What is Scipy?

Tool suite: Numpy, Scipy, matplotlib, IPython

Damian R. Eads
eads@soe.ucsc.edu

Computer Science
University of California, Santa Cruz
What is Scipy?

- a project, a library, a team, a community
- a tool suite for Python, written in C
- **Numpy**: arrays (stable, linear algebra, basic algorithms)
- **Scipy**: scientific library (large suite of tool boxes)
- **matplotlib**: powerful plotting with MATLAB-like syntax
- **IPython**: interactive, tab completion, distributed, readline, **matplotlib** integration
Scipy Packages

- **cluster**: Clustering, Vector Quantization
- **fftpack**: Fast Fourier Transforms (FFT Pack)
- **linalg**: Linear Algebra (BLAS, LAPACK, ATLAS)
- **sparse**: Sparse Matrices and Linear Algebra (UMFPack)
- **stats**: Random Numbers, Distribution Manipulations, Density Estimation, Moment Calculation
Scipy Packages

- **integrate**: numerical integration (quadrature)
- **optimize**: optimization (LP, QP, QCP)
- **signal**: signal processing, basic filters, wavelets
- **ndimage**: image processing, edge detection, morphology, image statistics, connected components, convolution, etc.
- **weave**: C/C++ integration with multiline strings for prototype code you can’t vectorize
non-BSD licenses allowed (GPL, LGPL, etc.)
still deciding on structure, packaging, standards; web page needs work

**ann**: interface to the popular approximate nearest neighbor library. Very fast.

**audiolab**: processing audio waveforms. Lots of formats.

**learn**: machine learning

**delauney**: Voronoi tesselations and Delauney triangulations
GNU Scientific Library (GSL): free software, very stable, lots of features

Numerical Recipes: restrictive licensing

great for production systems

hard to collaborate: some mathy friends are C-phobic

syntax isn’t so succinct

efficient
Scientific Computing in C

- high-level manipulative code is difficult but
  low-level is stuff is efficient and intuitive!
- Clean Python and C integration: the best of both worlds, just
  - write low-level vectorizable algorithms in Python
  - write low-level non-vectorizable algorithms in C
  - write high-level manipulative code in Python
Octave and Scilab (MATLAB-like)


- hard to write extensions
- confined to small programs
- *good* for prototyping algorithms
- *difficult* for developing larger systems
- based on old languages (MATLAB, 1982) and (S, S+, 1975, 1988)
Why I don’t like MATLAB?

- proprietary: hard to collaborate—collaborators need licenses
- licenses: per-machine, not per-user. As project grows, more licenses needed - $$$.
- one needs lots of toolboxes, one for each machine - more $$$.
- pass-by-value: makes large data sets painful; slows programs
- *global variables*: hack introduced to get around pass-by-value. Makes code hard to maintain.
Why I don’t like MATLAB?

- encourages interactive workflow. Batch scripting is difficult: reproducability of experiments and plots?
- hard to code rich data structures: trees, graphs, heaps, lists
- not conducive to developing open source packages. Subcommunities are rare.
hard to organize MATLAB code into one coherent package. Must traverse lots of files for small modules. Code not organized into modules.

we need OO to organize large scale systems

*we do object-oriented:* yeah, sure you do.
- must create a directory for every class
- a file for every function.
- objects are immutable. Changing involves a copy.

one function/file: hard to organize code
Why I don’t like MATLAB?

- large scale cluster computing is limited: licenses, bloated minimal memory footprint, starting is slow, crashes require restarts taking time

- packages for coordinating large jobs: must buy another toolbox

- no cvs update: bug fixes, new tools are not immediate

- black box: can’t track down why MATLAB crashes.
Why I don’t like MATLAB?

- can’t contribute code back—no larger open source community. MATLAB is evolved on Mathworks’ terms, not the user/scientist.
- MATLAB is very numeric-specific. Python is more universal.
- GUI and database toolkits don’t come with it
- parsing files is difficult based on old fscanf technology
- hard to work with non-matrix data sets (e.g. web data, text)
Why I don’t like MATLAB?

- mex is clunky: a lot of infrastructure is needed for a single external function. Documentation is incomplete.
- hard to wrap existing C libraries (e.g. GDAL). Must write large collection of wrappers.
- hard to debug mex C code
- hard to install on new machines. Sysadmins are sometimes needed to communicate with MATLAB sales office.
Why not Octave/Scilab?

- They aren’t universal languages
- Hard to write large applications—languages not designed for it (e.g. pass-by-value, global variable hacks)
- Thin spread: must focus on both language and interpreter design and science code
- Python/Scipy: separation of concerns
  - python team: focus on developing the language and base tools.
  - Scipy team: focus on developing large science toolset.
Why not Octave/Scilab?

- Not as much as external code is available as with python.
- Wrapping C libraries is difficult. MEX interface not intuitive.
- Richer data structures (e.g. trees) not available.
Why Scipy?

- free: easy-to-collaborate
- open source: you’re part of community, no black box
- python
  - universal: lots of people know and trust it
  - flexible: easy to do simple tasks
  - object-oriented: designed for writing applications
- large corpora of packages: cross cuts many fields and problem domains
Why Scipy?

- fast: C implementation of core code
- succinct syntax: python’s operator overloading [ ], *, -, **, and yes lots of in-place operators **=.
- crazily flexible indexing—key to Numpy’s success.
- lots of ways to vectorize (more than MATLAB?)
- in-place algorithms are easy with pass-by-reference and in-place operators
Why Scipy?

- rich data structures: text, graphs, trees, hash maps
- powerful parsing: binary file unpacking, text parsing
- network I/O
- GUI building
- dot notation is intuitive (e.g. \((x < 0).\text{mean}()\), arrays are objects
easy to transition from MATLAB, minor differences. Arrays are
- Python objects, implemented in C for efficiency
- not copied when sliced, reshaped or transposed
- sliced with square brackets (instead of parentheses)
- indexable with lists of indices and boolean arrays
- reshapable to flat views with .ravel()
Large Applications

- large data repository? no problem: MySQL databases
  - store terabytes of astronomy data. Queries returned as numpy arrays.
  - geospatial images organized by georeference queries
- GUI Building: Qt, GTK, Tcl – take your pick!
- C and C++ extensions are easy!
- Existing C libraries getting wrapped all the time!
Why Scipy?

- Quick migration from MATLAB
  - differences are largely semantic: values are references to arrays
  - personal experience: 15,000+ lines converted to Python/Scipy in three months
  - lots of functions w/ same calling convention
- Seamless interactive and batch processing
- Easy to prototype – no need to rewrite prototypes
Why Scipy?

- Weave and Cython: write C/C++ in Python!
- multi-threading with pyprocessing
- parallelized interactivity with new IPython 1
- wrapping existing C libraries is easy:

```python
# in python
from ctypes import load_library
mylib = load_library("/usr/lib/mylib.so")
mylib.compute(myArray.ctypes.data)
```
Vibrant, friendly, and helpful.

Scipy serves science.

Scientists represent the bulk of the community.

Enthought is available for hire: let your science money shape Scipy’s future.
lots of software packages depend on Scipy base tools: distributed computing, bioinformatics, geospatial, brain imaging, aeronautics, financial, physics, commercial end-user appliances, etc.
Onward to demo! Let’s see Scipy in action by playing with it at an IPython prompt. We will use matplotlib for plotting.