Managing computations through scripting
Newton's Method for solving non-linear equations

Arely Miramontes[1], Berenise Miramontes[2]

[1]Computational Mathematics, University of New Mexico
arelym@unm.edu
[2]Computational Mathematics, University of New Mexico
beremts12@unm.edu

Abstract – In this report we find roots of non-linear equations (f(x)=0) for different functions f(x)...

Keywords – Newton's Method, Modified Newton's Method, Non-linear equations, linear convergence, quadrature convergence.

I. INTRODUCTION

Using Newton's method to find the root of a function is one of many methods to do so. In the Newton's Method we start by stating an initial guess, x_0. Next, we draw the tangent line of the function y = f(x). Using this tangent line we intersect this tangent line with the x-axis and call the point as x_1. x_1 is the next approximation of the root. We iterate this method until we converge to the root of the function (f(x) = 0).

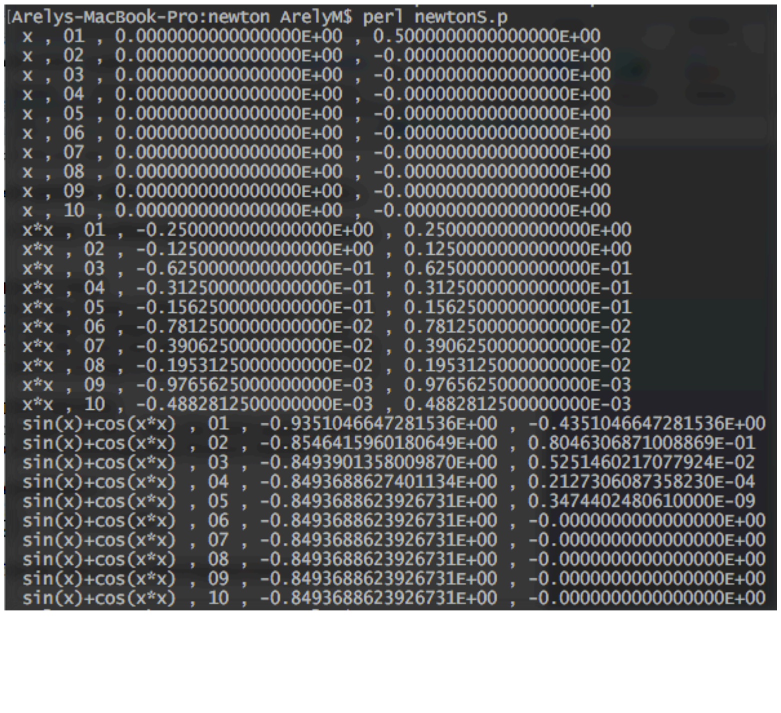
II. NEWTON'S METHOD

Solving for x gives an approximation for the root, which we call x_1. Next, the entire method will be iterative, so starting with x_1 will yield x_2, and so on. The following states Newton's Method:

$$x_0 = initial\ guess$$
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \qquad for\ i = 1,2,3,\ldots$$

In this report we will use Newton's Method to find the roots of the following functions: ***y = x***, ***y = x²*** , and ***y = sin(x) + cos(x²)***.

