Scientific visualization (with Python)

Basics

1D data

Semilog plot

Log plot

PDF

2D data

Types of plot (scalars)

Types of plot (flow)

Colormaps

3D data (animations)

Refs.

Scientific visualization (with Python)

Tomás Chor

May 11, 2017

Introduction

- Many plotting systems to choose from
  - matplotlib
  - gnuplot
  - paraview
  - scilab
  - basemap
  - matlab
  - techplot
- They can serve many purposes
  - quick glance at data
  - presentation to others
  - scientific publication
  - general public

Introduction

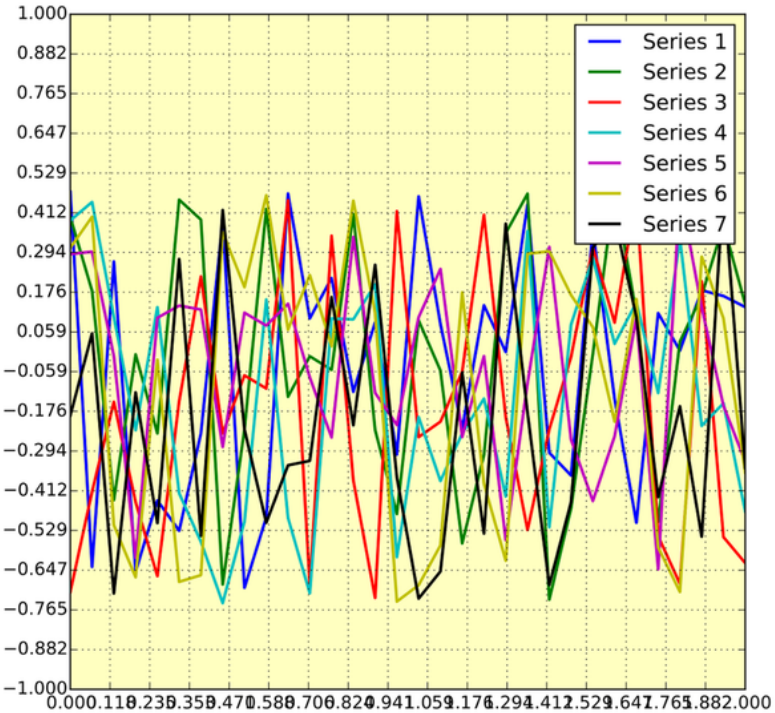
- The nature of the plot depends on the nature of data
  - gridded data
  - sparse data
  - dimensionality ($f(x)$, $f(x, y)$, $f(x, y, t)$ ...)
- Interactive plots are becoming very easy to make, but we won't focus on it
- Checkout basemap (python) for map-projection plotting

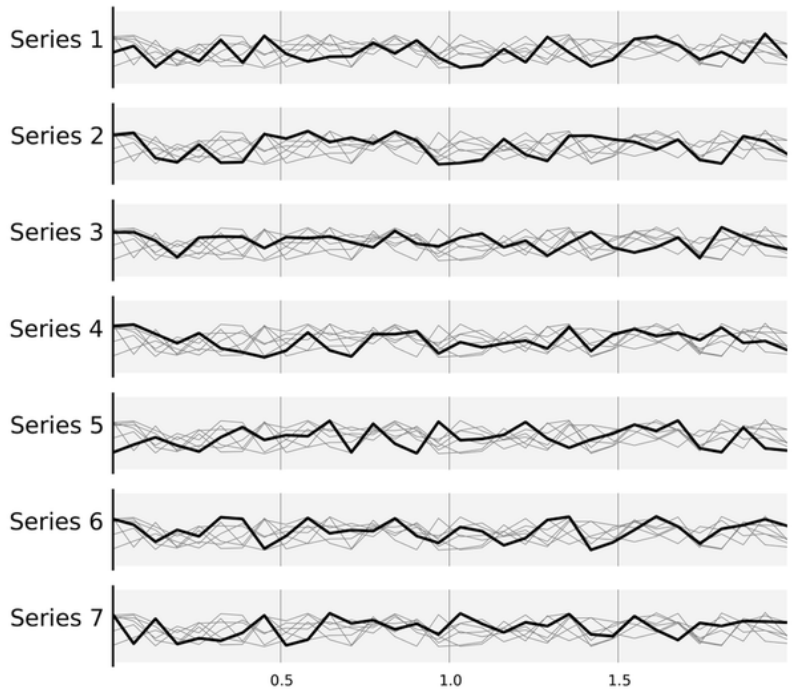
We have to get subjective

Some general subjective rules to aim for

- clarity
- precision
- efficiency

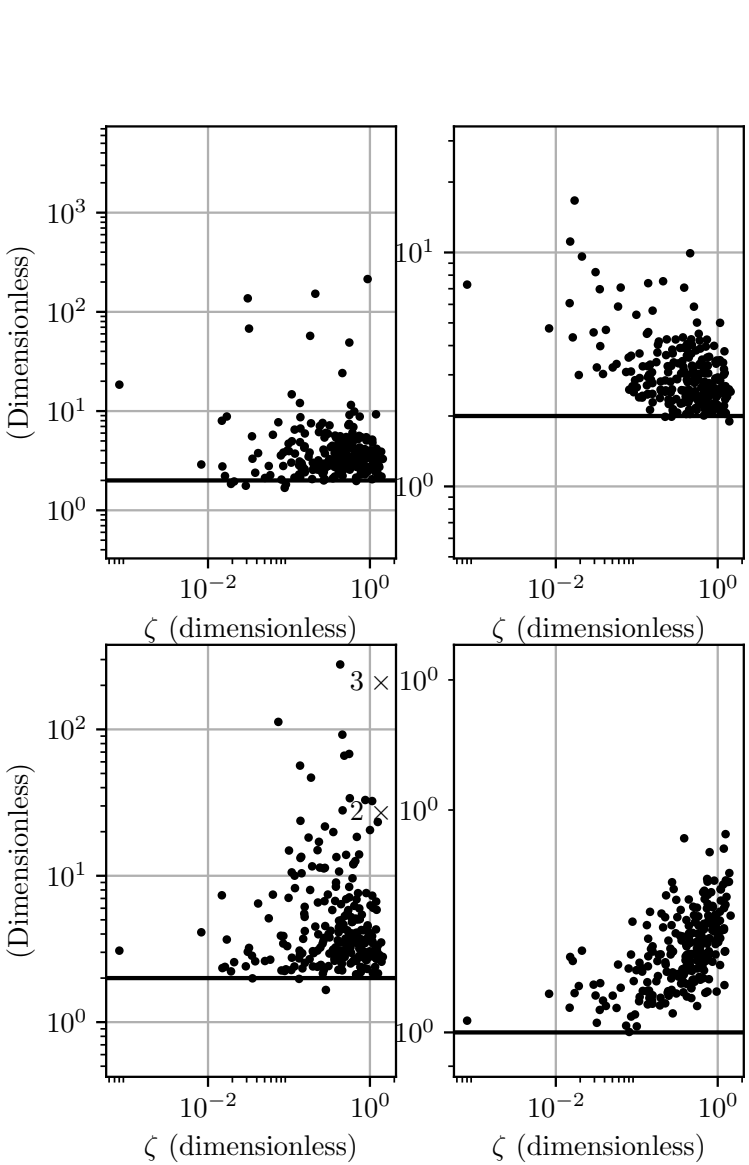
It might be subjective, but we can tell the difference

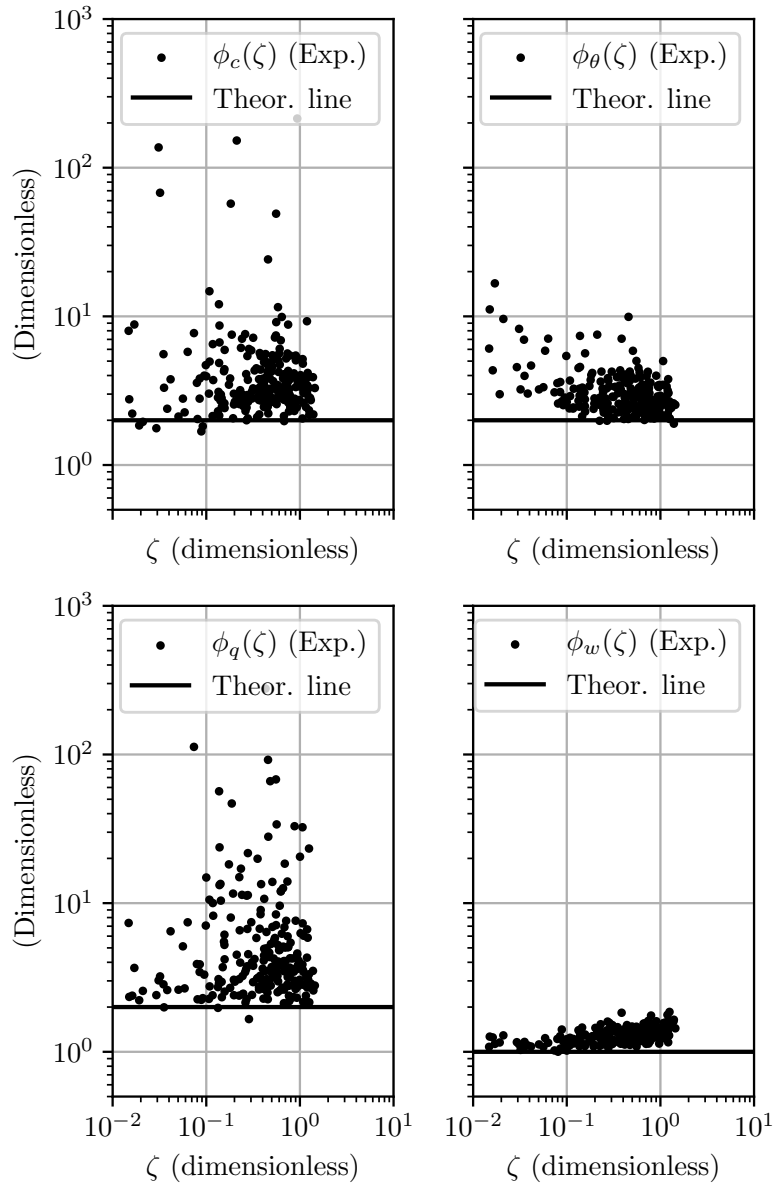




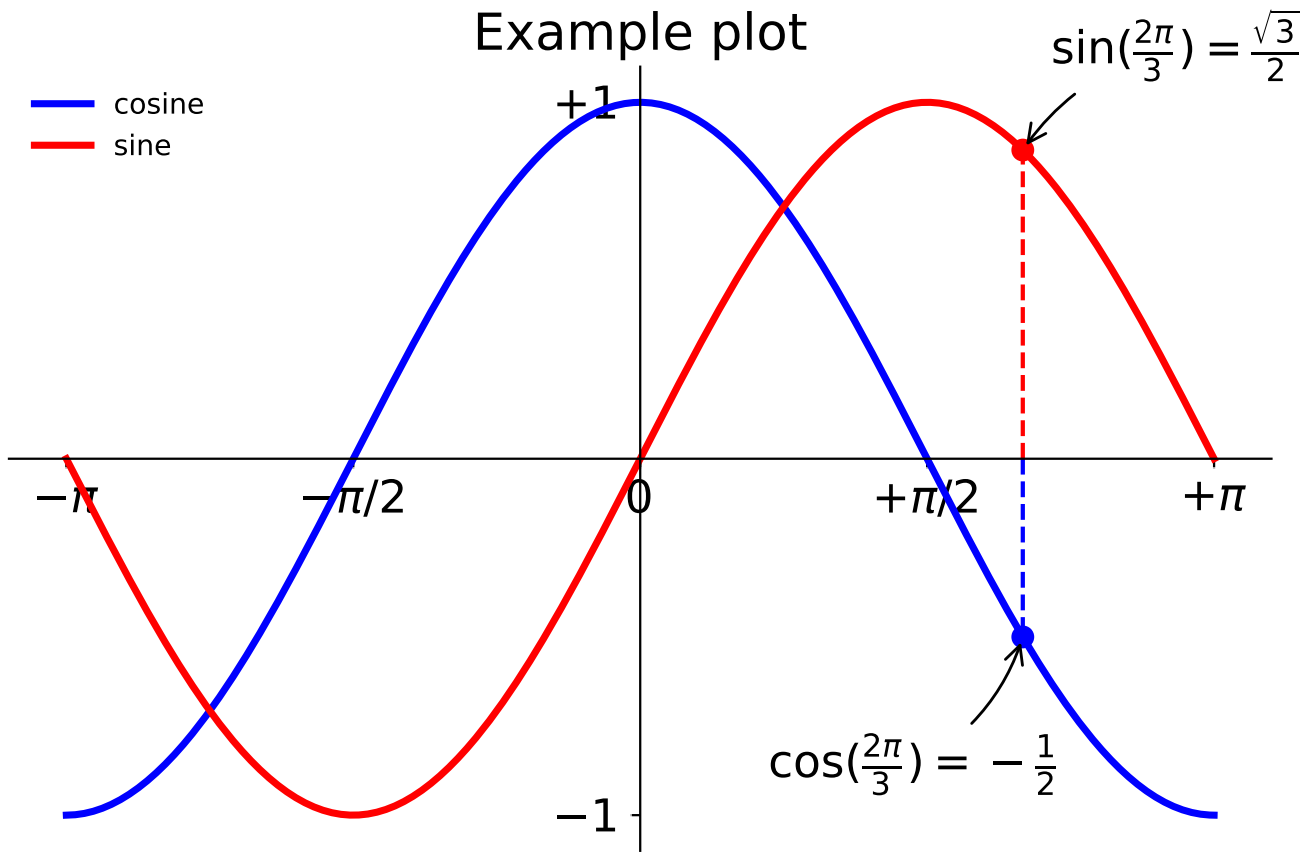
Reproduced from Rougier, Droettboom, and Bourne, (2014). (Although notice that there should be labels in the plot!)

If you're showing more than one plot on that are meant to be compared, keep the same scales





Make your message easy with your plot



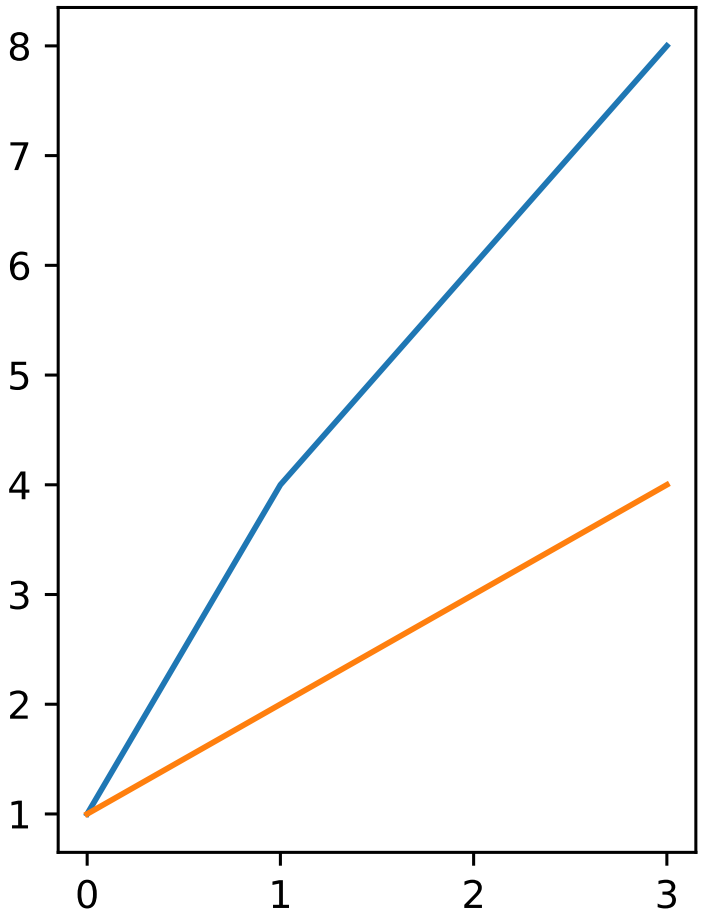
Taken from Rougier, (2015).

