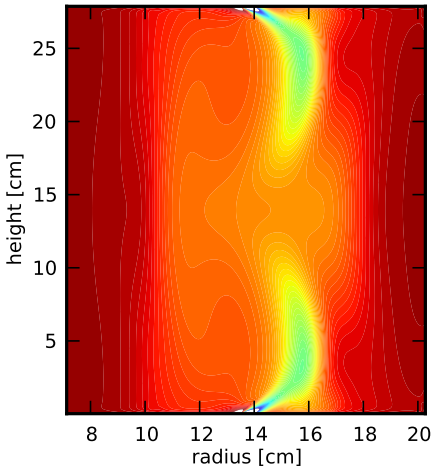
```
In [74]:
```



Another plotting example, continued

```
In [74]: V2 = np.linspace(-maxpsi,
  maxpsi,22)
```

```
In [75]:
```

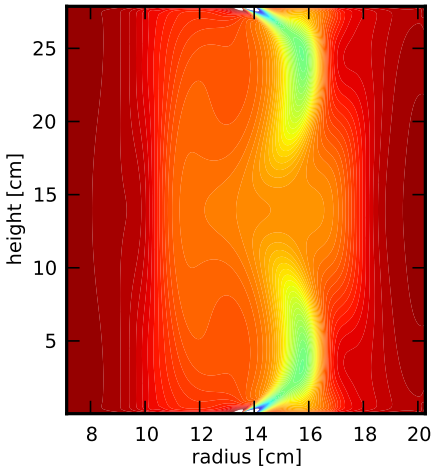


Another plotting example, continued

```
In [74]: V2 = np.linspace(-maxpsi,
  maxpsi,22)
```

```
In [75]: styles = []
```

```
In [76]:
```



Another plotting example, continued

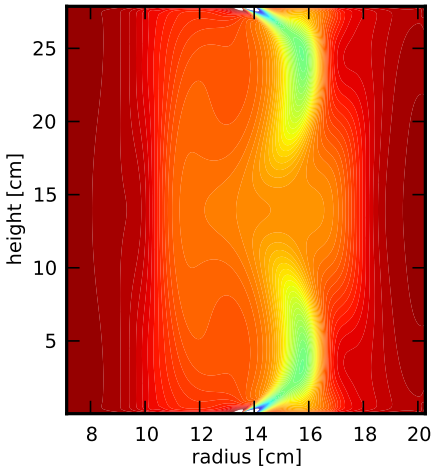
```
In [74]: V2 = np.linspace(-maxpsi,
  maxpsi,22)
```

```
In [75]: styles = []
```

```
In [76]: for i in range(11):
  styles.append('dashed')
```

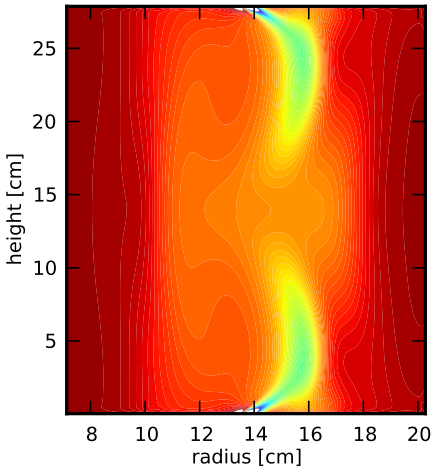
```
In [77]: for i in range(11):
  styles.append('solid')
```

```
In [78]:
```



