

COMPARISON OF MATLAB, OCTAVE, AND PYTHON/SCIPY/PYLAB EXAMPLES, FOR MATH 694

ED BUELER

Trefethen & Bau's *Numerical Linear Algebra* uses MATLAB (<http://www.mathworks.com>) for very good reason. MATLAB was designed by Cleve Moler for teaching numerical linear algebra. MATLAB has, since its invention, become a powerful programming language and engineering tool. The best single tool, in my opinion.

There are open source alternatives to MATLAB, and they'll work for this course. I like free, open source software. Among other open source alternatives, GNU OCTAVE is intended to be a MATLAB clone, and the examples below work in an identical way in MATLAB and in OCTAVE. Incompatibilities between OCTAVE and MATLAB are now treated by the OCTAVE developers as a reportable bug. I will mostly use OCTAVE myself for teaching, but I'll make an effort to test most examples in both OCTAVE and MATLAB. To download OCTAVE, go to <http://www.gnu.org/software/octave/>.

The general purpose language PYTHON has developed in the direction of MATLAB functionality with the SCIPY (<http://www.scipy.org/>) and PYLAB (<http://matplotlib.sourceforge.net/>) projects. The IPYTHON interactive shell gives, with PYLAB, the most MATLAB-like experience. I like PYTHON. The examples below hint at the computer language differences, and the different modes of thought, between MATLAB/OCTAVE and PYTHON. I suspect that only students who already use PYTHON will find PYTHON/SCIPY/PYLAB effective for this course.

Some brief "how-to" comments might help compare examples below. The MATLAB/OCTAVE examples `legendre.m` and `hello.m` are *scripts*. These are run by starting MATLAB/OCTAVE, making sure that the "path" includes the directory containing the examples. Then type the name of the script at the prompt, without the ".m": `>> legendre` or `>> hello`. For the first two PYTHON examples type `run legendre.py` or `run hello.py` at the IPYTHON prompt or `python legendre.py`, `python hello.py` at an ordinary shell prompt.

The last example `mgs.m`, `mgs.py` is a *function* which needs an input, and I'll address that later.

<u><code>legendre.m</code></u> (MATLAB & OCTAVE version)	<u><code>legendre.py</code></u> (PYTHON version)
<pre>% Trefethen & Bau, page 64 x = (-128:128)'/128; A = [x.^0 x.^1 x.^2 x.^3]; [Q,R] = qr(A,0); scale = Q(257,:); Q = Q * diag(1 ./ scale); plot(Q)</pre>	<pre># Trefethen & Bau, page 64 from pylab import * x = linspace(-1.0,1.0,257).reshape((257,1)) A = concatenate((x**0, x**1, x**2, x**3),axis=1) [Q,R] = qr(A) scale = Q[256,:] Q = dot(Q,diag(1.0 / scale)) plot(Q), show()</pre>

hello.m

```
% assembles HELLO matrix
% see Trefethen&Bau lecture 9, exer

bl = ones(8,6);
H = bl;
H(1:3,3:4) = zeros(3,2);
H(6:8,3:4) = zeros(3,2);
E = bl;
E(3,3:6) = zeros(1,4);
E(6,3:6) = zeros(1,4);
L = bl;
L(1:6,3:6) = zeros(6,4);
O = bl;
O(3:6,3:4) = zeros(4,2);

HELLO = zeros(15,40);
HELLO(2:9,2:7) = H;
HELLO(3:10,10:15) = E;
HELLO(4:11,18:23) = L;
HELLO(5:12,26:31) = L;
HELLO(6:13,34:39) = O;

spy(HELLO)
```

mgs.m

```
function [Q,R] = mgs(A);
% MGS computes QR factors

twoeps = 2*eps;
[m n] = size(A);
R = zeros(n,n);
scal = max(max(abs(A)));
if scal == 0
    Q = eye(m,n);
    return, end
Q = A;
for i = 1:n
    r = norm(Q(:,i),2);
    R(i,i) = r;
    if abs(r)/scal < twoeps
        error('nearly zero col'), end
    w = Q(:,i)/r;
    Q(:,i) = w;
    for j = i+1:n
        r = w'*Q(:,j);
        R(i,j) = r;
        Q(:,j) = Q(:,j)-r*w;
    end
end
end
```

hello.py

```
# assembles HELLO matrix
# see Trefethen&Bau lecture 9, exer
from pylab import ones, zeros, spy, show

bl = ones((8,6))
H = bl.copy()
H[0:3,2:4] = zeros((3,2))
H[5:8,2:4] = zeros((3,2))
E = bl.copy()
E[2,2:6] = zeros((1,4))
E[5,2:6] = zeros((1,4))
L = bl.copy()
L[0:6,2:6] = zeros((6,4))
O = bl.copy()
O[2:6,2:4] = zeros((4,2))

HELLO = zeros((15,40))
HELLO[1:9,1:7] = H
HELLO[2:10,9:15] = E
HELLO[3:11,17:23] = L
HELLO[4:12,25:31] = L
HELLO[5:13,33:39] = O

spy(HELLO,marker='.'); show()
```

mgs.py

```
def mgs(A):
    """MGS computes QR factors"""
    from pylab import shape, zeros, eye, norm, dot
    import sys, numpy

    twoeps = 2.0 * numpy.finfo(numpy.double).eps
    (m, n) = shape(A)
    R = zeros((n,n))
    scal = abs(A).max()
    if scal == 0:
        Q = eye(m,n)
        return (Q,R)
    Q = A.copy()
    for i in range(n):
        r = norm(Q[:,i],2)
        R[i,i] = r
        if abs(r)/scal < twoeps:
            sys.exit('nearly zero col')
        w = Q[:,i] / r
        Q[:,i] = w
        for j in range(i+1,n):
            r = dot(w,Q[:,j])
            R[i,j] = r
            Q[:,j] = Q[:,j] - r * w
    return (Q,R)
```