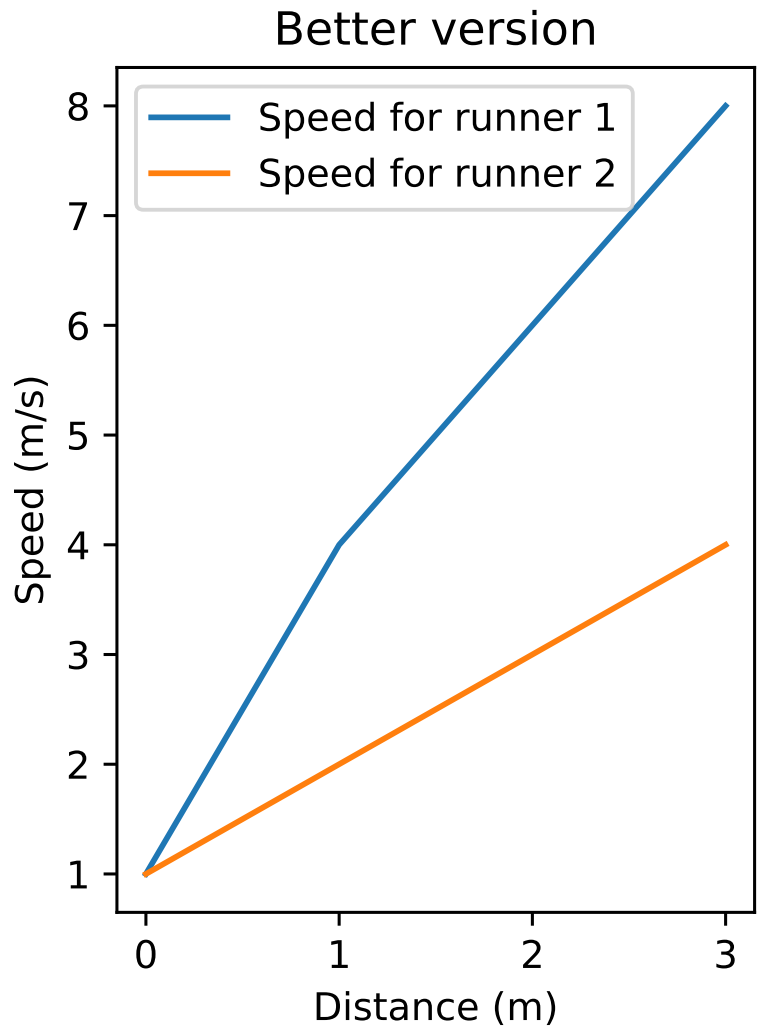
- Curves
- Labels
- Ticks (and labels)
- Title
- Legend
- Other helpful marks

Some things are almost mandatory

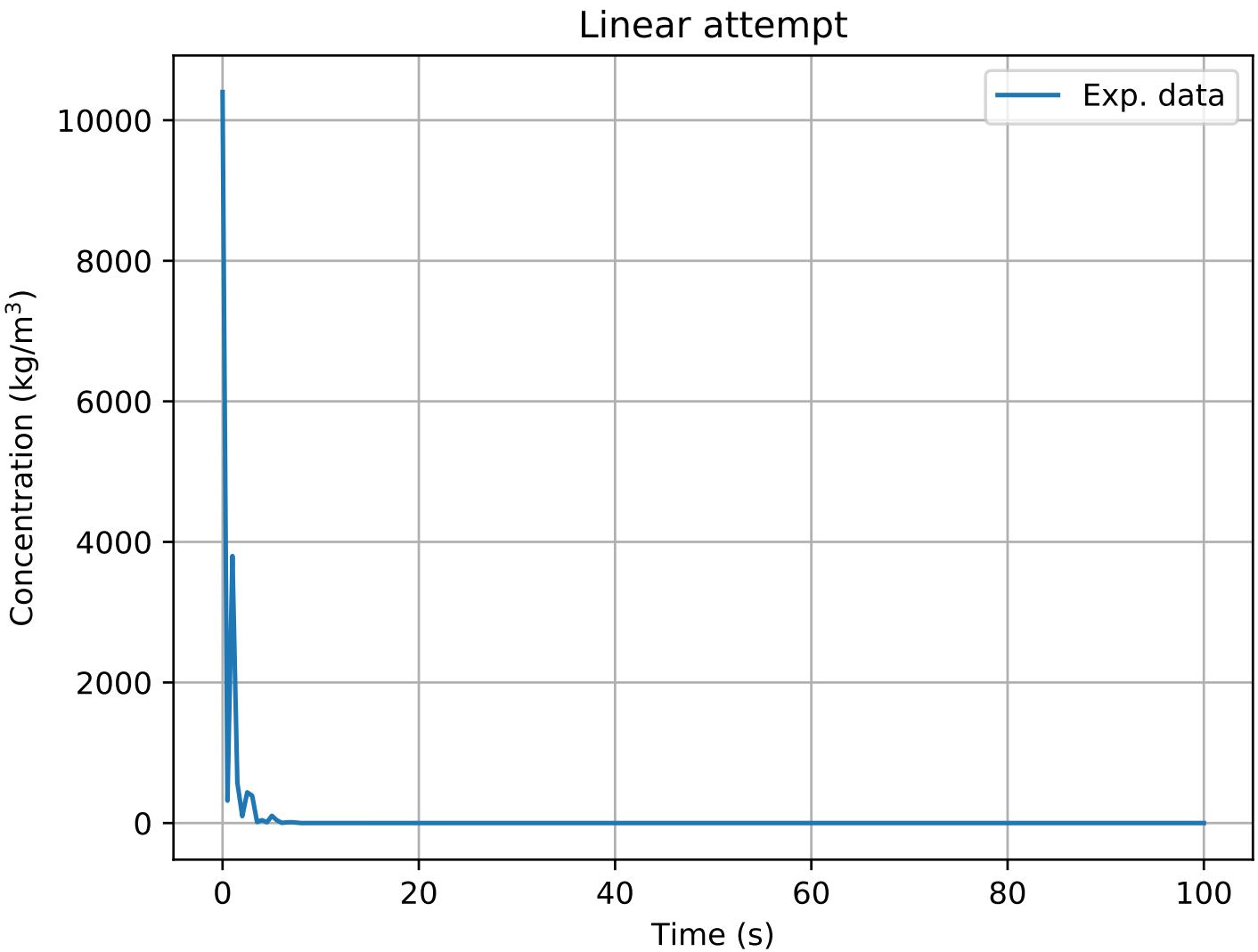
Every plot should be self-sufficient in its content. So they should include

- labels
- units (scientific world uses metric)
- legends

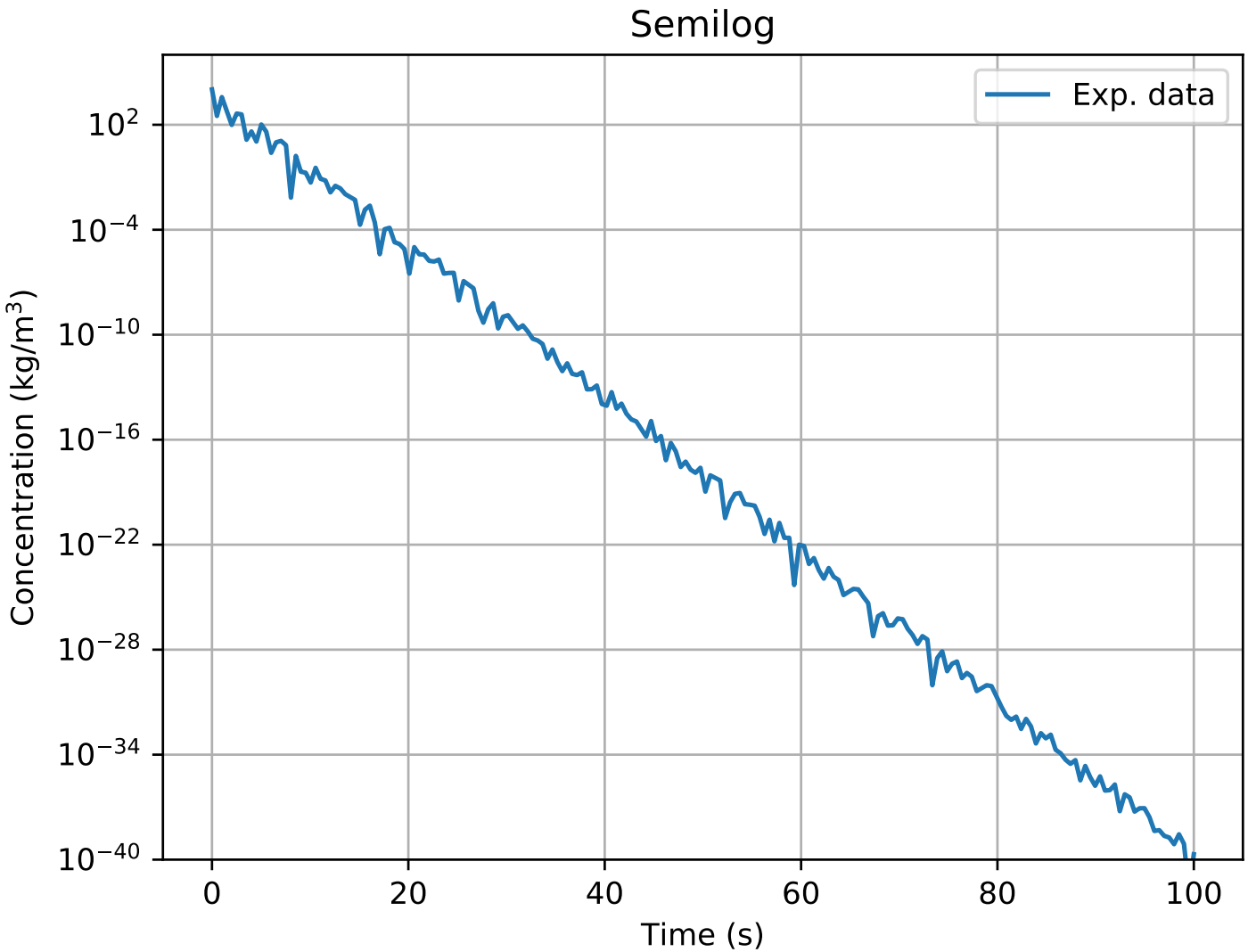

8
7
6
5
4
3
2
1
0
1
2
3


Better version
Speed for runner 1
Speed for runner 2
Speed (m/s)
Distance (m)
8
7
6
5
4
3
2
1
0
1
2
3