First, running Dr. Appelo's program *newtonS.p,* we inspect the outpout:



```
[Arelys-MacBook-Pro:newton ArelyM$ perl newtonS.p
 x , 01 , 0.0000000000000000E+00 , 0.5000000000000000E+00
 x , 02 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x , 03 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x , 04 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x , 05 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x , 06 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x , 07 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x , 08 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x , 09 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x , 10 , 0.0000000000000000E+00 , -0.0000000000000000E+00
 x*x , 01 , -0.2500000000000000E+00 , 0.2500000000000000E+00
 x*x , 02 , -0.1250000000000000E+00 , 0.1250000000000000E+00
 x*x , 03 , -0.6250000000000000E-01 , 0.6250000000000000E-01
 x*x , 04 , -0.3125000000000000E-01 , 0.3125000000000000E-01
 x*x , 05 , -0.1562500000000000E-01 , 0.1562500000000000E-01
 x*x , 06 , -0.7812500000000000E-02 , 0.7812500000000000E-02
 x*x , 07 , -0.3906250000000000E-02 , 0.3906250000000000E-02
 x*x , 08 , -0.1953125000000000E-02 , 0.1953125000000000E-02
 x*x , 09 , -0.9765625000000000E-03 , 0.9765625000000000E-03
 x*x , 10 , -0.4882812500000000E-03 , 0.4882812500000000E-03
 sin(x)+cos(x*x) , 01 , -0.9351046647281536E+00 , -0.4351046647281536E+00
 sin(x)+cos(x*x) , 02 , -0.8546415960180649E+00 , 0.8046306871008869E-01
 sin(x)+cos(x*x) , 03 , -0.8493901358009870E+00 , 0.5251460217077924E-02
 sin(x)+cos(x*x) , 04 , -0.8493688627401134E+00 , 0.2127306087358230E-04
 sin(x)+cos(x*x) , 05 , -0.8493688623926731E+00 , 0.3474402480610000E-09
 sin(x)+cos(x*x) , 06 , -0.8493688623926731E+00 , -0.0000000000000000E+00
 sin(x)+cos(x*x) , 07 , -0.8493688623926731E+00 , -0.0000000000000000E+00
 sin(x)+cos(x*x) , 08 , -0.8493688623926731E+00 , -0.0000000000000000E+00
 sin(x)+cos(x*x) , 09 , -0.8493688623926731E+00 , -0.0000000000000000E+00
 sin(x)+cos(x*x) , 10 , -0.8493688623926731E+00 , -0.0000000000000000E+00
```

As suspected from the Homework 2 notes, this version of Newton's Method iterates only 10 times, which is too much or too little for some functions. We have modified the program so that it only iterates until the quantity $(E_{abs})_{n+1} = |x_{n+1} - x_n|$ is reached – this means when the approximate absolute error is less than 10^{-15}.

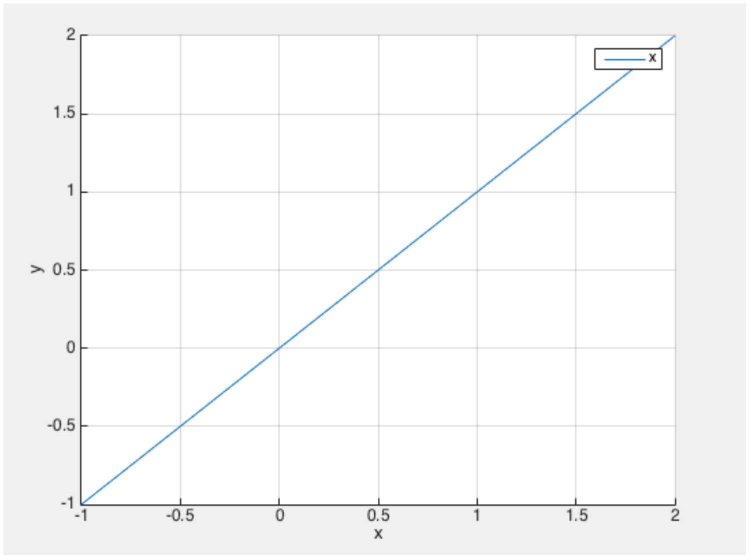
In MATLAB, we have achieved this, satisfying the condition of the error being less than 10^{-15}.

First, we created a function in Matlab called *newton.m* -- refer to the following:

```matlab
function [Xs,iter]=newton(f,fp,x0,Err,imax)

for i=1 : imax
    x(i) = x0 - feval(f,x0)/feval(fp,x0);
    approxErr(i) = abs((x(i) - x0)/x0)*100;
    if abs(approxErr(i) < Err)
        display(i);
        Xs = x(i);
        break
    end
    x0 = x(i);
    iter = [1:1:i];
end



if(i == imax)
    fprint('Solution was not obtained in %i iterations.\n',imax)
    Xs = ('No solution');
end
```

a) Now that we wrote the method in MATLAB, we now must state the function we want to find the root of, first we will find the root of $f(x) = x$.

To first get an idea of what the root will be we first plot the function, and we get:

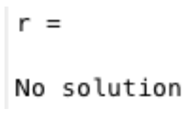


From what we see (graphically) the function crosses the x-axis somewhere between -0.5 and 0.5, so let's just take x_0 = 0.5. Then we can activate the method.

