COMPARISON OF MATLAB, OCTAVE, AND PYLAB

ED BUELER

Matlab (www.mathworks.com) was designed by Cleve Moler for teaching numerical linear algebra. It has become a powerful programming language and engineering tool. It should not be confused with the city in Bangladesh (en.wikipedia.org/wiki/Matlab_Upazila).

But I like free, open source software. There are free alternatives to Matlab, and they'll work well for this course. First, Octave is a Matlab clone. The examples below work in an identical way in Matlab and in Octave.[1] I will mostly use Octave myself for teaching, but I'll test examples in both Octave and Matlab. To download Octave, go to www.gnu.org/software/octave.

Second, the scipy (www.scipy.org) and pylab (matplotlib.sourceforge.net) libraries give the general-purpose interpreted language Python (python.org) all of Matlab functionality plus quite a bit more. This combination is called pylab. Using it with the ipython interactive shell (ipython.scipy.org) gives the most Matlab-like experience. However, the examples below hint at the computer language differences and the different modes of thought, between Matlab/Octave and Python. Students who already use Python, or have computer science inclinations, will like this option.

On the next page are two algorithms each in Matlab/Octave form (left column) and pylab form (right column). To download these examples, follow links at the class page

www.dms.uaf.edu/∼bueler/Math615S12.htm.

Here are some brief "how-to" comments for the Matlab/Octave examples: `expint.m` is a *script*. A script is run by starting Matlab/Octave, either in the directory containing the examples, or with a change to the "path". Then type the name of the script at the prompt, without the ".m":

```
>> expint
```

The second algorithm `bis.m` is a *function* which needs inputs. At the prompt enter

```
>> f = @(x) cos(x) - x
>> bis(0,1,f)
```

for example. Doing `help expint` or `help bis` shows the block of comments as documentation.

For the Python versions: Type `run expint.py` at the ipython prompt or `python expint.py` or `./expint.py`. Note that a script like `expint.py` is made executable by adding a "shebang" in the first line ("`#!/usr/bin/env python`") and adding executable permissions (do: `chmod a+x expint.py`). For the function `bis.py`, run Python or ipython and do: `from bis import bis`. In ipython you can then do `bis?` to get documentation for that function, and run the example as shown in the docstring.

Date: December 27, 2011.

[1] Incompatibilities between Octave and Matlab are a reportable Octave bug.

expint.m

```
% plot the integrand and approximate
% the integral
%   / 1
%   |    exp(-x^2/pi) dx
%   / 0
% by left-hand, right-hand, and
% trapezoid rules

N = 1000;
dx = (1 - 0) / N;
x = linspace(0,1,N+1);
y = exp(- x.^2 / pi);

plot(x,y)
axis([0 1 0 1]), grid

format long
lhand = dx * sum(y(1:end-1))
rhand = dx * sum(y(2:end))
trap  = (dx/2) * sum(y(1:end-1)+y(2:end))
exact = (pi/2) * erf(1/sqrt(pi))
```

expint.py

```
#!/usr/bin/env python

# plot the integrand and approximate
# the integral
#    / 1
#    |    exp(-x^2/pi) dx
#    / 0
# by left-hand, right-hand, and
# trapezoid rules

from pylab import plot,axis,linspace,sum, \
                  pi,sqrt,exp,show,grid
from scipy.special import erf

N = 1000
dx = (1.0 - 0.0) / N
x = linspace(0.0,1.0,N+1)
y = exp(- x**2 / pi)

plot(x,y)
axis([0.0,1.0,0.0,1.0]); grid(True)

lhand = dx * sum(y[:-1])
print "lhand = %.15f" % lhand
rhand = dx * sum(y[1:])
print "rhand = %.15f" % rhand
trap  = (dx/2) * sum(y[:-1]+y[1:])
print "trap = %.15f" % trap
exact = (pi/2) * erf(1/sqrt(pi))
print "exact = %.15f" % exact
show()  # allow user to close figure
```

bis.m

```
function c = bis(a,b,f)
% BIS  Apply the bisection method to solve
%   f(x) = 0
% with initial bracket [a,b].
% example:
%   >> f = @(x) cos(x) - x      % define fcn
%   >> r = bis(0,1,f)           % find root
%   >> f(r)                     % confirm

if (feval(f,a)) * (feval(f,b)) > 0
  error(’not a bracket!’), end
for k = 1:100
  c = (a+b)/2;
  r = feval(f,c);
  if abs(r) < 1e-12
    return  % we are done
  elseif feval(f,a) * r >= 0.0
    a = c;
  else
    b = c;
  end
end
error(’no convergence’)
```

bis.py

```
def bis(a,b,f):
  """ BIS  Apply the bisection method to solve
    f(x) = 0
  with initial bracket [a,b].
  example (after "from bis import bis"):
    def f(x): return cos(x) - x   # define fcn
    r = bis(0.0,1.0,f)            # find root
    print(r); print(f(r))         # confirm"""

  if f(a) * f(b) > 0.0:
    print "not a bracket!"; return
  for k in range(100):
    c = (a+b)/2
    r = f(c)
    if abs(r) < 1e-12:
      return c # we are done
    elif f(a) * r >= 0.0:
      a = c
    else:
      b = c
  print "no convergence"; return
```