

M439 081 Numerical Analysis Lab Week Three

The Newton-Raphson Method for Finding Roots

Save the following two files to your N: drive in your m439 folder: [newton.m](#) [newton2.m](#) and open MATLAB, making N:\m439 folder your current directory.

The Algorithm for Newton-Raphson Method

1. The MATLAB code for implementing the Newton algorithm is in the file **Newton.m**. To run this algorithm we must have a file defining the function (f) for which we want to find a zero, as well as a file containing defining its derivative (df)

a) Create a file saved as **f.m** (under the directory **N:\439**) using MATLAB's editor (Choose File - New - M-file) which contains the following code:

```
function y = f(x)
y = x.^3 - x - 3;
```

b) Create a second file saved as **df.m** (under the directory **N:\439**) using MATLAB's editor (Choose File - New - M-file) which contains the following code:

```
function y1 = df(x)
y1 = 3*x.^2 - 1;
```

c) Plot the function f(x) by executing the following commands:

```
>> fplot(@f,[-2 2])
>> grid on;
```

2. Looking at the graph, approximately where is the zero?

3. Set the format to long so we display more decimal digits.

```
>> format long
```

Then execute a call to Newton by entering the MATLAB commands:

```
>> p0 = 2, delta = 1e-12, epsilon = 1e-12, max1 = 50
>> [p,err,k,y] = newton(@f,@df,p0,delta,epsilon,max1)
```

For this call, what approximation is returned for the zero? _____

How many iterations are performed? _____

What is the approximate value of the function at this point? _____

What is the error estimate? _____

4. Newton's method is **locally convergent** -- it may or may not converge to a zero, depending upon the choice of the initial guess. That is if the initial approximation p_0 is too far away from the desired root, the sequence may not converge or may converge to another root if there is more than one root. Explain why another problem can result if the derivative is equal (or approximately equal) to 0 at the initial point (or subsequent points)

5. Execute a second call to newton by entering the MATLAB command:

```
>> p0 = 0; delta = 1e-12; epsilon = 1e-12; max1 = 50;  
>> [p,err,k,y] = newton(@f,@df,p0,delta,epsilon,max1)
```

What happens in this call and why?

You can see this more clearly by executing newton2 – which shows the successive iterations

```
>> [p,err,k,y] = newton2(@f,@df,p0,delta,epsilon,max1)
```

6. Experiment with 3 other initial choices p_0 for iteration with the Newton-Raphson method. Do this by executing the following with different choices of p_0 . (Replace ??? below with your choice - try some values between the first guess above and the second guess above -- For example try $p_0 = 0.1$)

```
delta = 1e-12  
epsilon = 1e-12  
max1 = 100  
p0 = ???  
[p,err,k,y] = newton(@f,@df,p0,delta,epsilon,max1)
```

List values you chose for p_0 and whether you had convergence or divergence:

First choice: _____ convergence or divergence.

Second choice: _____ convergence or divergence.

Third choice: _____ convergence or divergence.

Order of Convergence

7. **Definition Order of a Root** Assume that $f(x)$ and its derivatives

$f'(x), f''(x), \dots, f^{(m)}(x)$ are defined and continuous on an interval about $x = p$. We say that $f(x) = 0$ has a root of order m at $x = p$ if and only if $f(p) = f'(p) = \dots = f^{(m-1)}(p) = 0$, but $f^{(m)}(p) \neq 0$

A root of order $m = 1$ is called a **simple root**, and if $m > 1$ it is called a **multiple root**. A root of order $m = 2$ is called a **double root**.

Definition Order of Convergence: Assume that p_n converges to p . If two positive constants L and R exist such that

$\lim_{n \rightarrow \infty} \frac{|p - p_{n+1}|}{|p - p_n|^R} = L$, where L is finite and positive, then the order of convergence of the sequence $\{p_n\}$ is R .

To investigate the order of convergence of an iterative sequence, we must compare the ratio of successive errors to find if possible an $R > 0$ such that

$\frac{|p - p_{n+1}|}{|p - p_n|^R}$ is approximately constant (close to a value L) for larger enough n .

For the Newton-Raphson method for **simple roots**, convergence is **quadratic** ($R = 2$). For **multiple roots**, the convergence is slower -- it is **linear** ($R = 1$).

Why should we get **quadratic convergence** for the function we have been investigating above:

8. Verify this by computing the errors $E_n = |p - p_{n+1}|$ and the ratios $|E_{n+1}| / |E_n|^2$, as follows: **newton2.m** contains some modifications to **newton** -- The extra parameter P is a matrix that stores the iterative sequence of approximations for the zero. Execute a call to **newton2** with the following:

```
>> p0 = 2
>> [P,p,err,k,y] = newton2(@f,@df,p0,delta,epsilon,max1)
```

We assume that the last value in P stores our approximation to the root -- so we store our successive errors in E with the commands: (Note that **length(P)** returns how many elements are in P -- 6 in this case.)

```
>> length(P)
>> E = P(6)-P
```

9. Next compute the ratios as follows:

```
>> abs(E(2)/E(1)^2)    >> abs(E(3)/E(2)^2)
>> abs(E(4)/E(3)^2)    >> abs(E(5)/E(4)^2)
```

The ratios appear to be approaching what approximate constant? _____

Thus this indicates what kind of convergence? _____

Programming Assignment:

1. Newton's Square Root Formula: Perform the following iteration to compute an approximation to the square root of a nonnegative number A , with a starting guess p_0

$$p_n = \frac{p_{n-1} + \frac{A}{p_{n-1}}}{2}, n = 1, 2, \dots$$

Verify this formula below:

We are finding a zero of the function

$$f(x) = \underline{\hspace{2cm}}$$

Substituting into the Newton-Raphson iterative formula, the appropriate values for $f(x)$ and $f'(x)$ gives:

2. Modify `newton.m` and save your modified program as `squareroot.m` to implement Newton's Square Root Algorithm using the formula above. To develop the correct program consider:

- What should the interface (Inputs and Output arguments) be for **squareroot.m**?
- You must replace the line that contains Newton's iteration formula (`p1 = ...`) with the equivalent of the formula that you verified in step 1 above.
- With what should you replace the line `y = feval(f,p0)`?
- Document your program appropriately with comments (lines preceded with `%`)

Due Friday Sept 12th by 11:00 a.m. e-mail to me the m-file `squareroot.m` as an attachment. **In the message of the email provide the answers to the following questions which you should compute with your program.**

1. Approximate the square root of 8 with a starting value of 3.
2. Approximate the negative square root of 8 with a starting value of -3.
3. Approximate the square root of 91 with a starting value of 10.