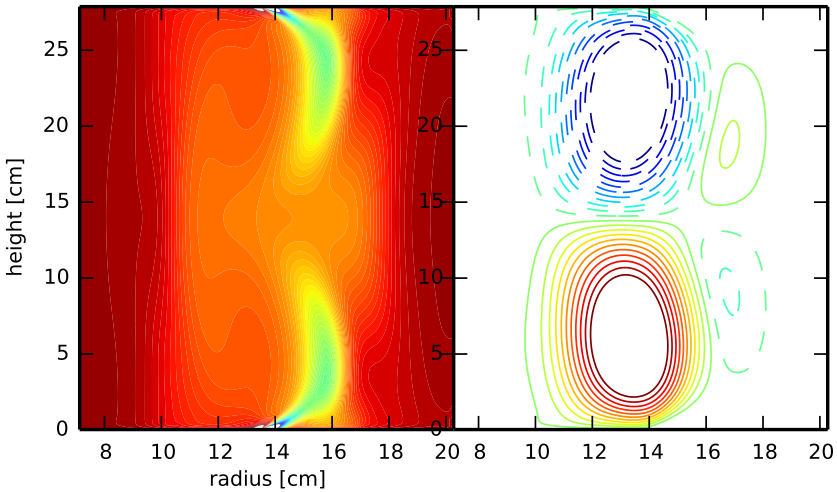
Another plotting example, continued

```
In [74]: V2 = np.linspace(-maxpsi,
  maxpsi,22)
In [75]: styles = []
In [76]: for i in range(11):
  styles.append('dashed')
In [77]: for i in range(11):
  styles.append('solid')
In [78]: a2 = fig.add_subplot(1,2,2)
In [79]:
```



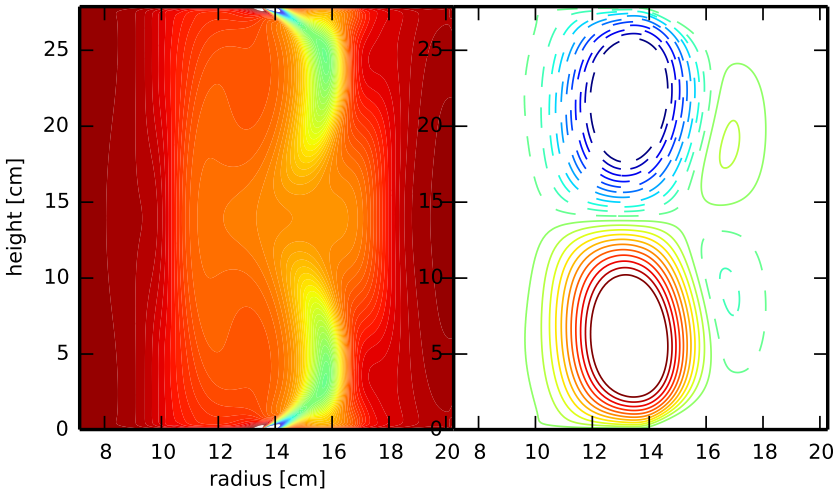
Another plotting example, continued

```
In [74]: V2 = np.linspace(-maxpsi,
  maxpsi,22)
In [75]: styles = []
In [76]: for i in range(11):
  styles.append('dashed')
In [77]: for i in range(11):
  styles.append('solid')
In [78]: a2 = fig.add_subplot(1,2,2)
In [79]: contour(r, z, psi, V2,
  linewidths = 0.5, linestyles = styles)
In [80]:
```



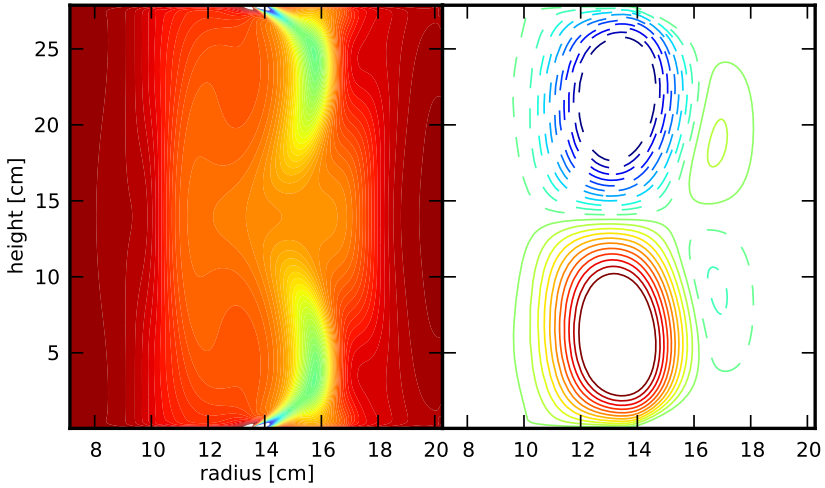
Another plotting example, continued

```
In [74]: V2 = np.linspace(-maxpsi,
  maxpsi,22)
In [75]: styles = []
In [76]: for i in range(11):
  styles.append('dashed')
In [77]: for i in range(11):
  styles.append('solid')
In [78]: a2 = fig.add_subplot(1,2,2)
In [79]: contour(r, z, psi, V2,
  linewidths = 0.5, linestyles = styles)
In [80]: a2.set_yticklabels([])
In [81]:
```



