Laboratory Exercise 1: Root searches

OBJECTIVES: To put into practice some root-finding algorithms and to study their behaviours.

Work through the exercises on this sheet. Try to complete as much as possible. If you finish early, work through the optional material at the end.

1 Fixed point iteration

We would like to find a value of x that satisfies $x - \cos(x) = 0$. We could try a fixed point iteration, starting from $x = 1$. At the terminal command window type python. You should see the >>> command prompt. At the prompt type

```
Import numpy as np
x=1
x=cos(x)
print x
```

Repeat the last command line several times (use the up-arrow key). Is the iteration converging? If so, to what value of x?

The same value of x should also satisfy $x - \text{acos}(x) = 0$ (where acos is the inverse of the cosine function). Reset x to 0.7 and iterate `x=acos(x)`. Is the iteration converging? If so, to what value of x?

Can you explain the behaviour in the two cases using the theory from lectures?

Use an editor to create the following procedure, which uses a fixed point iteration to calculate $\sqrt{2}$.

```
#
# This program finds roots
#
count = 0
err = 1
x = 1

while count < 20 and (err > .001):
    x = .5*(x+2./x)
    err = abs(x*x-2.)/2.
    count += 1

print 'After ',count,' iterations x=',x
```

Save it in `root.py`. Compile and run the procedure. Does it converge? To the correct answer?

Try modifying the procedure to solve

(i) $e^{-x} - x = 0$,

(ii) $x^x = 5$.

2 Bracketing methods

Use an editor to create the following function and procedure, which uses the bisection method to bracket a root of a cubic polynomial. Save it in `bisect.py`.

```
def poly(x):
 """ This function evaluates a polynomial in x """
    a = 3.
    b = 1.
    c = -5.
    d = -1.
   return d + x*(c + x*(b + x*a))

xl = -1.
fl = poly(xl)
xr = 1.
fr = poly(xr)
for iter in range(1,5):
    xnew = .5*(xl+xr)
    fnew = poly(xnew)
    if fl*fnew > 0:
        xl = xnew
        fl = fnew
    else:
        xr = xnew
        fr = fnew
print 'Bisect Root lies between ',xl,' and',xr
print 'Best guess is ',xnew,' Function value ',fnew
```

Compile and run the procedure. How accurate is the answer?

Modify the procedure so that it calculates `xnew` using the regula falsi method instead of the bisection method. Compile and run the procedure again. Is the answer more accurate?

By choosing different starting intervals, can you find the other two roots of the cubic?

3 Optional

1. Create a procedure that will use Newton's method to find roots of the same polynomial as used in section 2. You will need a new function that returns not just the value of the polynomial but also its derivative:

```
def dpoly(x):
    """ This function evaluates a polynomial and its derivative in x
        It returns a 2D array"""
    a = 3.
    b = 1.
    c = -5.
    d = -1.
   return [d + x*(c + x*(b + x*a)),c + x*(2.*b+ x*3*a)]
```

2. For the bisection, regula falsi, and Newton methods, try saving the successive estimates for $f(x)$ in an array: e.g. `guess(iter)=fnew`. After the iteration loop plot the values of `guess`, versus iteration number. Does this help to compare the rates of convergence for the three methods?