We now are ready to write a script to find the root of the function, $f(x) = x$, with the error being less than $10^{\wedge} - 15$, and the maximum number of iterations to be 1000:

```
f = @(x) x;
fp = @(x) 1;
% To first have an idea of what the roots of the this polynomial
will be we will plot the function.
hold all
x = [-1.0:0.01:2.0]';
y = f(x);
plot(x,y)
grid on
legend('x')
xlabel('x');
ylabel('y');
x0 = 0.5;
Err = 1.e-15;
imax = 1000;
[r,iter] = newton(f,fp,x0,Err,imax)
hold all
plot(r,0,'r*')
```

But when running the program, we get an error, because there is no root for the function, $f(x) = x$.

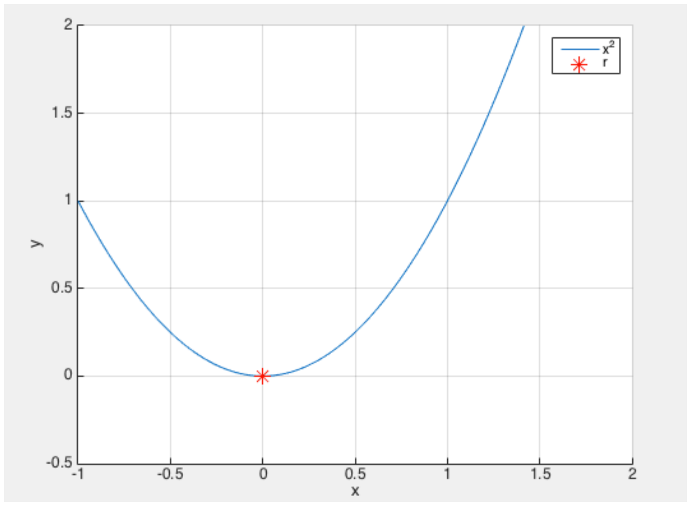


```
r =

No solution
```

Thus, there is no root for the function.

b) Next, we will find the root of the function, $f(x) = x^{\wedge}2$.

To first get an idea of what the root will be we first plot the function, and we get:

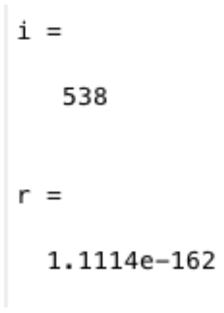


From what we see (graphically) the function crosses the x-axis somewhere between -0.5 and 0.5, so let's just take x_0 = 0.5. Then we can activate the method.

We now are ready to write a script to find the root of the function, $f(x) = x^2$, with the error being less than $10\char`^ - 15$, and the maximum number of iterations to be 1000:

```
f = @(x) x.^2;
fp = @(x) 2*x;

% To first have an idea of what the roots of the this polynomial
% will be we will plot the function.
hold all
x = [-1.0:0.01:2.0]';
y = f(x);
plot(x,y)
grid on

xlabel('x');
ylabel('y');
axis([-1 2 -0.5 2])



x0 = 0.5;
Err = 1.e-15;
imax = 1000;

[r,iter] = newton(f,fp,x0,Err,imax)

hold all
plot(r,0,'r*', 'MarkerSize',12)
legend('x^2','r')
```

When running the program, we get the root to be:



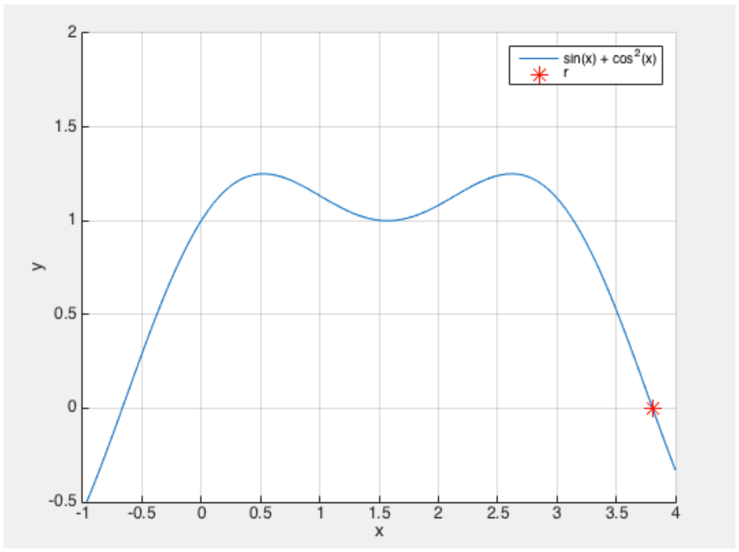
```
i =

   538


r =

  1.1114e-162
```

This method converged after 538 iterations.