Log-Lin plots

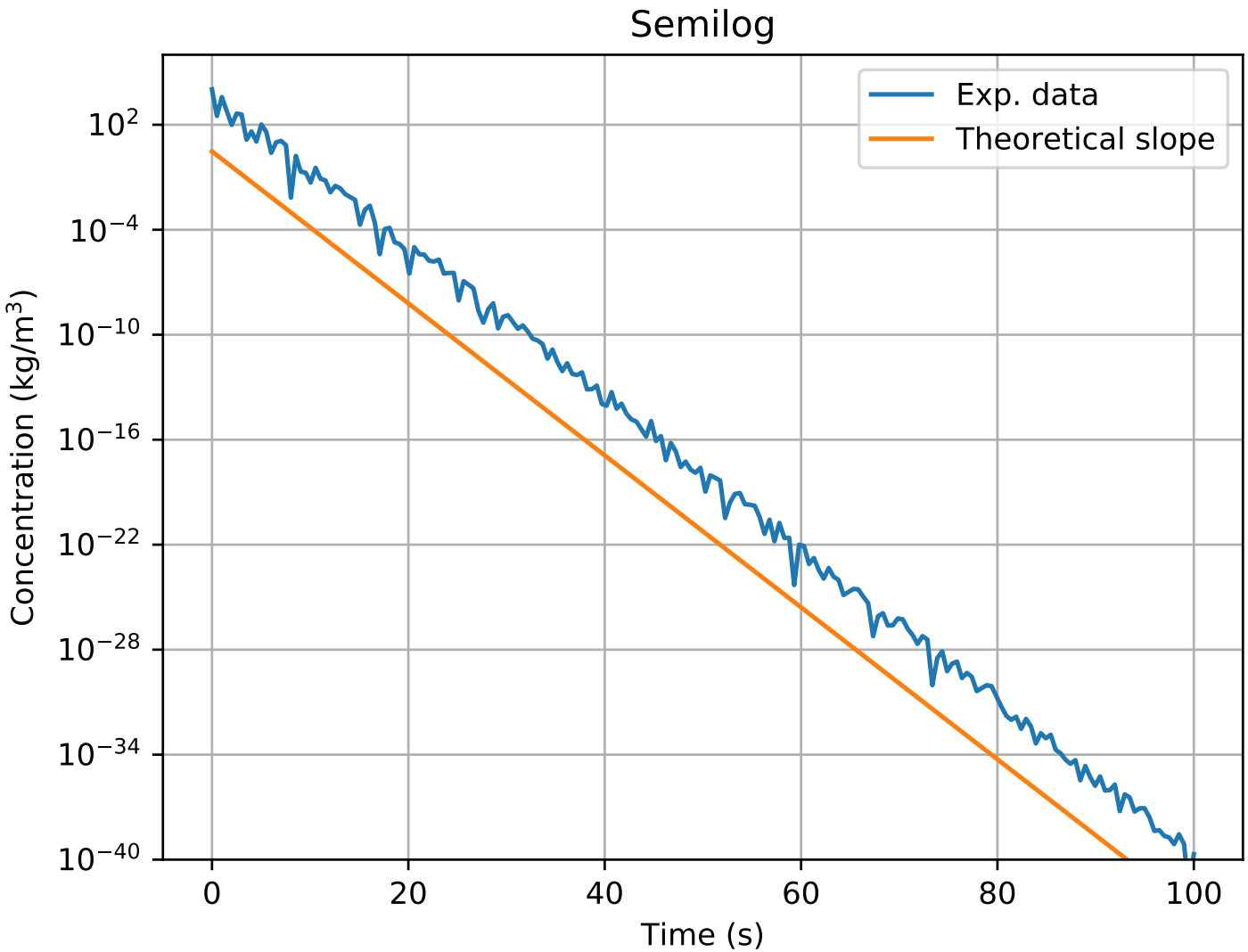


Log-Lin plots



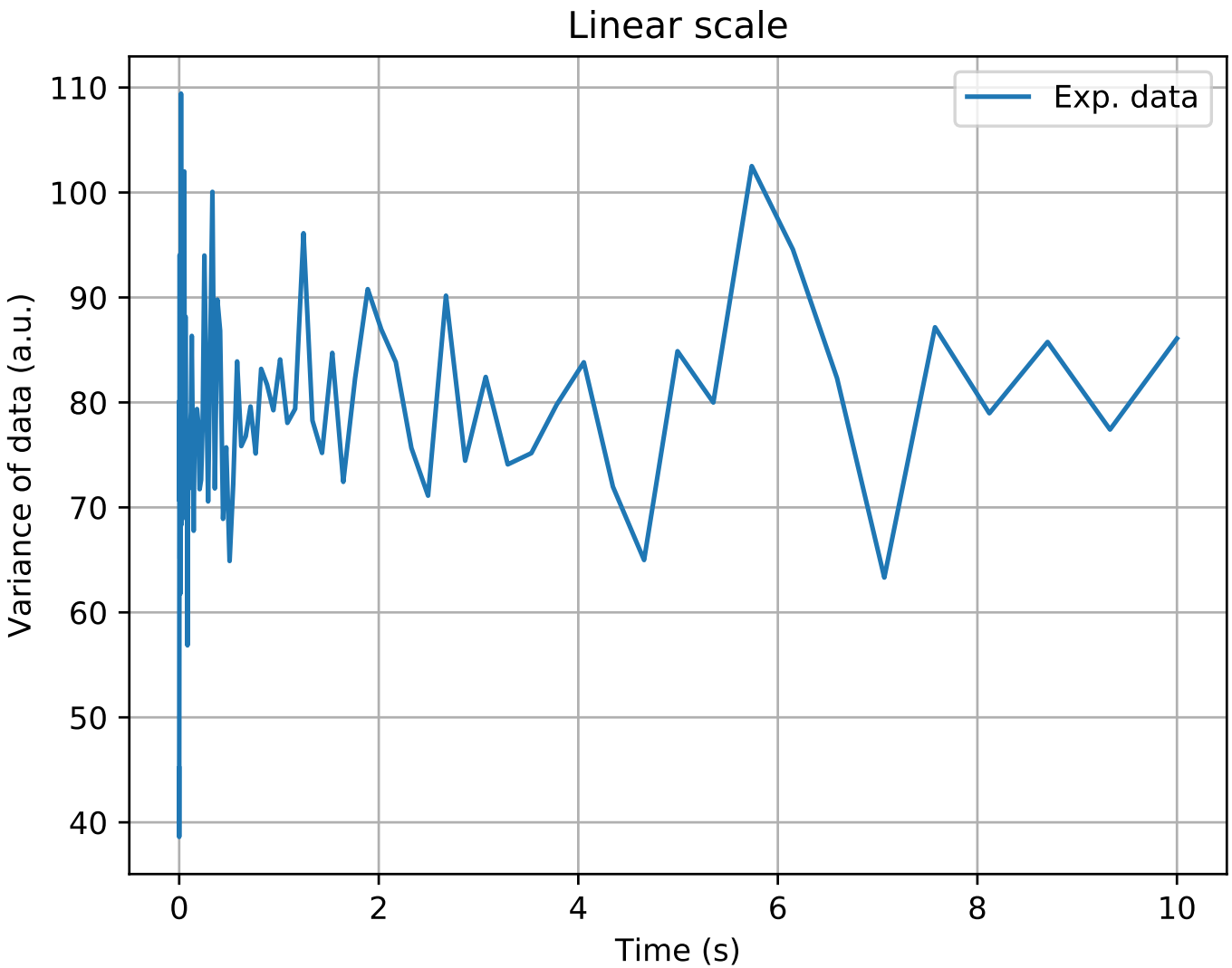

```
plt.semilogy(X, Y)
```

Log-Lin plots

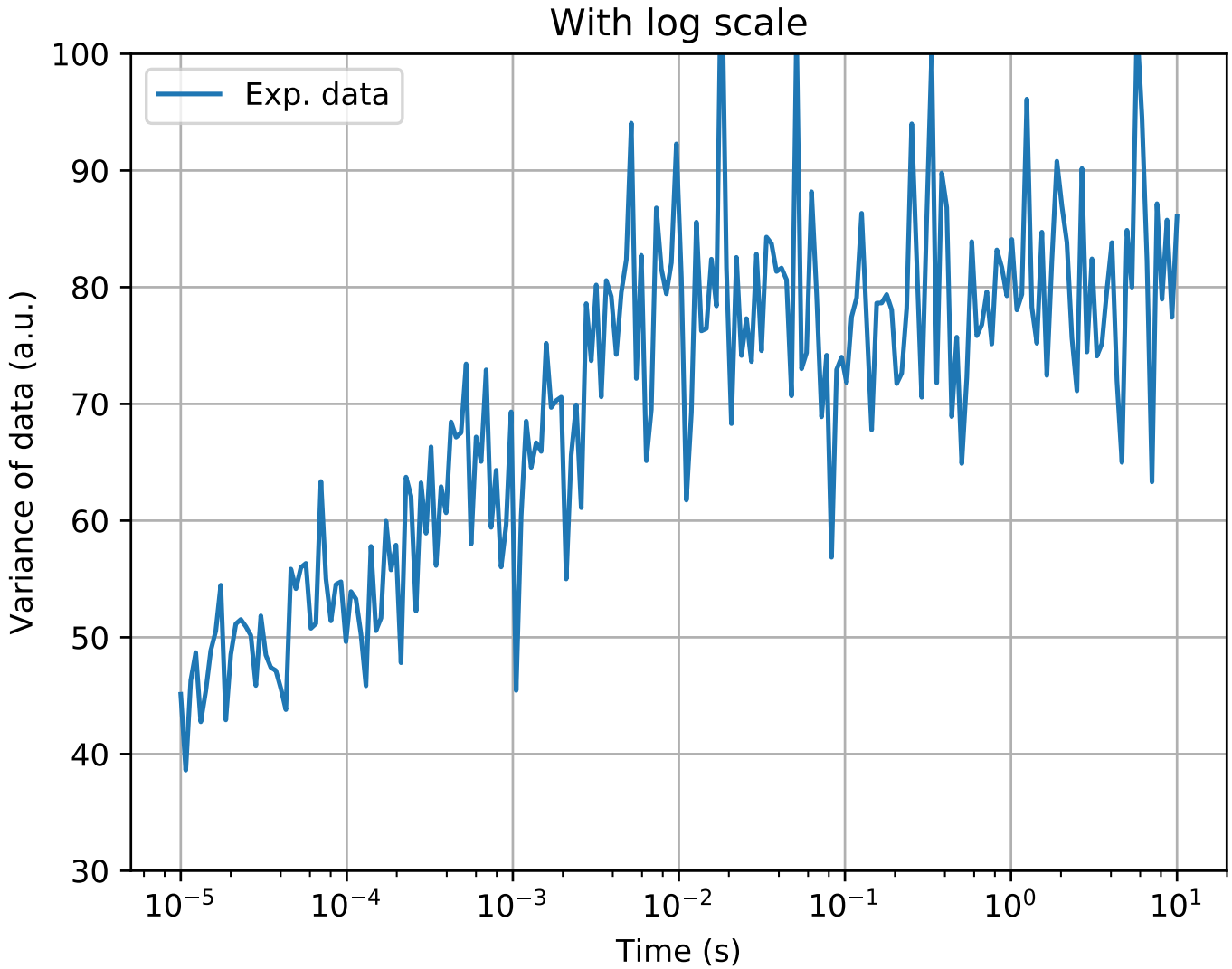



```
plt.semilogy(X, Y)
```

Log-Lin plots

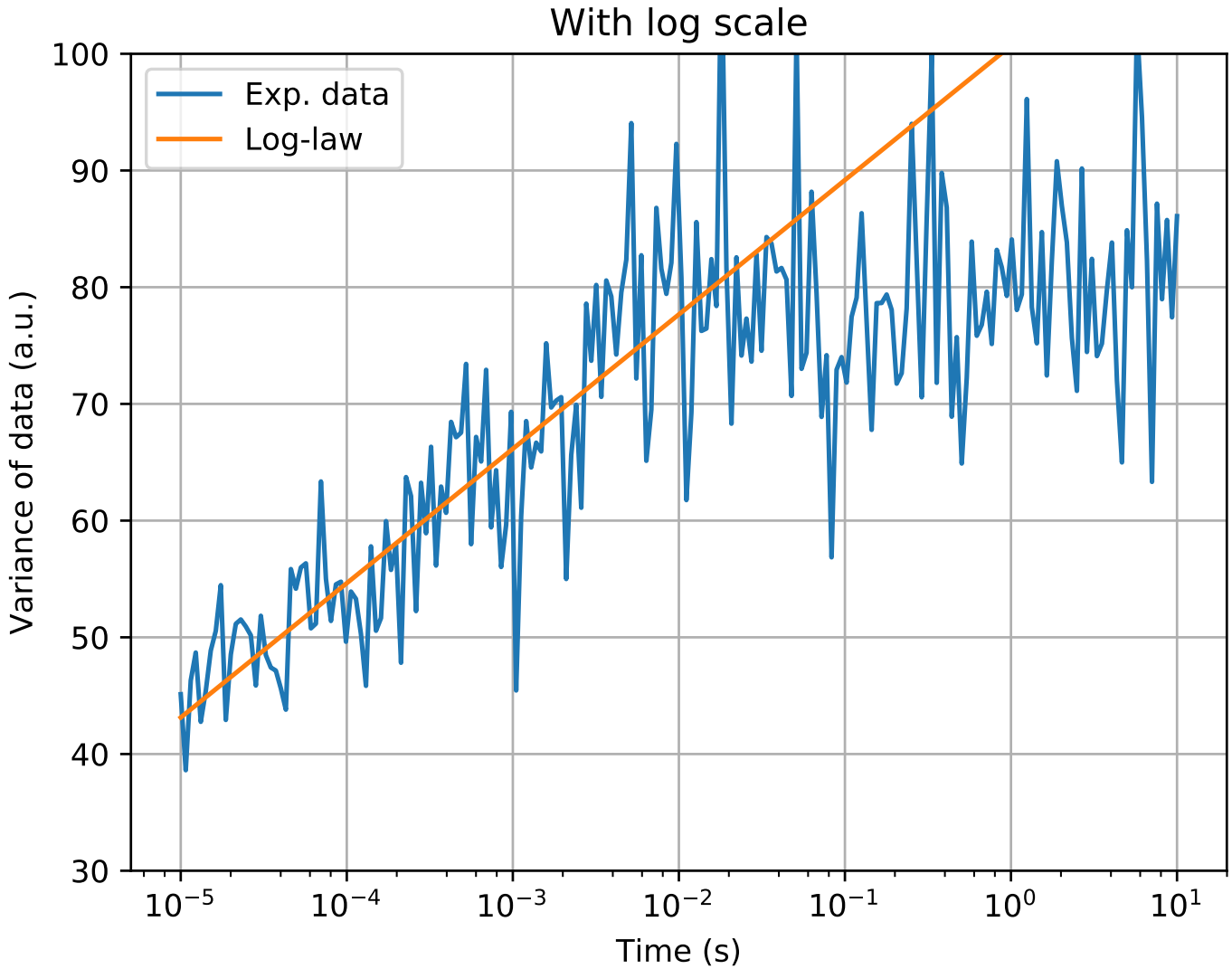


Log-Lin plots



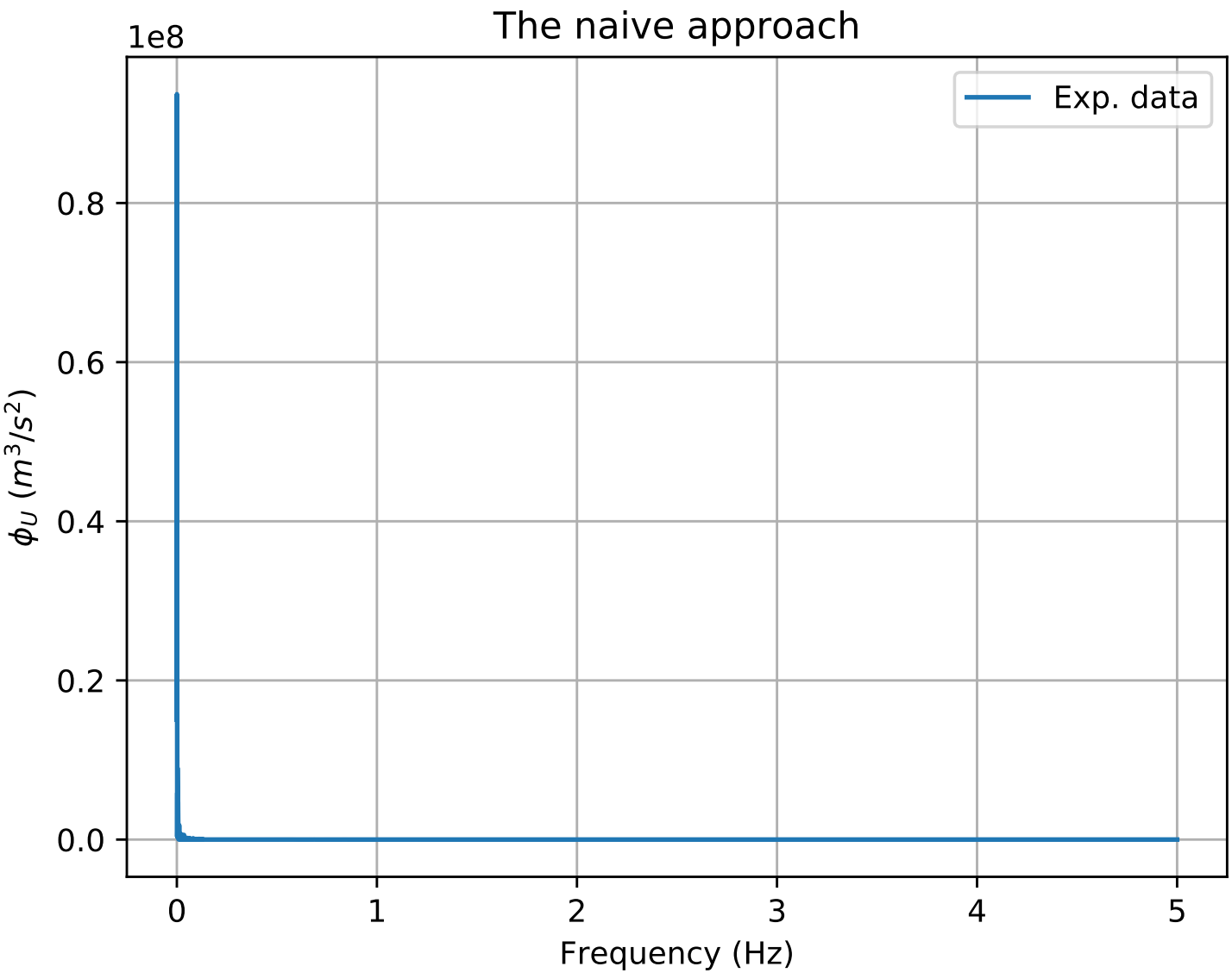

```
plt.semilogx(X, Y)
```

Log-Lin plots

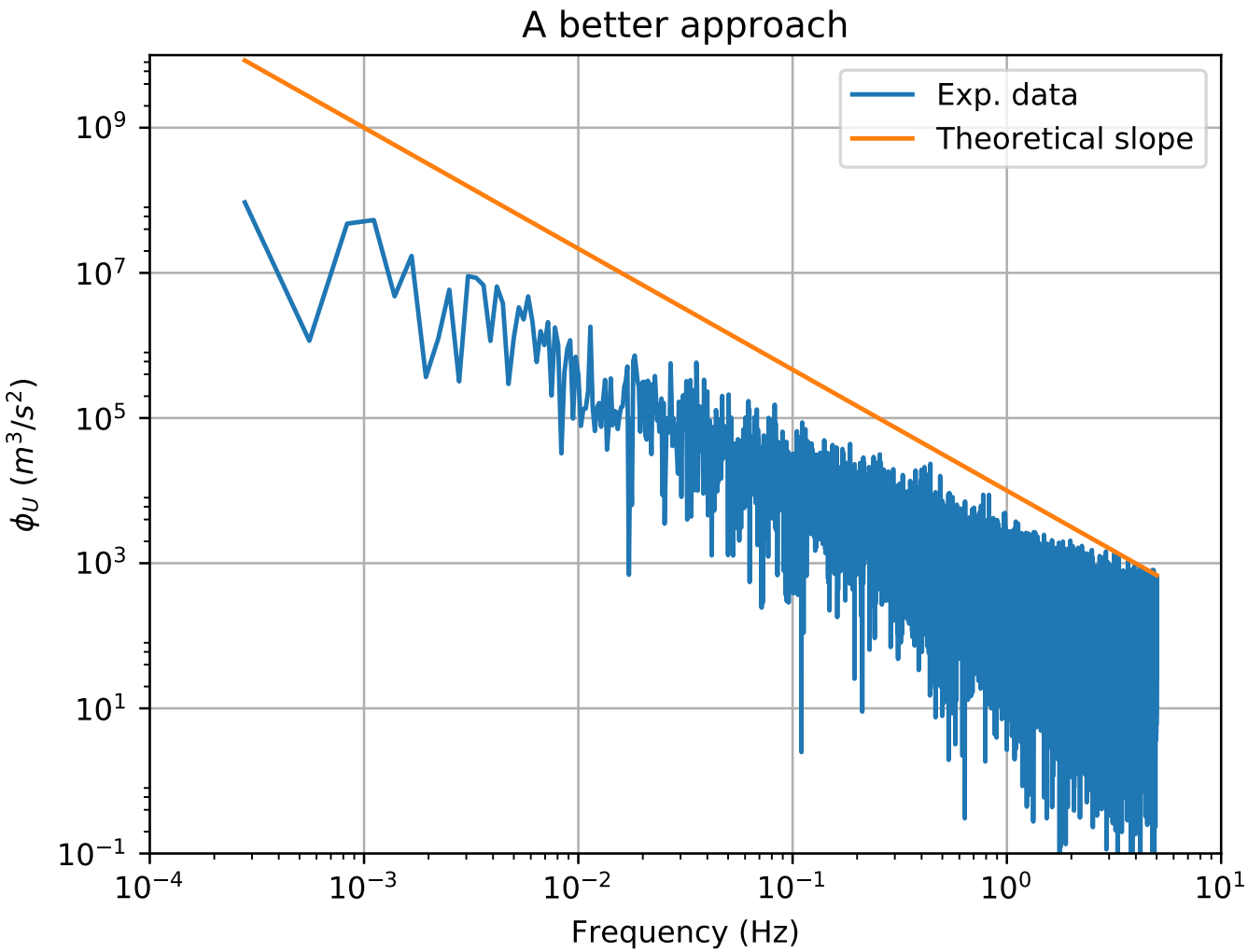



