Teaching Numerical and Symbolic Computation with Open Source Software

Randall J. LeVeque
Department of Applied Mathematics
University of Washington

With thanks to

Kyle Mandli, Kirsten Fagnan, Donna Calhoun,
William Stein, Josh Cantor,
NSF

Outline

Python

- Why switch to Python?
- Sample code for BVP
- Fortran interface
- Web tools

Sage

- Symbolic $+$ Numerical computing
- Sage notebooks
- Unified interface to other languages, packages

Why Python?

Provides the advantages of Matlab, e.g.

- interactive, simple sytax, dynamic typing
- similar user interface to LAPACK, FFTW, etc.
- matplotlib provides Matlab-style plotting in 1d, 2d

Why Python?

Provides the advantages of Matlab, e.g.

- interactive, simple sytax, dynamic typing
- similar user interface to LAPACK, FFTW, etc.
- matplotlib provides Matlab-style plotting in 1d, 2d

Open source

- Basic Python already available on most computers
- Enthought Python Distribution (enthought.com)
  makes it easy to install NumPy, SciPy, matplotlib
- Greater availablity for students
- Teach them a language they can continue to use beyond the academic environment
- More widely available to collaborators in industry, labs

Why Python? Powerful data structures

- N-dim arrays (arbitrary N),
- mixed-type data structures, e.g.

```
a = [37, 'astring', [5,6], linspace]
```

- Dictionaries, e.g.

```
errors = {('LW',0.1): 0.042, \
          ('LW',0.05): 0.0105}
errors[('upwind', 0.1)] = 0.35

h = []; e = []
for (key,val) in errors.iteritems():
    if key[0]=='LW':
        h.append(key[1]);   e.append(val)
loglog(h,e);   title('Lax-Wendroff errors')
```

Sample Python script for 2-point BVP

```
from numpy import *
from scipy import sparse, linsolve

def bvp(m):

    # True solution and second derivative:
    beta = 30.
    def solution(x):
        u = cos(beta*x); u2 = -beta**2*cos(beta*x))
        return (u,u2)

    ax = 0;    bx = 1.
    h = (bx-ax)/(m+1.)
    x = linspace(ax,bx,m+2)    # x[0]=ax
                               # x[m+1] = bx
```

Sample Python script for 2-point BVP

```
e = ones(m+2)
A = sparse.spdiags([e, -2*e, e], [-1, 0, 1], \
                   m+2, m+2) / h**2

A[0,0] = 1.;   A[0,1] = 0.;
A[m+1,m] = 0.;  A[m+1,m+1] = 1.

(utrue,f) = solution(x)     # sets RHS
f[0] = utrue[0]             # BC at ax
f[m+1] = utrue[m+1]         # BC at bx

u = linsolve.spsolve(A,f)

print "max error = ", norm(abs(u-utrue), 'inf')
```

Sample Python script for 2-point BVP

```
from pylab import *
clf()
plot(x,u,'bo')

hold(True)
xfine = linspace(ax,bx,1001)
(ufine,ffine) = solution(xfine)
plot(xfine,ufine,'r')

axis([ax,bx,-1.1,1.1])
legend(['computed','true'])
title('Solution with %s grid points' % str(m+2))
hold(False)
```

Interfacing with Fortran: f2py

Fortran program myf.f:

```
      double precision function myf(x)
      double precision x
      myf = 3.d0 * x
      return
      end
```

```
unix% f2py -c -m mymodule myf.f

unix% python
>>> import mymodule
>>> mymodule.myf(5)
15.0
```

Why Python? Web tools....

Can open files remotely or download from web, e.g.

```
import urllib
URLfile = 'http://www.clawpack.org/clawlogo.jpg'

try:
    urllib.urlretrieve(URLfile, 'mylogo.jpg')
except:
    print 'Sorry, could not download file'
```

Start a local web server...

```
import BaseHTTPServer, CGIHTTPServer, sys
ServerClass = BaseHTTPServer.HTTPServer
HandlerClass = CGIHTTPServer.CGIHTTPRequestHandler
HandlerClass.protocol_version = "HTTP/1.0"
server_address = ('127.0.0.1', 50005)

try:
    httpd = ServerClass(server_address, HandlerClass)
except:
    print '*** Error starting server'
    sys.exit(1)

try:
    httpd.serve_forever()
except KeyboardInterrupt:
    print "Server shutting down"
```

cgi-scripting

html file:

```
<form action="http://.../cgi-bin/mycgi.py.cgi"
      method="POST">
x = <input type="text" name="x" value="1.0">
<input type="submit" name=request value="Submit">
</form>
```

---

cgi-script mycgi.py.cgi:

```
import cgi
form = cgi.FieldStorage()
x = float(form.getvalue('x'))

print 'Content-type: text/html\n'
print 'x = ', x
```

Sage – www.sagemath.org

Why use Sage?

- Python based front end to both numerical and symbolic packages
- includes NumPy, SciPy, maxima, etc. plus many more
- Web-based notebooks
  Uses jsMath to allow typesetting
  Input boxes can contain Python, Sage,
  Fortran, C, Matlab, latex, etc.
- Can try out online without downloading.