Thus the root of this function is 1.114e^{-162}, which is essentially 0.

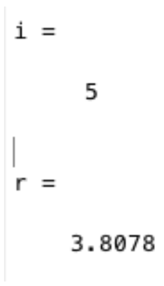
c) Finally, we will find the root of the function, $f(x) = \sin(x) + \cos^2(x)$.

To first get an idea of what the root will be we first plot the function, and we get:



From what we see (graphically) the function crosses the x-axis somewhere between 3.5 and 4, so let's just take $x_0 = 4$. Then we can activate the method.

We now are ready to write a script to find the root of the function, $f(x) = \sin(x) + \cos^2(x)$, with the error being less than $10^\wedge - 15$, and the maximum number of iterations to be 1000:



```
i =

     5

r =

    3.8078
```

This method converged after 5 iterations.

Thus the root of this function is 3.8078.

III. LINEAR AND QUADRATURE CONVERGENCE

Let us consider the statements of the theorem of linear and quadrature convergence.

1) Let f be twice continuously differentiable and $f(r) = 0$. If $f'(r) \neq 0$, then Newton's Method is locally and quadratically convergent to r.

   The error at step *i,* satisfies the following:

$$\log_{i\to\infty} \frac{e_{i+1}}{e_i^2} = M$$

$$And, \qquad M = \left|\frac{f''(r)}{2f'(r)}\right|$$

2) If (m+1) times continuously differentiable function on [a,b] has multiplicity m at root r, then Newton's Method is locally convergent to r, and the error at step *i,* satisfies the following:

$$\log_{i\to\infty}\frac{e_{i+1}}{e_i} = S$$
$$And, \qquad S = \frac{m-1}{m}$$

Now, using the given statements to determine local or quadratic convergence, we consider the three equations we found roots.

a) Consider the equation:

$$f(x) = x^2$$

Since we know there is a root at $r = 0$, let's do Newton's method:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$
$$= x_i - \frac{x_i^2}{2x_i}$$
$$= \frac{x_i}{2}$$

Now, let's take $x_i = 1$:

i	**x$_i$**	**e$_i$=|x$_i$-r|**	**e$_i$/e$_{i-1}$**
0	1.000	1.000	
1	0.500	0.500	0.500
2	0.250	0.250	0.500
3	0.125	0.125	0.500
⋮	⋮	⋮	⋮

Newton's Method converges to the root $r = 0$. The error formula is $\frac{e_i}{2}$, so the convergence is linear with convergence proportionality constant S = ½.

b) Consider the equation:

$$f(x) = \sin(x) + \cos^2(x)$$

$$Differentiate\ f(x) with\ respect\ to\ x:$$
$$f'(x) = \cos(x) - \sin(2x)$$

Since we know there is a root at $r =\ 3.8078$, let's do Newton's method:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$
$$= x_i - \frac{\sin(x_i) + \cos^2(x_i)}{\cos(x_i) - \sin(2x_i)}$$

To find the divergence we evaluate the derivatives at the root 3.8078.

$$f'(3.8078) = -1.75789 \ldots < 0$$
$$f''(3.8078) = 0.145748 \ldots > 0$$

Newton is quadratically convergent, since $f'(r) = -1.757 \neq 0$, and the convergence rate is
$$e_{i+1} = Me_i^2$$

where M = 0.145748/(2*-1.75789) = - 0.041455.

QUESTION: Is it possible to come up with a modified Newton's Method to regain the quadratic convergence for the problematic function?

Yes, in fact if f(x) is (m+1)-times continuously differentiable (such as our function above, f(x) = sin(x) + $\cos^2$(x)), on [a,b] and also contains a root of multiplicity m>1, then Newton's Modified method will be:

$$x_{i+1} = x_i - \frac{mf(x_i)}{f'(x_i)}$$

This method will converge locally and quadratically to the root r.

IV. CONCLUSIONS

Newton's method is a good method to determine the roots of functions; of course it can also fail in other ways. If f(x_i) = 0 at any iterations step, the method cannot continue. The method can diverge to infinity. If you use the local and quadratic convergence to determine how much iterations you need, you can guarantee a neighborhood of initial guesses that surround the root for which the convergence to that root is certain.

V. REFERENCES

[1]: Appelo, Daniel. "Homework 3." *Math 471*. UNM, n.d. Web. <http://math.unm.edu/~appelo/teaching/Math471F15/html/Homework2.html>.