```
plt.semilogx(X, Y)
```

Log-Log plots

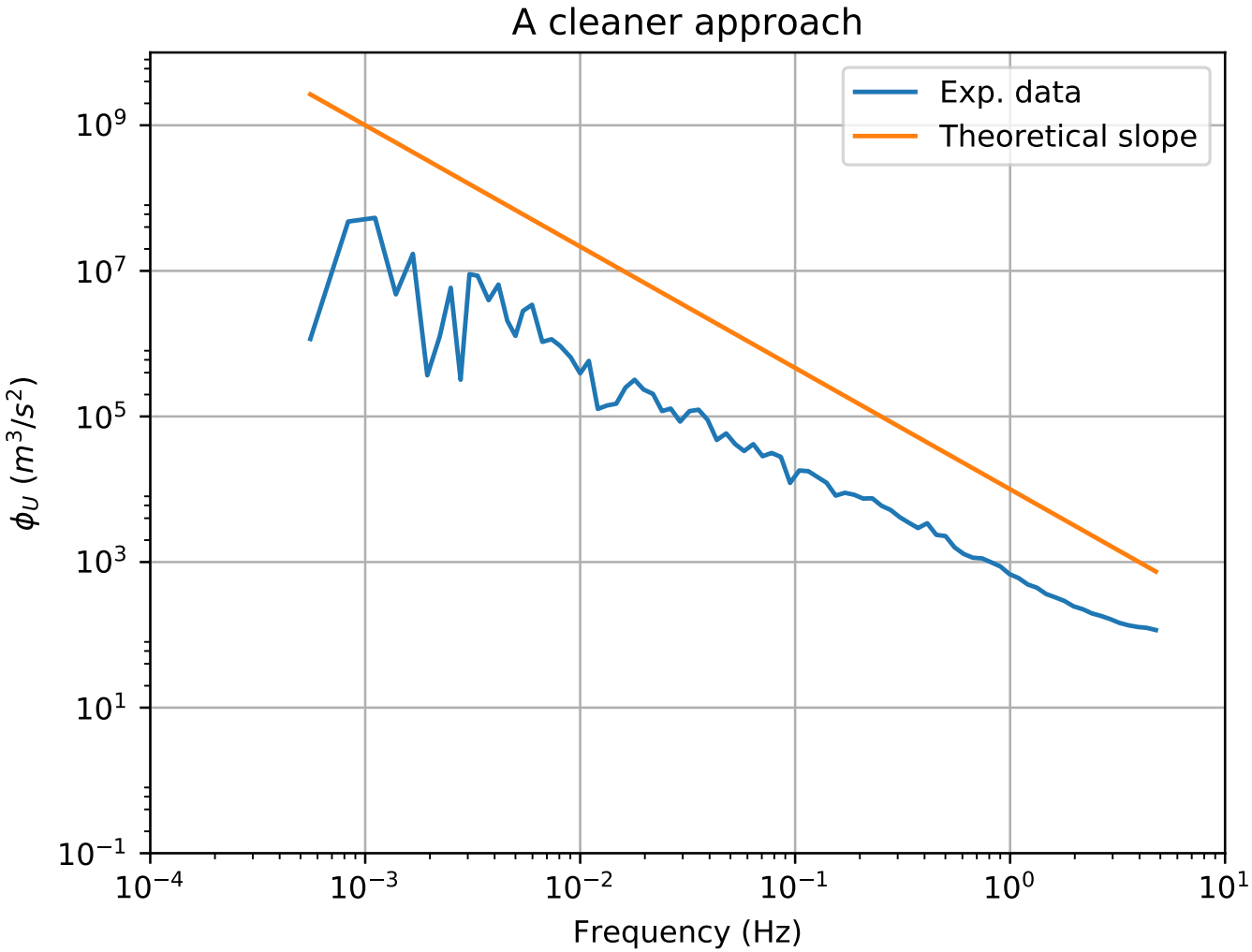


Log-Log plots



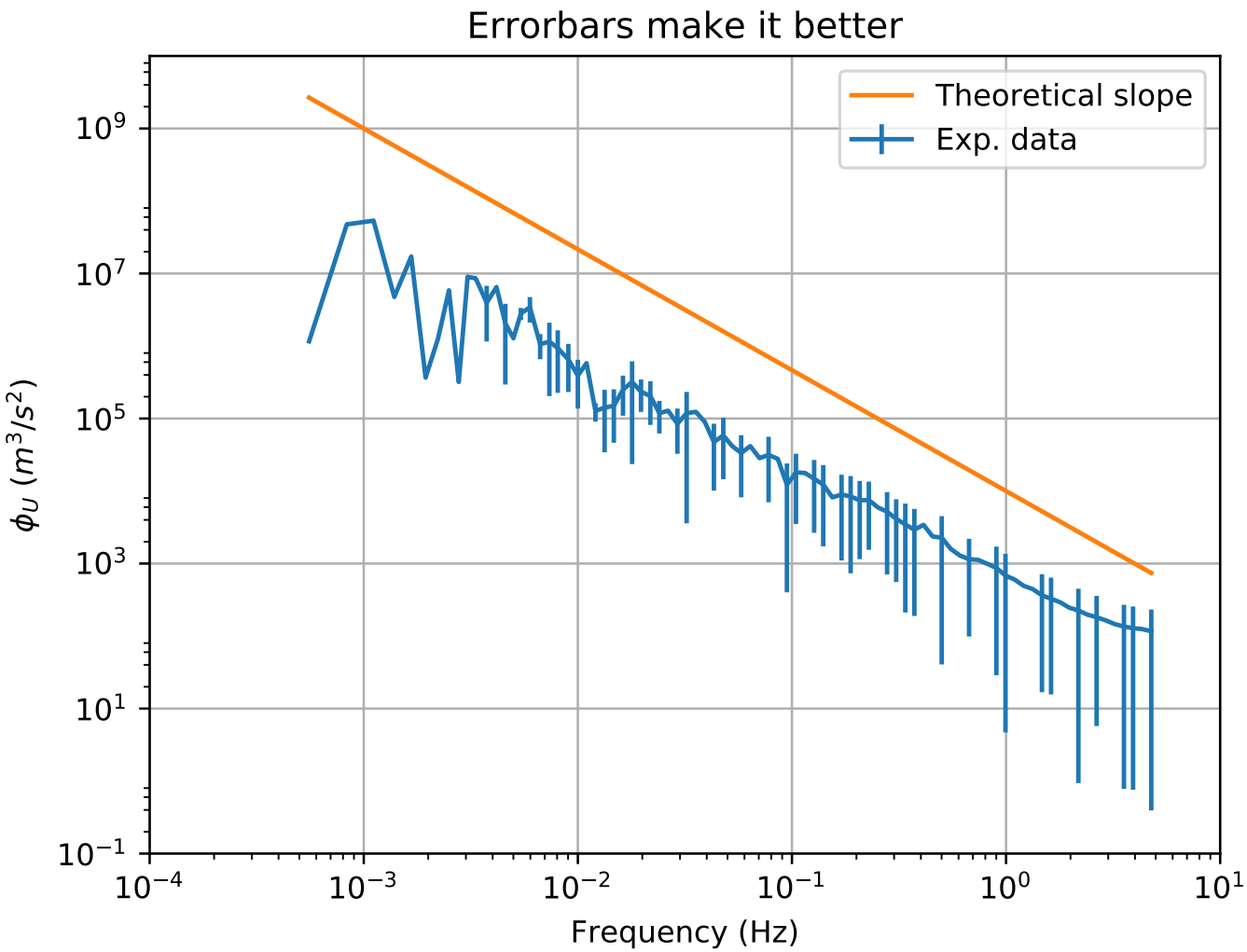

```
plt.loglog(X, Y)
```

Binning



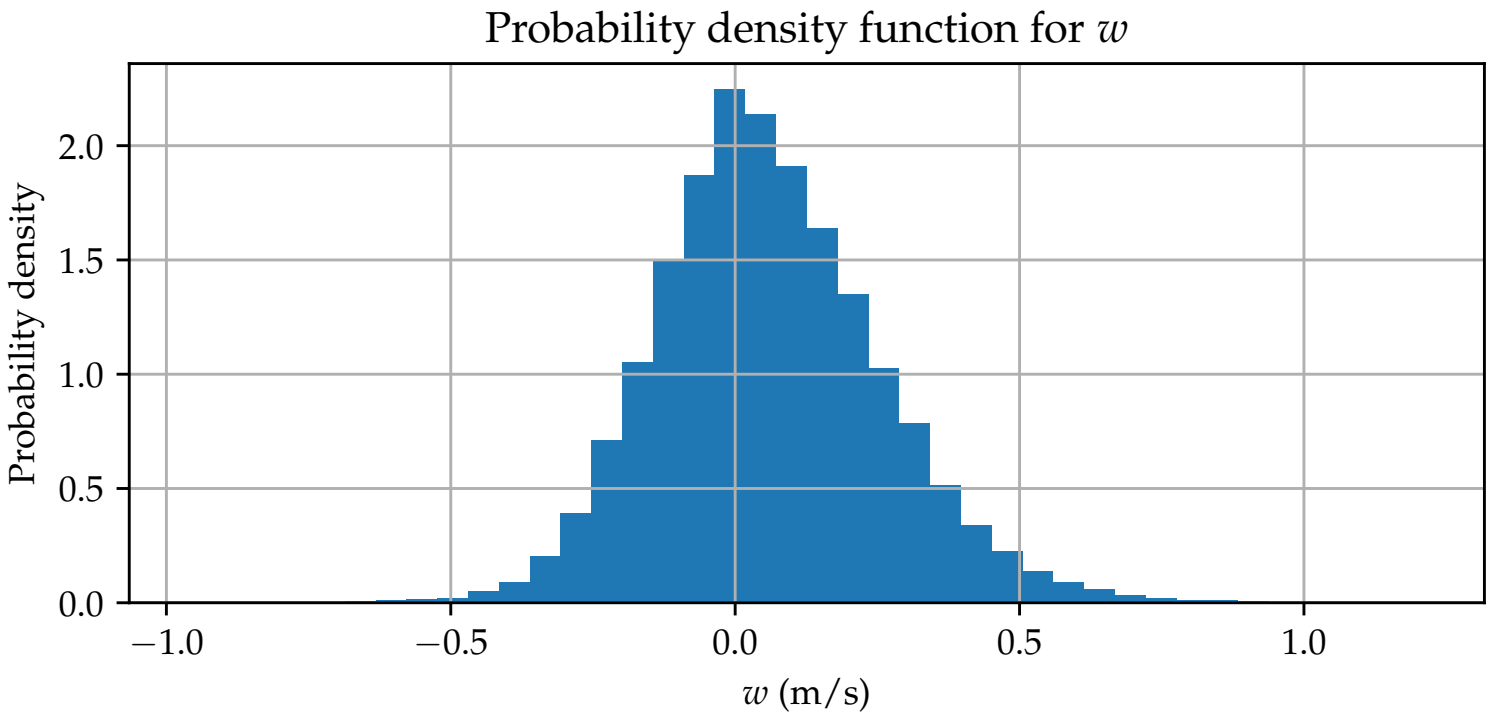

```
plt.loglog(X, Y)
```

Binning and errorbars




```
plt.errorbar(X, Y, yerr=yerrors, xerr=xerrors)
plt.yscale('log')
plt.xscale('log')
```

Creating a Prob. Dens. Function (and using LaTeX)