Another plotting example, continued

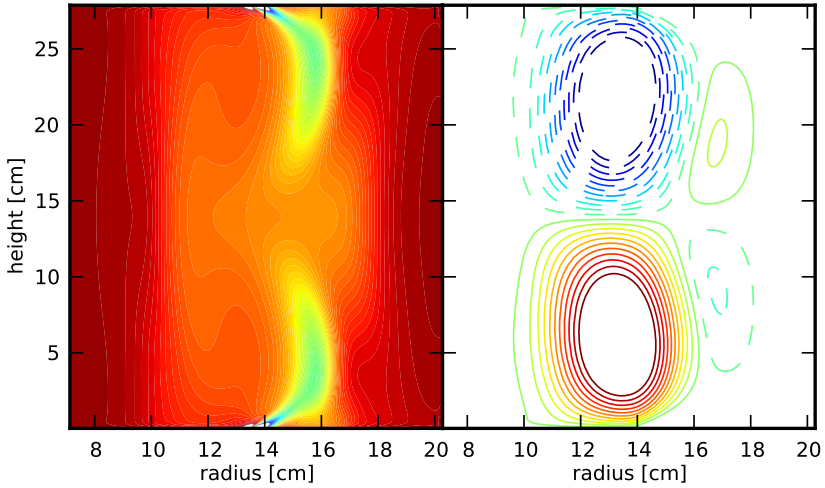
```
In [74]: V2 = np.linspace(-maxpsi,
  maxpsi,22)
In [75]: styles = []
In [76]: for i in range(11):
  styles.append('dashed')
In [77]: for i in range(11):
  styles.append('solid')
In [78]: a2 = fig.add_subplot(1,2,2)
In [79]: contour(r, z, psi, V2,
  linewidths = 0.5, linestyles = styles)
In [80]: a2.set_yticklabels([])
In [81]: plt.show()
```




```
In [82]:
```

