COMPARISON OF MATLAB, OCTAVE, AND PYLAB

ED BUELER

MATLAB (www.mathworks.com) was designed by Cleve Moler around 1980 for teaching numerical linear algebra without needing FORTRAN. It has become a powerful programming language and engineering tool. More than half of Math 615 students are already familiar with it. It is available in most labs and graduate student offices at UAF.

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. 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 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
bueler.github.io/M615S14/.

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` at a shell prompt. 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: January 16, 2014.

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))
```

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')
```

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.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
```