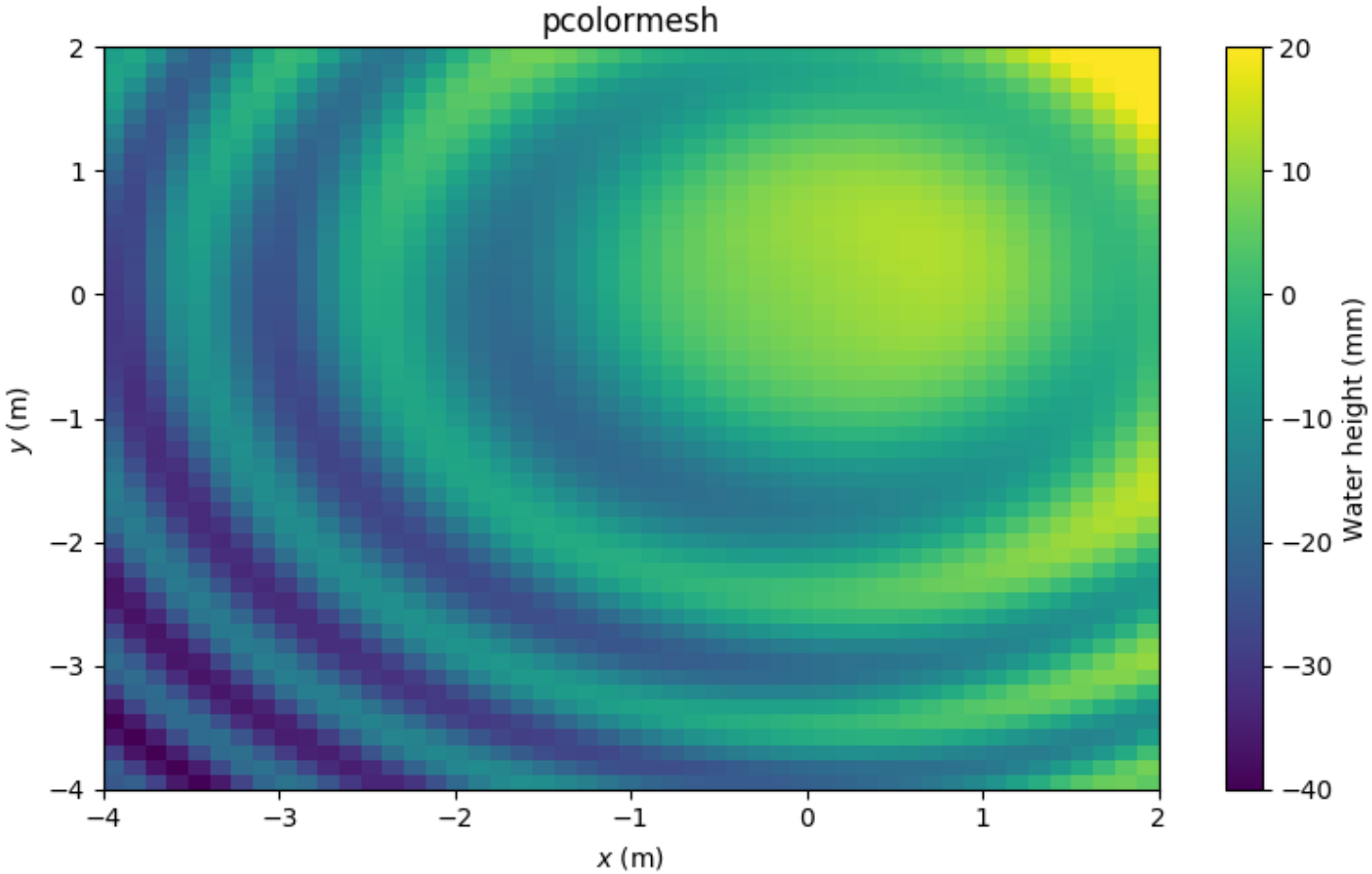



```
from matplotlib import rc
rc('font', family='serif', serif='Palatino')
rc('text', usetex=True)
PDF, bin_edges = np.histogram(W, bins=40, density=True)
bin_mid = (bin_edges[:-1]+bin_edges[1:])/2
plt.bar(bin_mid,PDF, width=width)
```

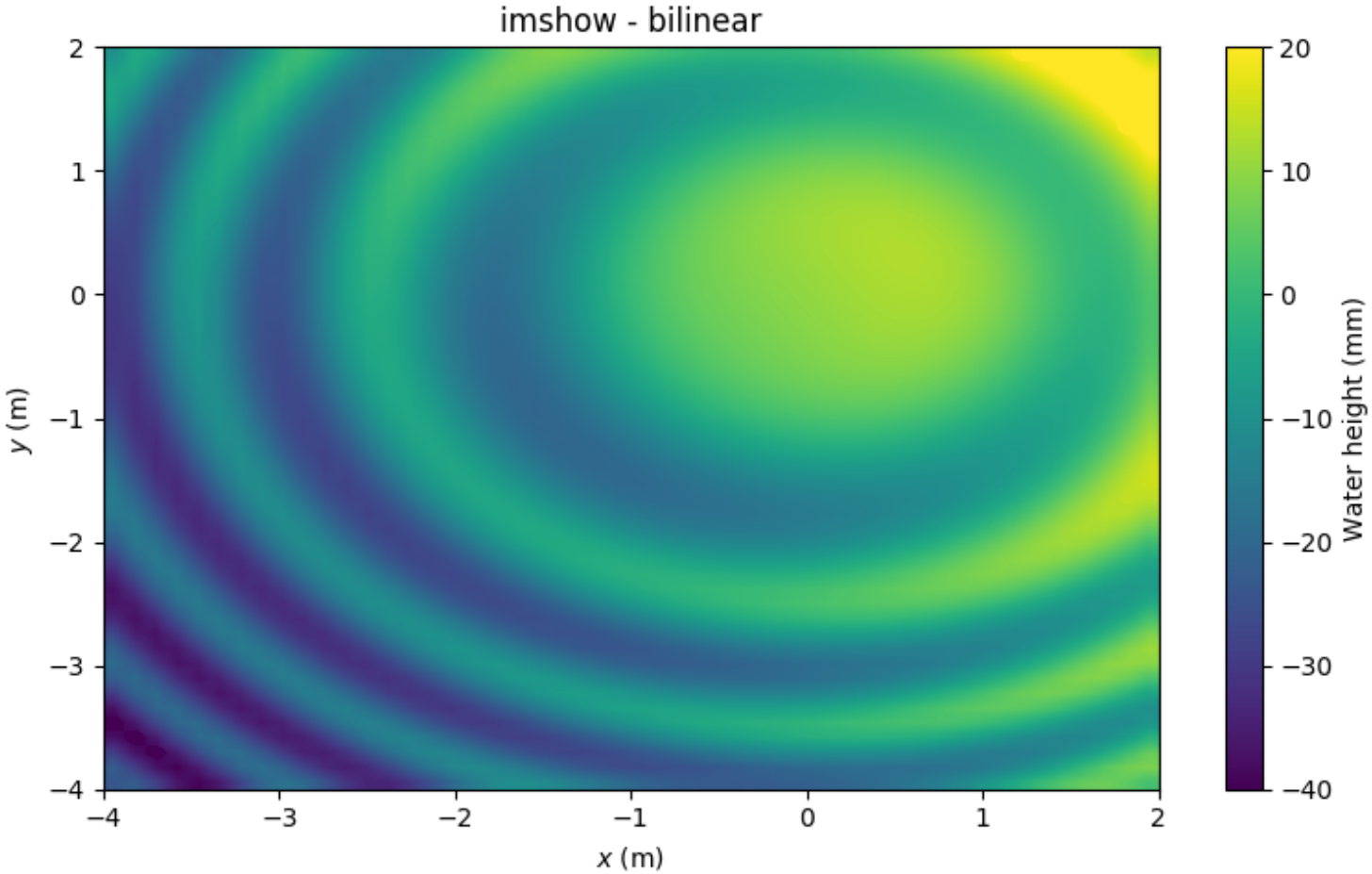
Plotting 2D data

Two more things to worry about when plotting 2D data

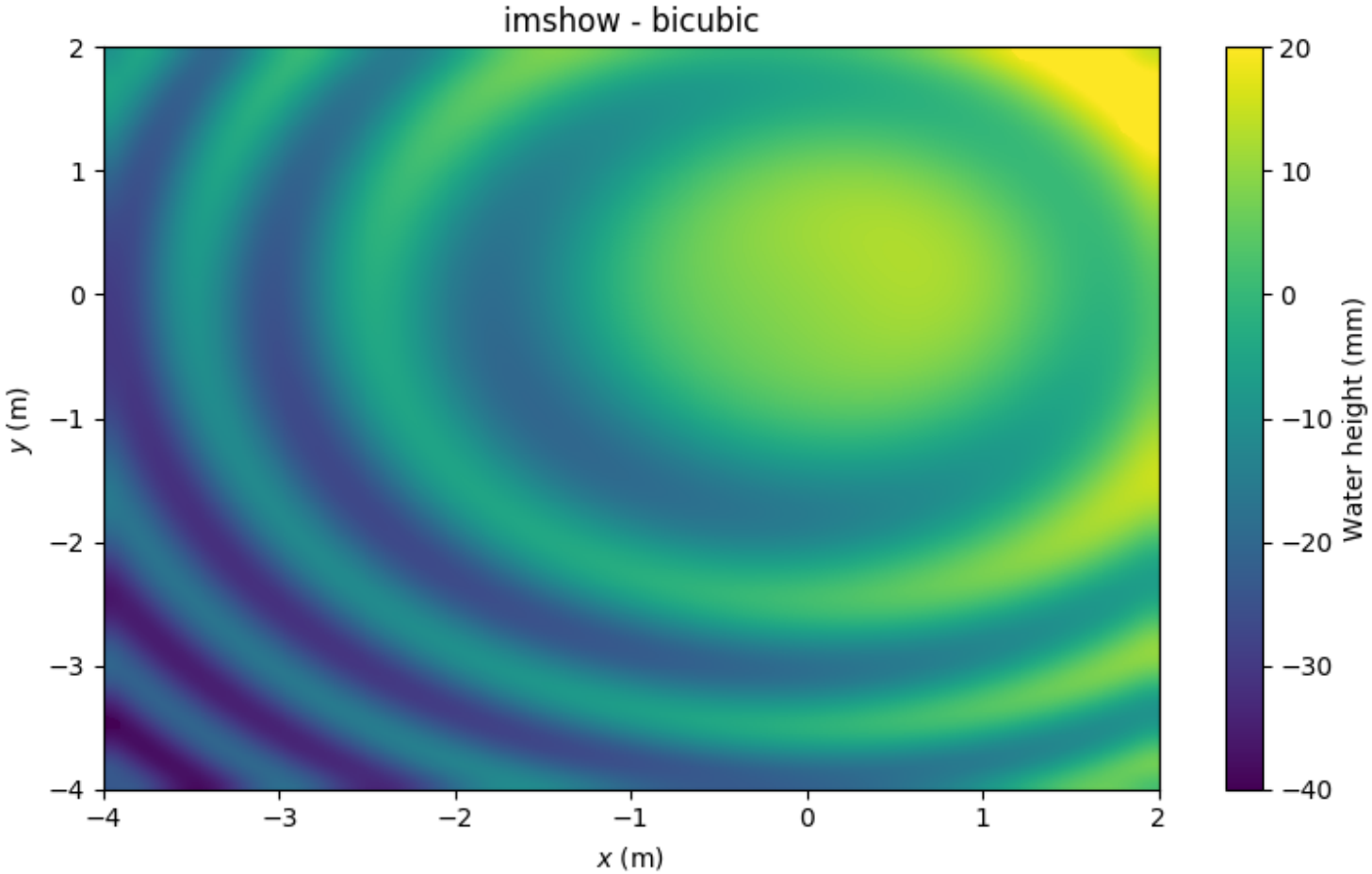
- Type of plot
- Colorbar




```
plt.pcolormesh(x, y, data, vmin=-40, vmax=20)
plt.colorbar(label='Water height (mm)')
```

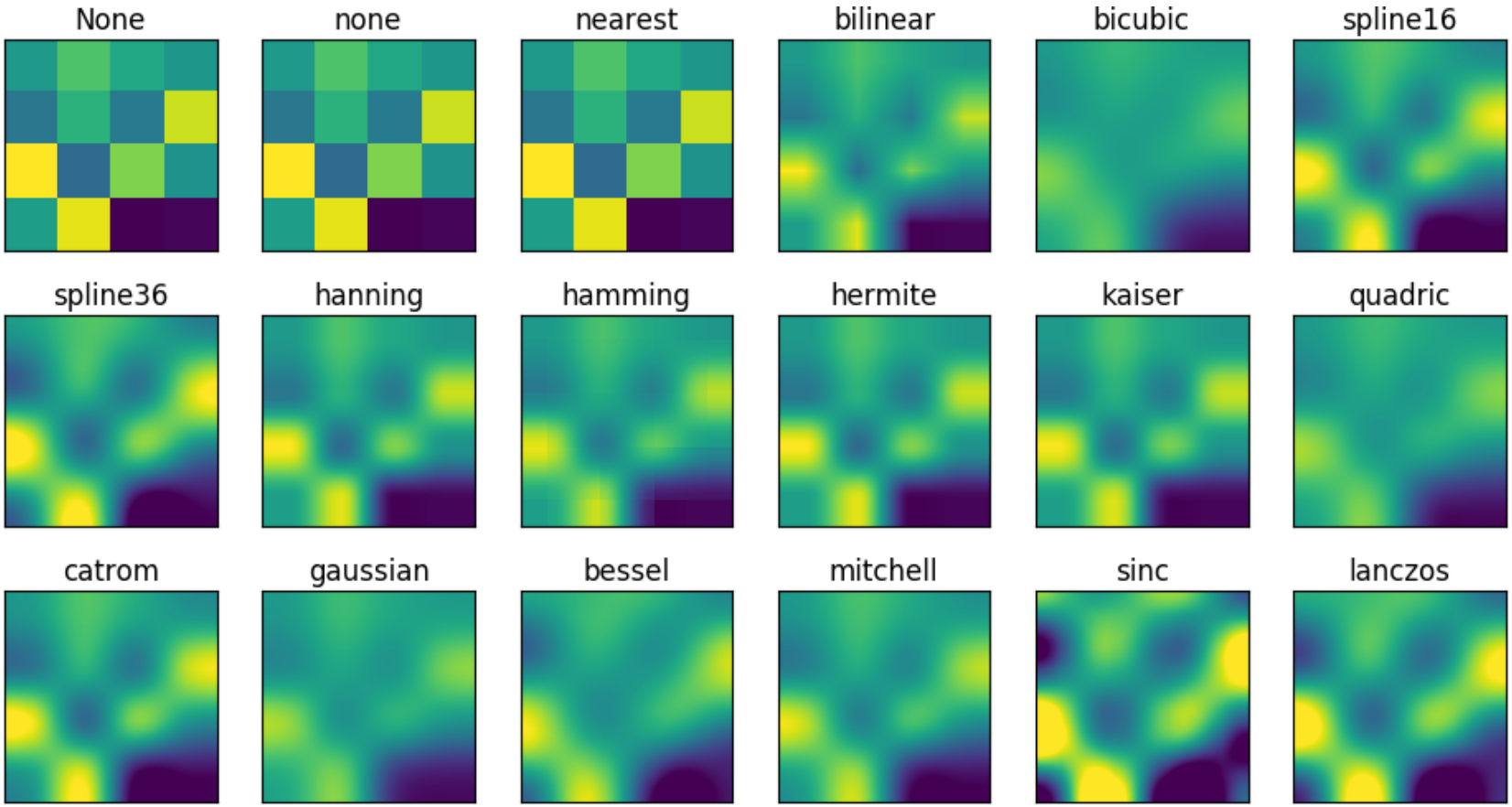



```
plt.imshow(data, interpolation='bilinear', origin='lower',
  aspect='auto', extent=[x.min(), x.max(), y.min(), y.max()],
  vmin=-40, vmax=20)
plt.colorbar(label='Water height (mm)')
```