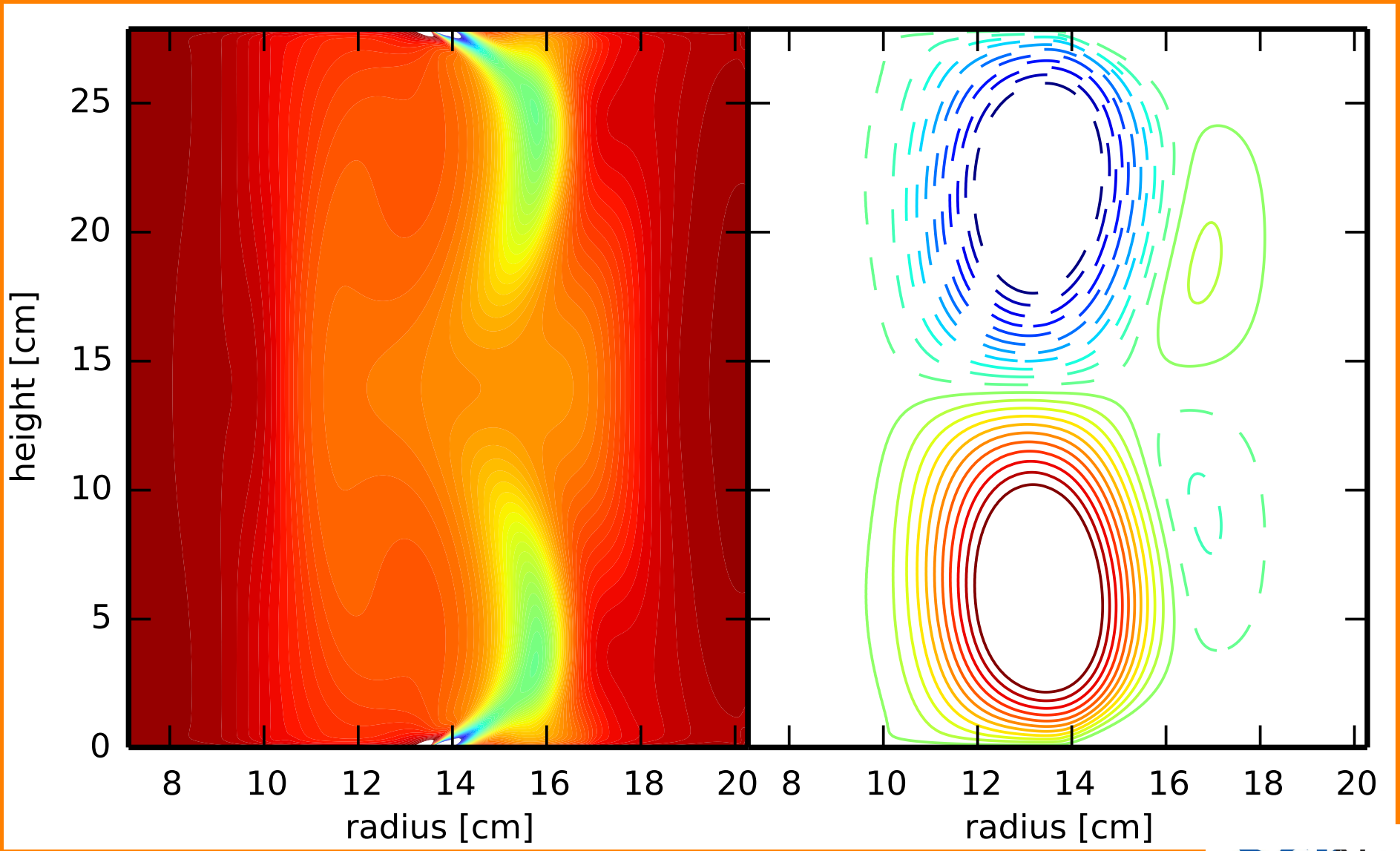
Another plotting example, continued

```
In [74]: V2 = np.linspace(-maxpsi,
  maxpsi,22)
In [75]: styles = []
In [76]: for i in range(11):
  styles.append('dashed')
In [77]: for i in range(11):
  styles.append('solid')
In [78]: a2 = fig.add_subplot(1,2,2)
In [79]: contour(r, z, psi, V2,
  linewidths = 0.5, linestyles = styles)
In [80]: a2.set_yticklabels([])
In [81]: plt.show()
```




```
In [82]: xlabel('radius [cm]',
  fontsize = 6,
  verticalalignment='center')
```

