Spyder I(ntegrated) D(evelopment) E(nvironment)

- A python IDE aimed at scientific programmers
- Tries to be a bit like MATLAB
- I care most edit/run interaction
- Editor/debugger interaction in ***compiled*** languages
- Editor/shell interaction in ***interactive*** languages
- http://pythonhosted.org/spyder/
- Runs on proper computers, MacOS X and Windows.

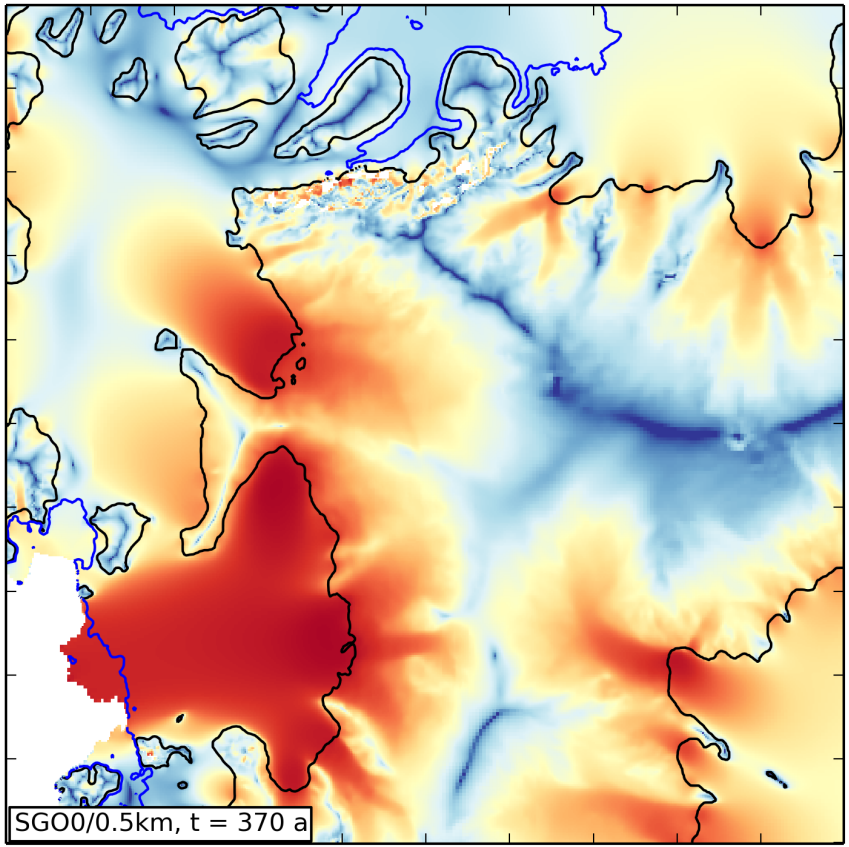
Vanilla Python is terrible for scientific programming

```
"""
Scientific programmers need long arrays of
numbers. Vanilla python has a list. Looks
like an array, but does much more.

Price: slow, and syntax is a bit tedious
"""
import math as ma
nx = 2**20
dx = 1.0/float(nx)
w = 16.0 * ma.pi
xa = [i * dx for i in range(0, nx, 1)]
sinxa = [ma.sin(w * xi) for xi in xa]
"""
sin(w*x) calculation takes about 1 second

R is 10 times faster
> system.time( sinx <- sin(16*pi*x) )
   user  system elapsed
  0.092   0.000   0.092
"""
```

```
"""
Fortunately, there is numpy. Don't even
think about scientific programming without
it
"""

import numpy as np
xb = np.arange(0.0,1.0,dx)
sinxb = np.sin(w*xb)

"""
On top of that, the matplotlib plotting
packages are based around numpy, and
spyder is designed around both
"""

import matplotlib.pyplot as plt
plt.plot(xb,sinxb)
plt.xlabel('x')
plt.ylabel(r'$\sin(\omega * x)$')
```

The three musketeers: numpy, scipy, matplotlib

- Numpy provides fast (C/Fortran fast) numerical array operations : creating, reshaping, arithmetic, standard functions (sin, log,...)
- Numpy array corresponds directly to a C/Fortran array. Easy to call C/Fortran functions
- Scipy provides common scientific stuff : quadrature, optimization, linear algebra. Uses numpy
- Matplotlib provides graphics, uses numpy.

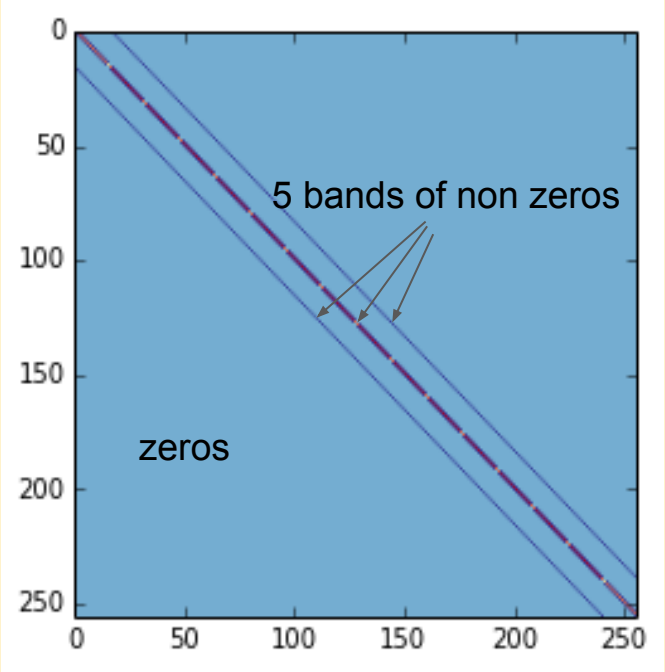


[Sparse] Linear Algebra

- Numerical solution of PDEs involves sparse linear systems
- (1) find u where $Lu = r$:

L is an $N*N$ sparse matrix, r is known

- You could construct an $N*N$ numpy array - a dense matrix with many zeros - and solve (1) using scipy.linalg.solve
- That would be idiotic: $O(N^2)$ storage, $O(N^3)$ time.
- scipy.sparse, scipy.sparse.linalg



Matrix structure ,2D, nx*nx diffusion equation.

Sparse storage : 5 nx^2 (10 MB for a 512*512 mesh)

Dense storage : nx^4 (512 GB for a 512*512 mesh)

Sparse Matrix Tools

```
#numpy is always on the table
import numpy as np

# sparse matrix formats
import scipy.sparse as sma

#sparse matrix linear alegbra.
import scipy.sparse.linalg as sla

#dense matrix linear algebra
import scipy.linalg  as la

#solve a sparse system, uses UMFPack
#iterative methods also available
Ls = sparsePoisson(n,n) # n*n * 5 arrays
u = sla.spsolve(Ls, r)

#solve a dense system, uses LAPACK
Ld = densePoisson(n,n) # n*n * n*n array
u = la.solve(Ld, r)
```