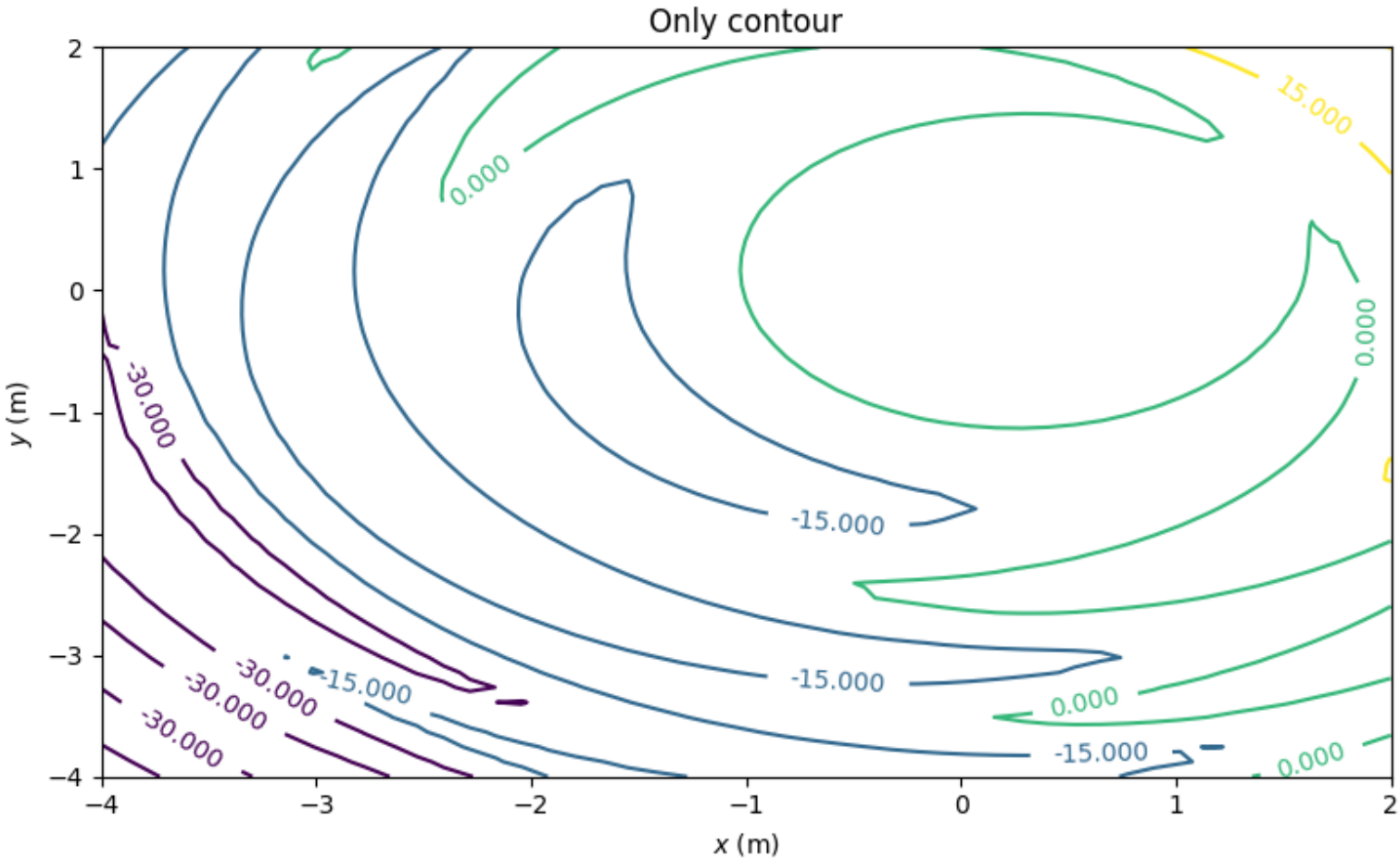



```
plt.imshow(data, interpolation='bicubic', origin='lower',
  aspect='auto', extent=[x.min(), x.max(), y.min(), y.max()],
  vmin=-40, vmax=20)
plt.colorbar(label='Water height (mm)')
```

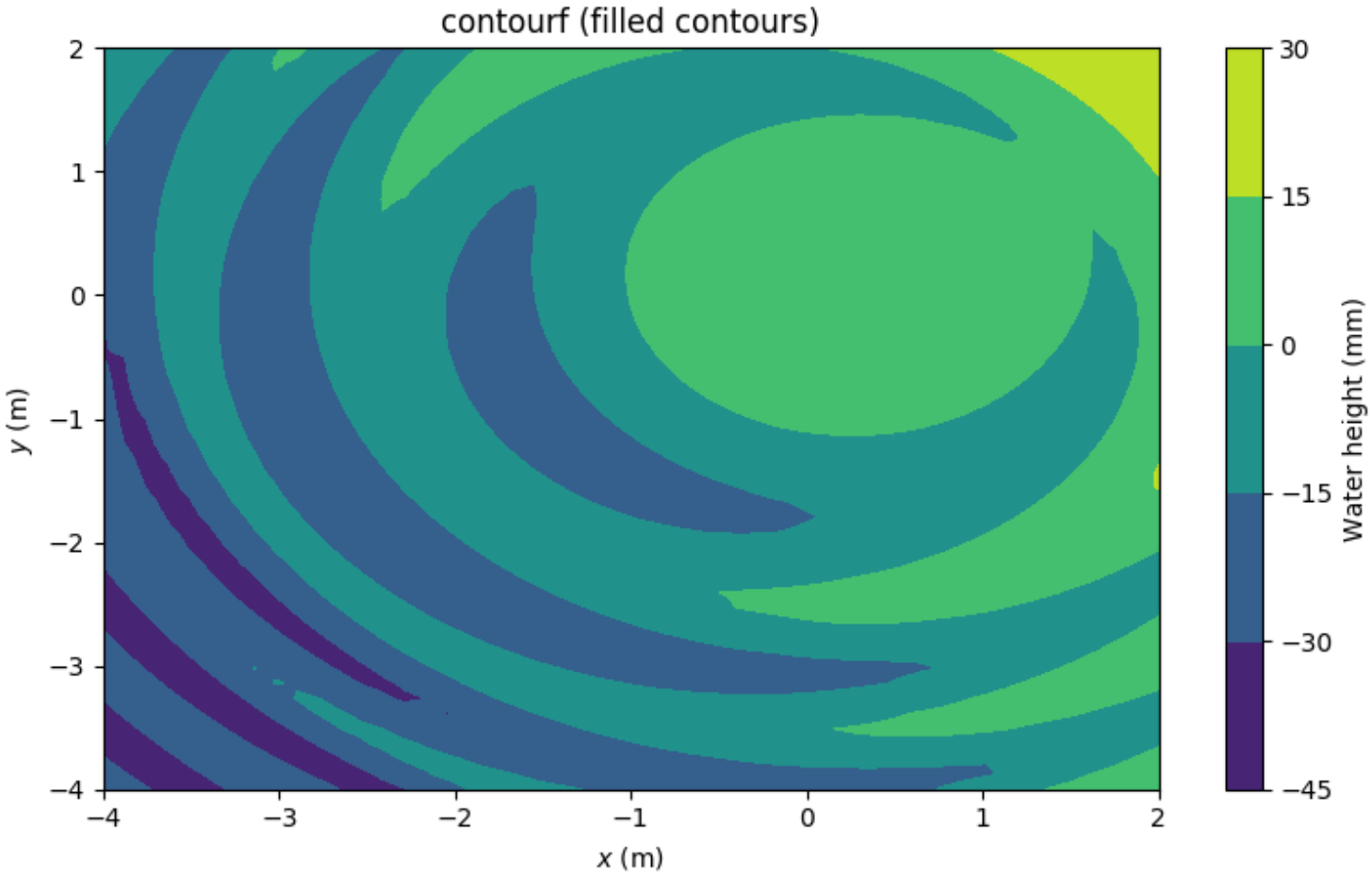
Sometimes the interpolation can make a big difference



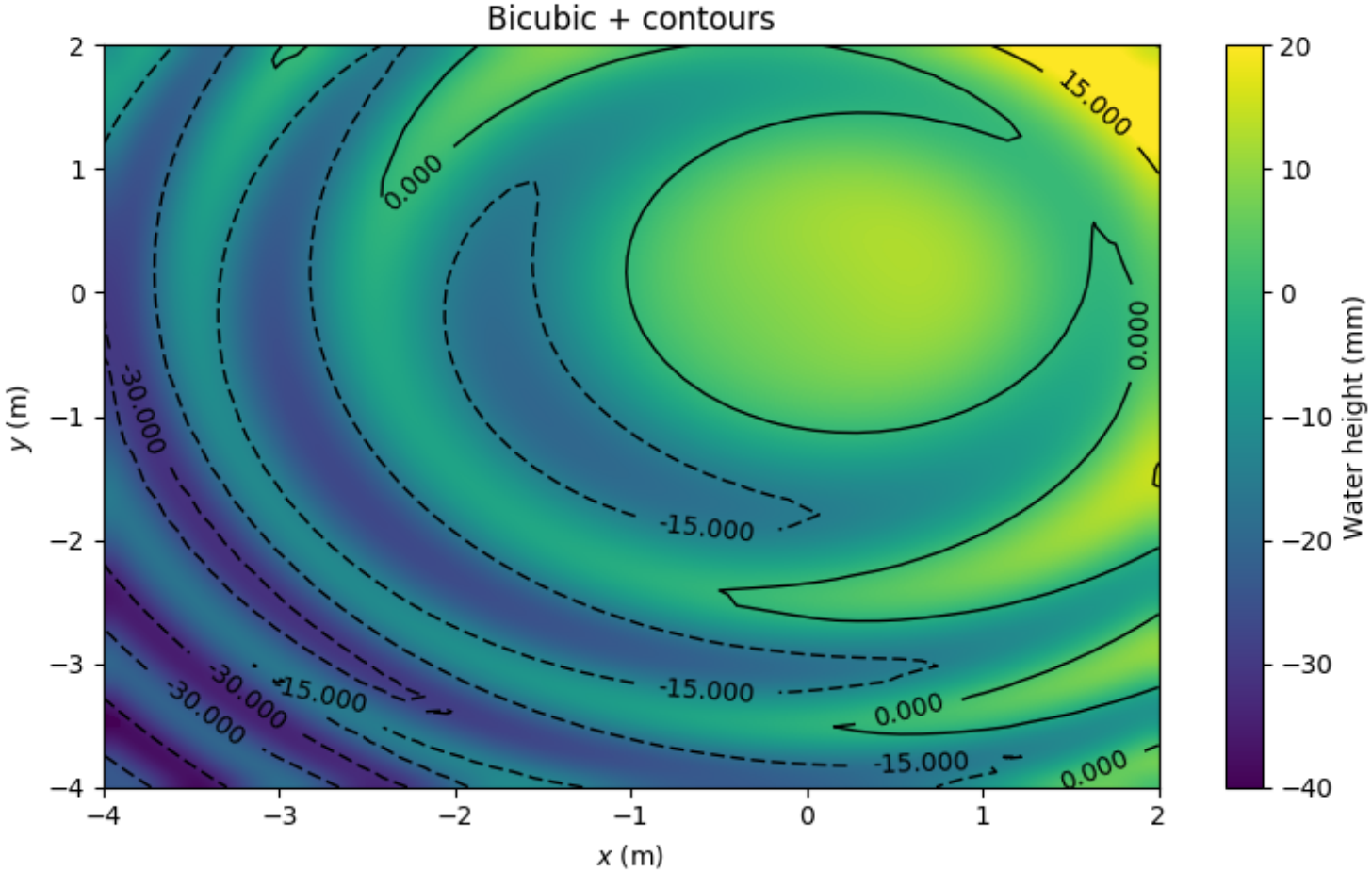
Taken from *Matplotlib Gallery* (2017).



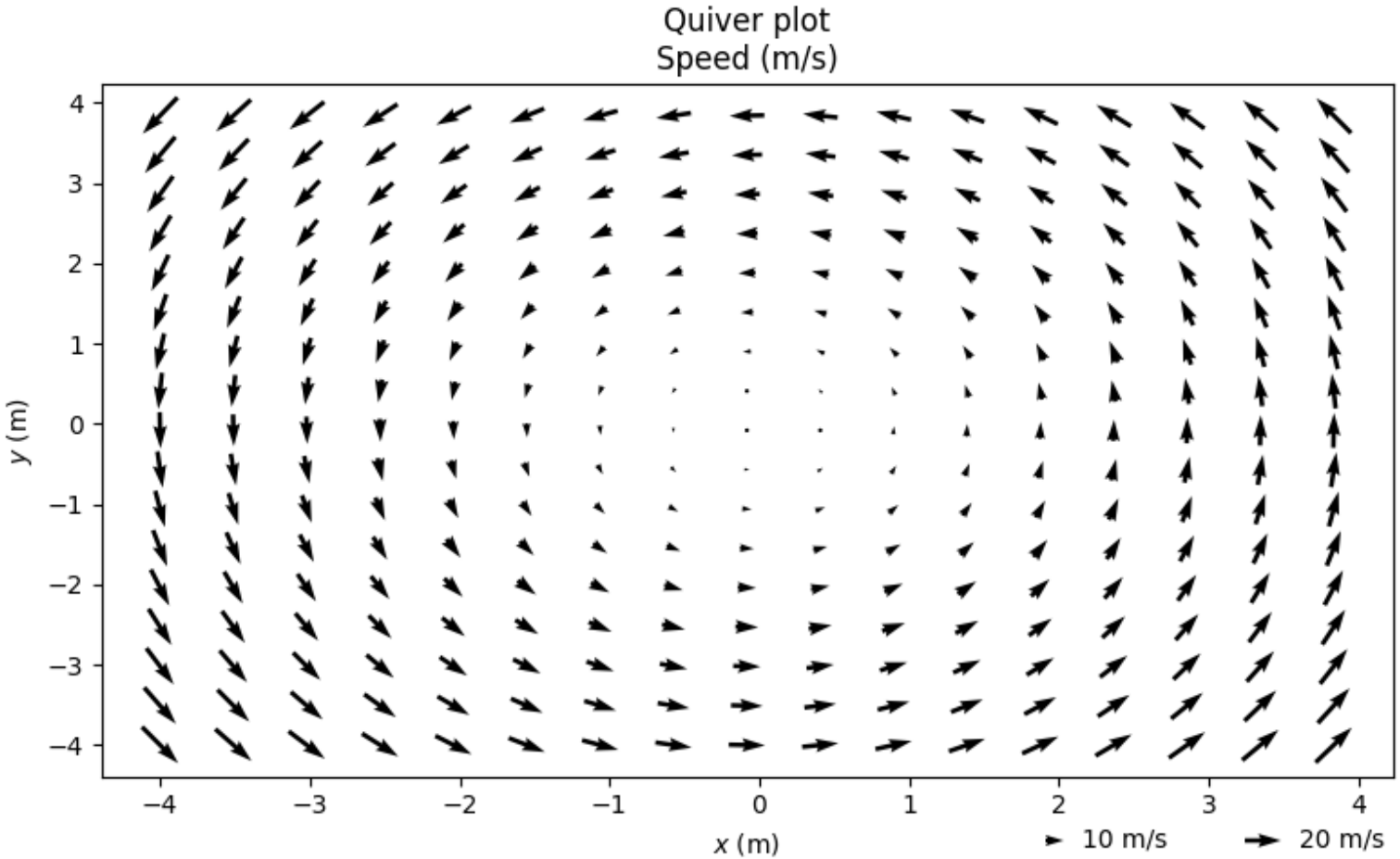

```
CS = plt.contour(x, y, z, 5)
plt.clabel(CS, inline=1, fontsize=10)
```