Another final product



Animations

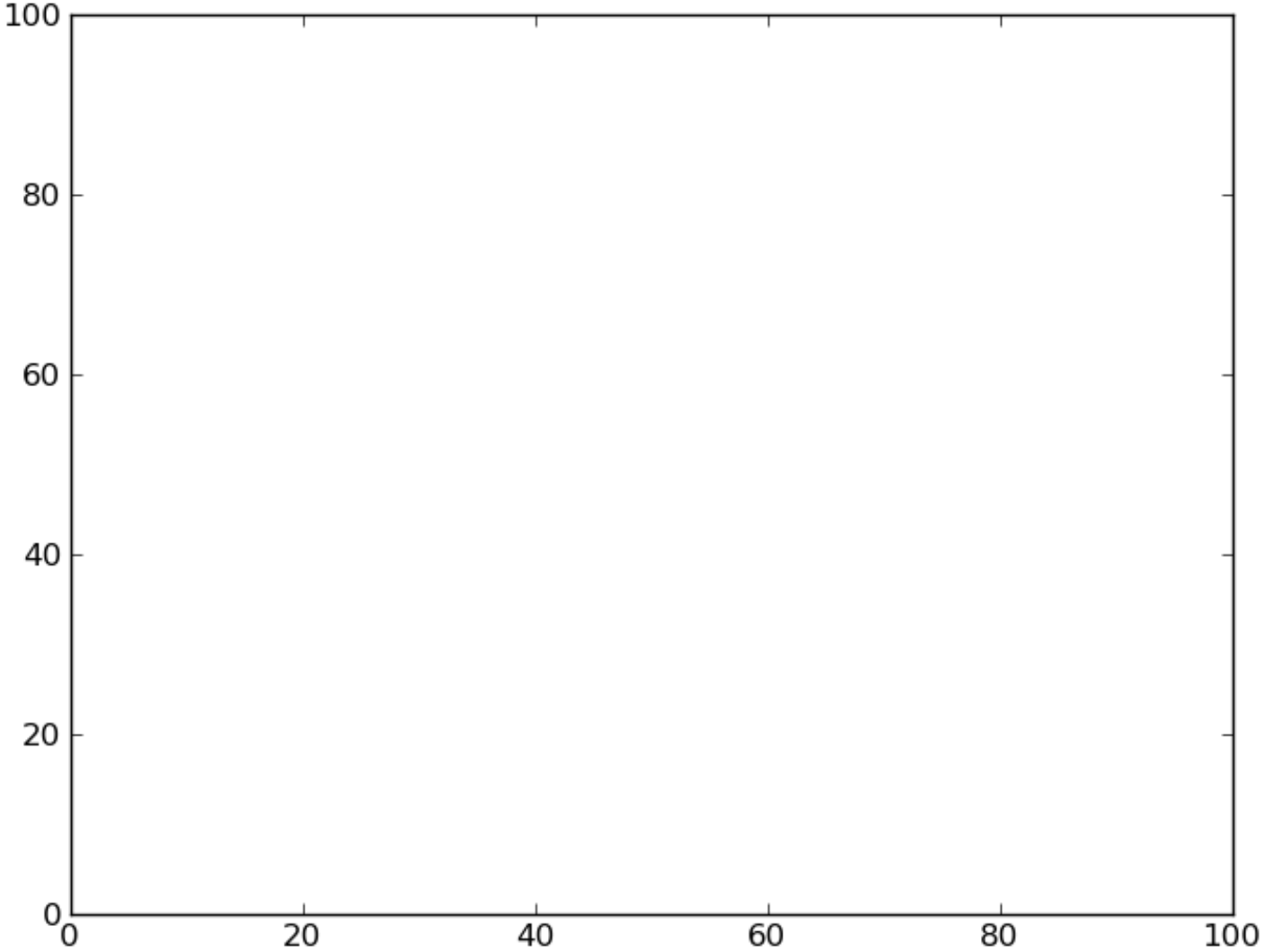
To make animations in Python, use the matplotlib.animation module.

```
# moviewriter.py
import numpy as np
import pylab as p
import matplotlib.animation as an

# Create an FFMPEG writer.
ffwriter = an.writers['ffmpeg']

# Metadata is good practice.
metadata = dict(title = 'My First
  Animation', artist = 'me')

# Specify some features.
writer = ffwriter(fps = 15,
  metadata = metadata)
```

```
fig = p.figure()
# Open an empty plot.
l, = p.plot([], [], 'ko')

p.xlim(-3, 3);    p.ylim(-3, 3)
x0 = np.zeros(10.0)
y0 = np.zeros(10.0)

with writer.saving(fig,
  'myanimation.mp4', 200)
  for i in range(200):
    x0 += 0.1 * np.random.randn(10)
    y0 += 0.1 * np.random.randn(10)

    # Specify the new x and y data.
    l.set_data(x0, y0)
    writer.grab_frame()
```

Heavily stolen from matplotlib.org/examples/animation.

Our first animation



About animations

Some notes about animations in Python:

- Python itself does not create animations, it can only create images.
- Python streams the images into an external animation program.
- Two encoders can be used, ffmpeg and mencoder. These must be installed on your computer separately for Python to access and use them.
- More advanced examples of animations can be found at `http://matplotlib.org/1.3.0/examples/animation`

SciNet compute • calcul CANADA

Homework 3

Questions for this week:

1. In the following two questions, use version control on your answers. Submit the output of 'hg log' for each question. We expect to see several commits.
2. Write a Python script which creates a professional-quality plot of your data. Use any data which is relevant to the work that you do. Please submit the code, the plot and the data which was used.
3. The NetCDF file, trajectory.nc, which is on the website, contains the positions and velocities for a set of particles at a number of points in time. Write a Python program which creates an animation of the data using just the positions at different times. Please submit both the code and your final animation.

SciNet compute • calcul CANADA