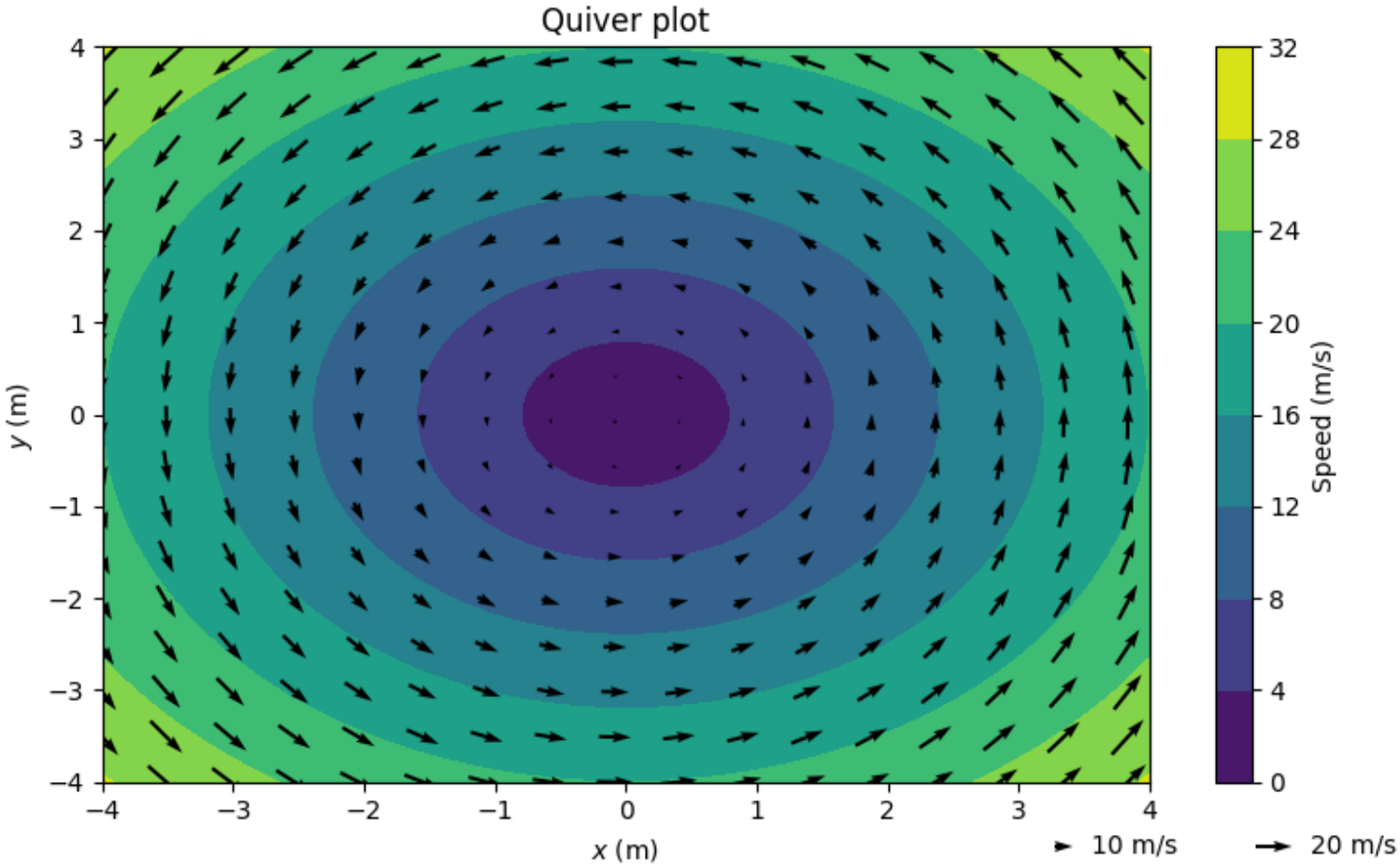

```
plt.contourf(x, y, z, 5)
plt.colorbar(label='Water height (mm)')
```



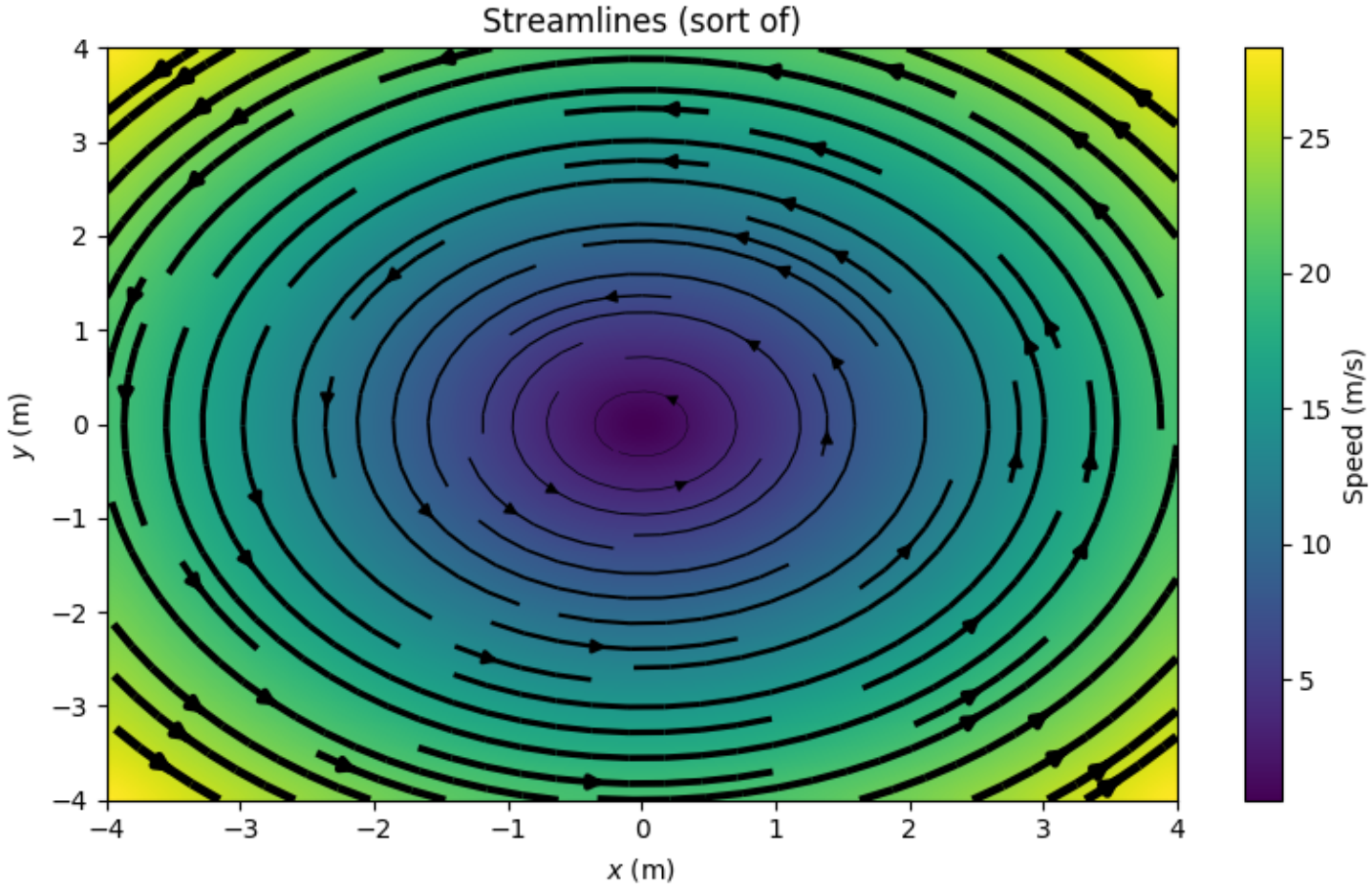

```
plt.imshow(z, interpolation='bicubic', origin='lower',
    extent=[-4,2,-4,2], aspect='auto', vmin=-40, vmax=20)
plt.colorbar(label='water height (mm)')
cs = plt.contour(x, y, z, 5, colors='black', linewidths=1.0)
plt.clabel(cs, inline=1, fontsize=10)
```



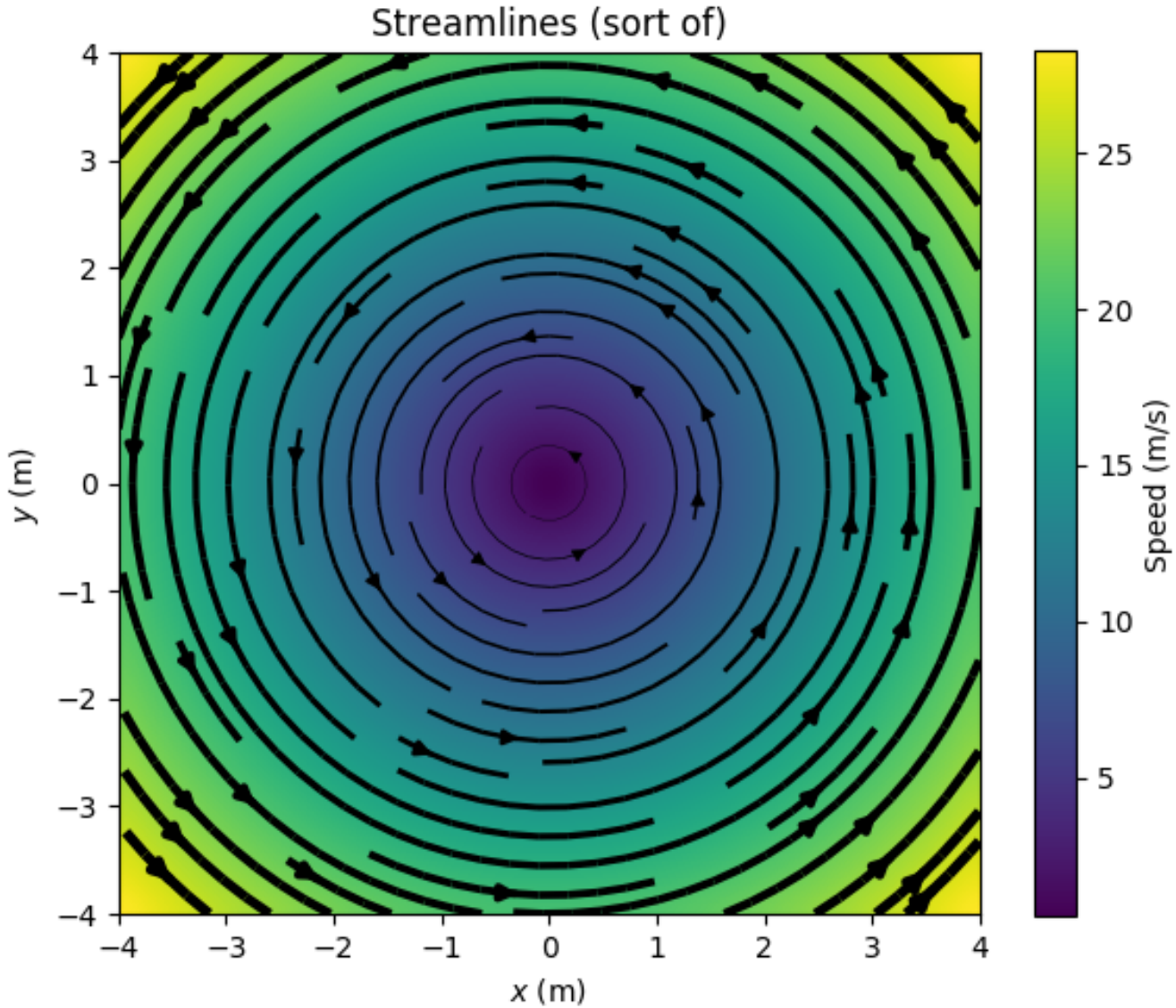

```
Q = plt.quiver(x, y, U, V, pivot='mid',
    scale_units='dots', scale=1)
plt.quiverkey(Q, 0.9, 0.05, 20, r'20 m/s',
    labelpos='E', coordinates='figure')
plt.quiverkey(Q, 0.75, 0.05, 10, r'10 m/s',
    labelpos='E', coordinates='figure')
```




```
plt.contourf(x,y,speed)
plt.colorbar(label='Speed␣(m/s)')
Q = plt.quiver(x, y, U, V, pivot='mid',
    scale_units='dots', scale=1)
plt.quiverkey(Q, 0.9, 0.05, 20, r'20␣m/s',
    labelpos='E', coordinates='figure')
plt.quiverkey(Q, 0.75, 0.05, 10, r'10␣m/s',
    labelpos='E', coordinates='figure')
```




```
plt.imshow(speed, origin='power', interpolation='bicubic',
    extent=[x.min(),x.max(),y.min(),y.max()], aspect='auto')
plt.colorbar(label='Speed␣(m/s)')
plt.streamplot(x,y,U,V, linewidth=nspeed, color='k')
```



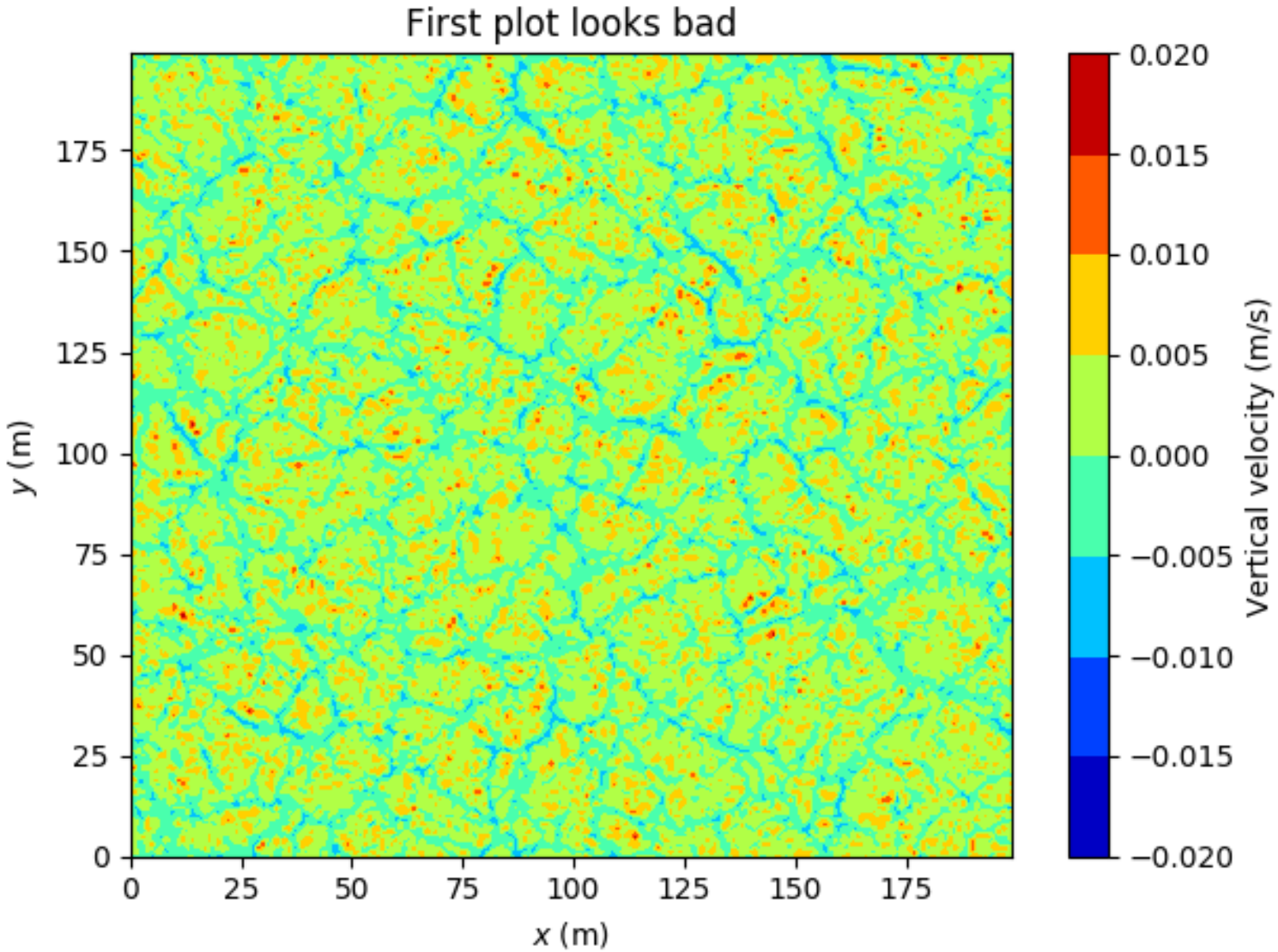

```
plt.imshow(speed, origin='power', interpolation='bicubic',
    extent=[x.min(),x.max(),y.min(),y.max()], aspect=1)
plt.colorbar(label='Speed␣(m/s)')
plt.streamplot(x,y,U,V, linewidth=nspeed, color='k')
```

The color scale is important

- Prefer a scale that has continuous color perception
- Scale depends on the characteristic of your data (clear +/- dichotomy?)
- Plotting log also might be useful

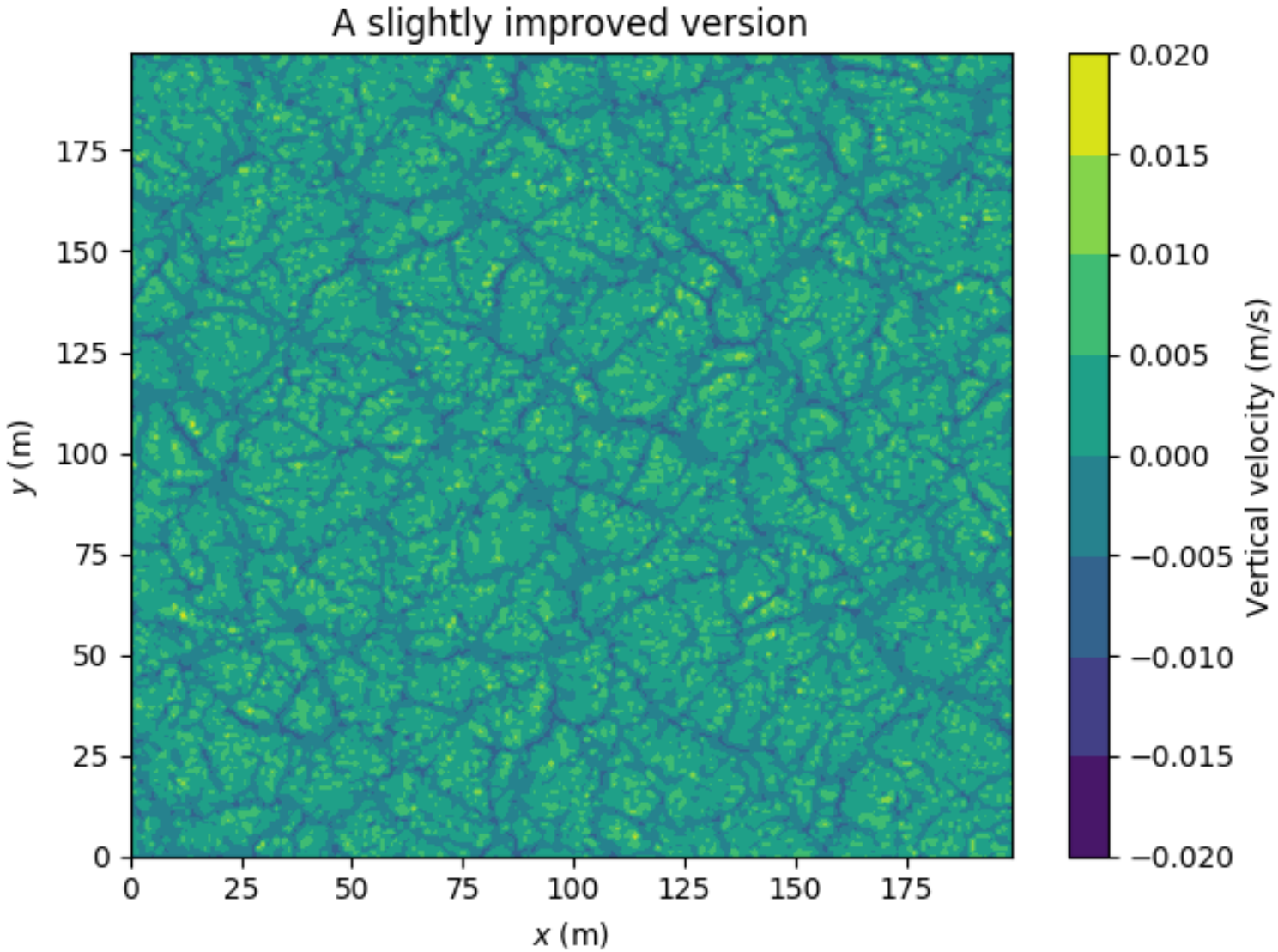
Different colormaps animation

Jet colormap



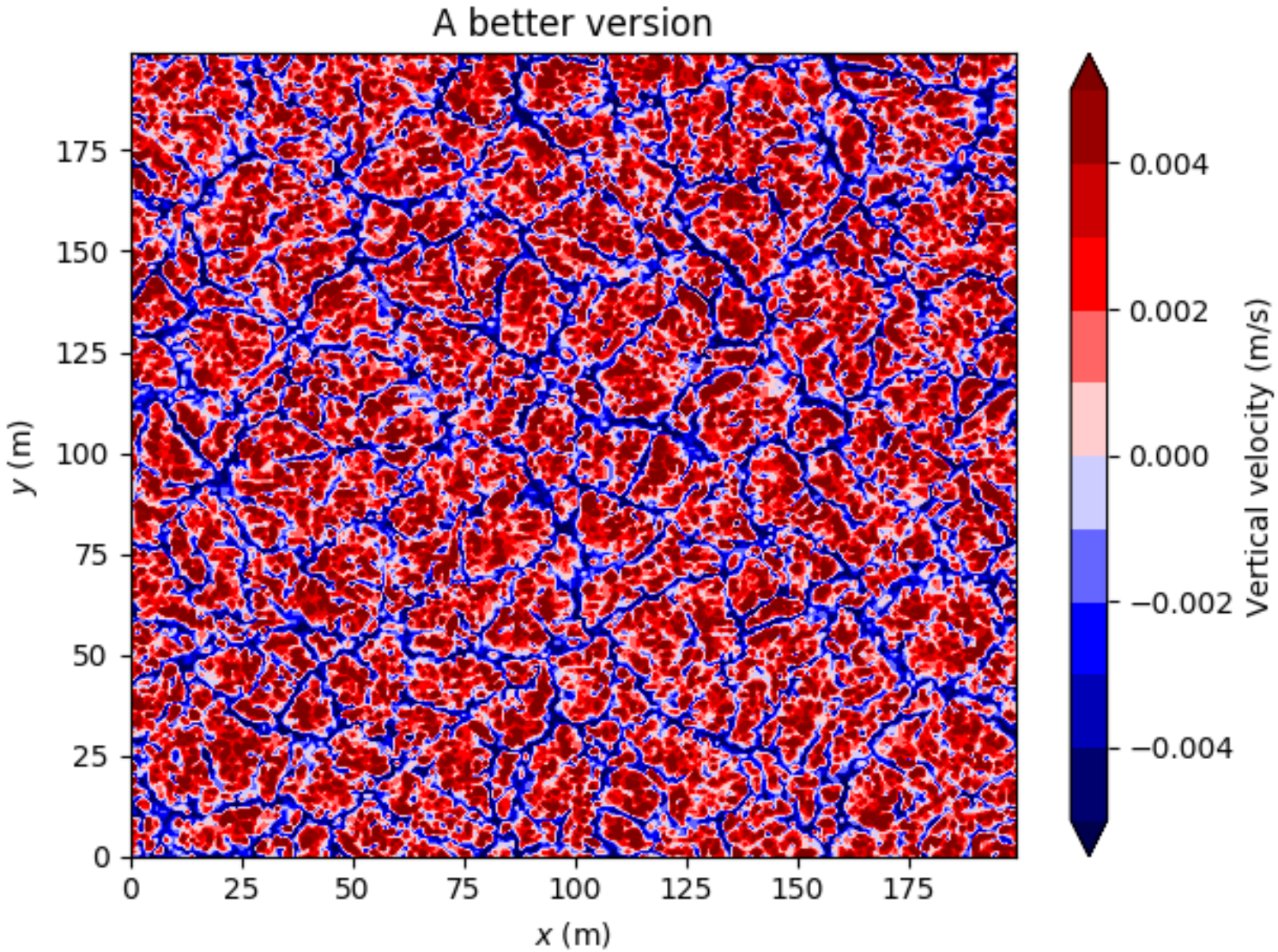

```
plt.contourf(w0, cmap='jet')
```

Viridis colormap



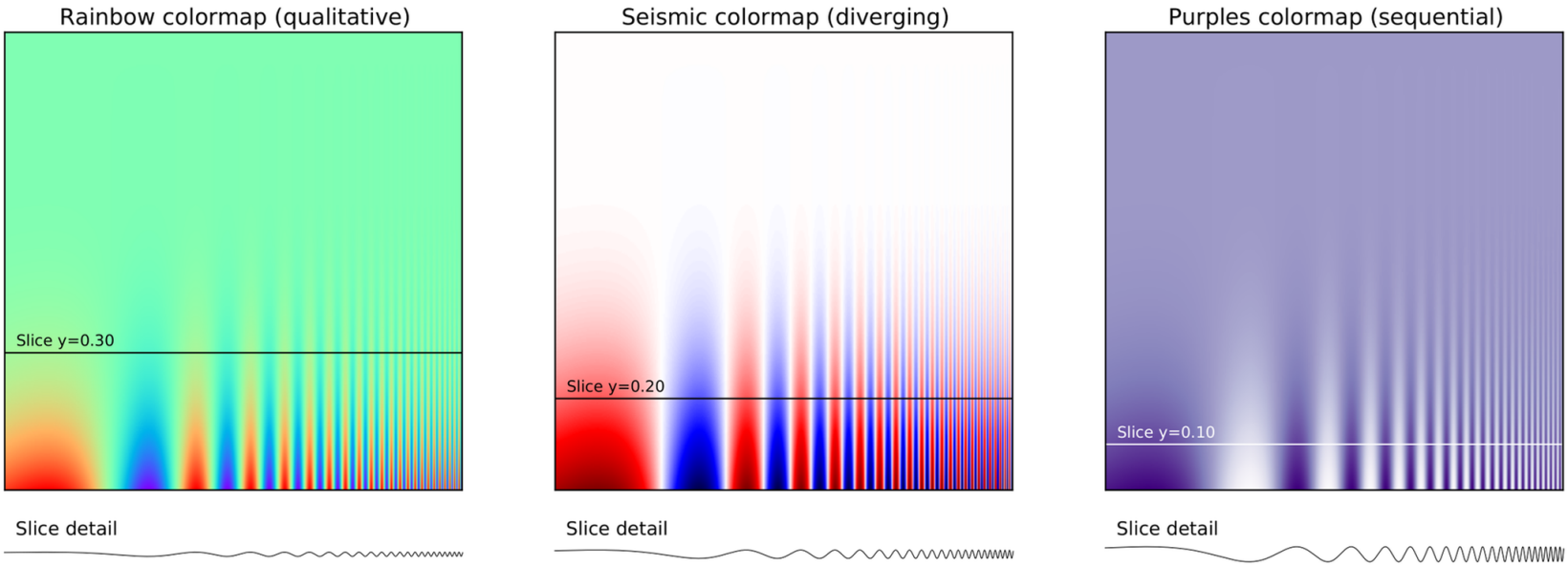

```
plt.contourf(w0, cmap='viridis')
```

Symmetric (divergent) colormap




```
plt.contourf(w0, cmap='seismic')
```

Some features might be easier to see



Taken from Rougier, Droettboom, and Bourne, (2014).

Temperature animation

```
# Data reading happens before this
fig = plt.figure()
ims = []

for it in range(300):
    im = plt.imshow(T[it], animated=True, cmap='viridis',
      origin='lower', extent=[0, 1e3, 0, 1e3],
      vmin=T.min(), vmax=T.max(), interpolation='bicubic')
    tx = im.axes.text(0, 0, 'Timestep: {}'.format(it),
      color='black')
    ims.append([im, tx])

plt.colorbar(label='T (K)')
plt.xlabel('$x$ (m)')
plt.ylabel('$y$ (m)')
plt.title('Temperature at the top of a convective ocean')
plt.tight_layout()
ani = animation.ArtistAnimation(fig, ims, interval=50,
  blit=True, repeat_delay=1000)
ani.save('T_surf.mp4', dpi=150)
```

Oil animation

```
# Data reading happens before this
fig = plt.figure()
ims = []
for it in range(300):
    im = plt.imshow(C[it], animated=True, cmap='plasma',
      origin='lower', extent=[0, 1e3, 0, 1e3],
      vmin=0., vmax=10., interpolation='bicubic')
    tx = im.axes.text(0, 0, 'Timestep: {}'.format(it),
      color='white')
    ims.append([im, tx])

plt.colorbar(label='Oil Concentration (kg/m$^3$)')
plt.xlabel('$x$ (m)')
plt.ylabel('$y$ (m)')
plt.title('Concentration at the top of a convective ocean')
plt.tight_layout()
ani = animation.ArtistAnimation(fig, ims, interval=50,
  blit=True, repeat_delay=1000)
ani.save('C_linear.mp4', dpi=150)
```

Oil animation

```
# Data reading happens before this
from matplotlib.colors import LogNorm
fig = plt.figure()
ims = []
for it in range(300):
    im = plt.imshow(C[it], animated=True, cmap='plasma',
      origin='lower', extent=[0, 1e3, 0, 1e3],
      vmin=1.e-4, vmax=10., interpolation='bicubic',
      norm=LogNorm())
    tx = im.axes.text(0, 0, 'Timestep: {}'.format(it),
      color='white')
    ims.append([im, tx])

plt.colorbar(label='Oil Concentration (kg/m$^3$)')
plt.xlabel('$x$ (m)')
plt.ylabel('$y$ (m)')
plt.title('Concentration at the top of a convective ocean')
plt.tight_layout()
ani = animation.ArtistAnimation(fig, ims, interval=50,
  blit=True, repeat_delay=1000)
ani.save('C_log.mp4', dpi=150)
```

Final comments

- Slides and scripts are going to be available
- Written in Python3, but should work with Python2

Matplotlib Gallery. 2017. URL: https://matplotlib.org/gallery.html (visited on 05/05/2017).

Nicolas P. Rougier. *matplotlib-tutorial: Version 1.0*. Aug. 2015. DOI: 10.5281/zenodo.28747. URL: https://doi.org/10.5281/zenodo.28747.

Nicolas P. Rougier, Michael Droettboom, and Philip E. Bourne. "Ten Simple Rules for Better Figures". In: *PLOS Computational Biology* 10.9 (Sept. 2014), pp. 1–7. DOI: 10.1371/journal.pcbi.1003833. URL: https://doi.org/10.1371/journal.pcbi.1003833.