Worksheet 1 Report

Mike Knee

Issue: 1

Date: October 14, 2016

School of Physics and Astronomy
University of Birmingham
Birmingham, B15 2TT

Contents

1 Question 1

For the first excercise a number of commands were input to the linux bash prompt, in order to understand what the different commands do and how to use them. The commands are listed here in order, with the output following and finally a description of what the command has done.

a) `mkdir compphys`

   No output, but a new directory has been created called "compphys"

b) cd compphys

   No output again, but now the current working directory is " /compphys"

c) cat > file1.txt [rtn] this is my first file [rtn][ctrl-c]

   No output is printed to the screen, however a new file called "file1.txt" has been created, containing the text "this is my first file"

d) `ls`

   Ouput is:

```
file1.txt
```

   The ls command lists the contents of the current working directory.

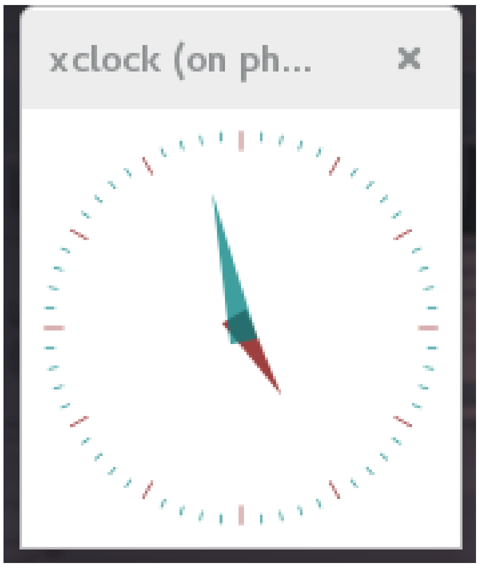


Figure 1: xclock open through the ssh session

e) more file1.txt

   Output is:

   ```
   this is my first file
   ```

   The more command pages files to the standard output, seen as the file “file1.txt” only has one line, that line is simply printed to the terminal.

f) `xclock&`

   Output is shown in figure 1.

   The `xclock&` command starts an xclock process. This will be opened on the client side through ssh if X11 forwarding is enabled, and the client is able to display xwindow objects. The ampersand is to tell the process to start the process in the background, ie. to allow the shell session to continue while xclock is still running.

g) `whoami`

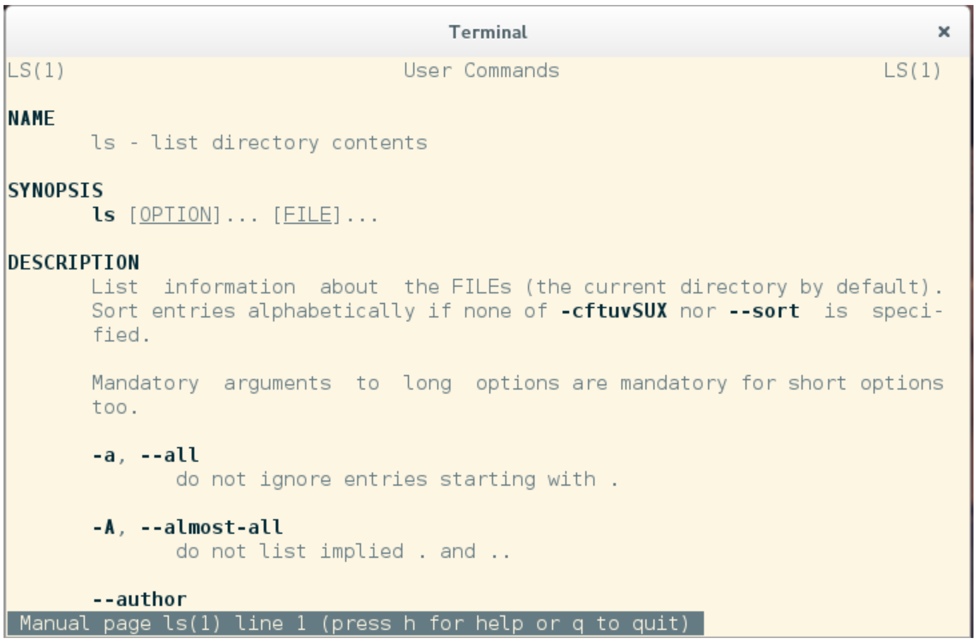
   Output is:

   ```
   mfk364
   ```

   This command prints the username of the current user.

h) `man ls`

   Output is a man page, a text document describing the usage of the `ls` command. Calling `man <command>` will display a man page on any command with proper documentation. Figure 2 shows the top of the ls man page.



```
LS(1)                        User Commands                        LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
 Manual page ls(1) line 1 (press h for help or q to quit)
```

Figure 2: Top of the man page for the `ls` command

i) `top`

Output is a display of running processes, ordered by CPU usage. The column processes are sorted by, and other options can be changed using commands while top is running. Figure 3 shows top while running, with the columns sorted by CPU usage.

j) `kill`

The `kill` command is used to stop running processes. In order to use kill one needs the PID of the process to be stopped. For this the `ps` command is used, which lists all the processes running under the current user's UID. Once a PID is known `kill [PID]` will send a terminate signal to the process.

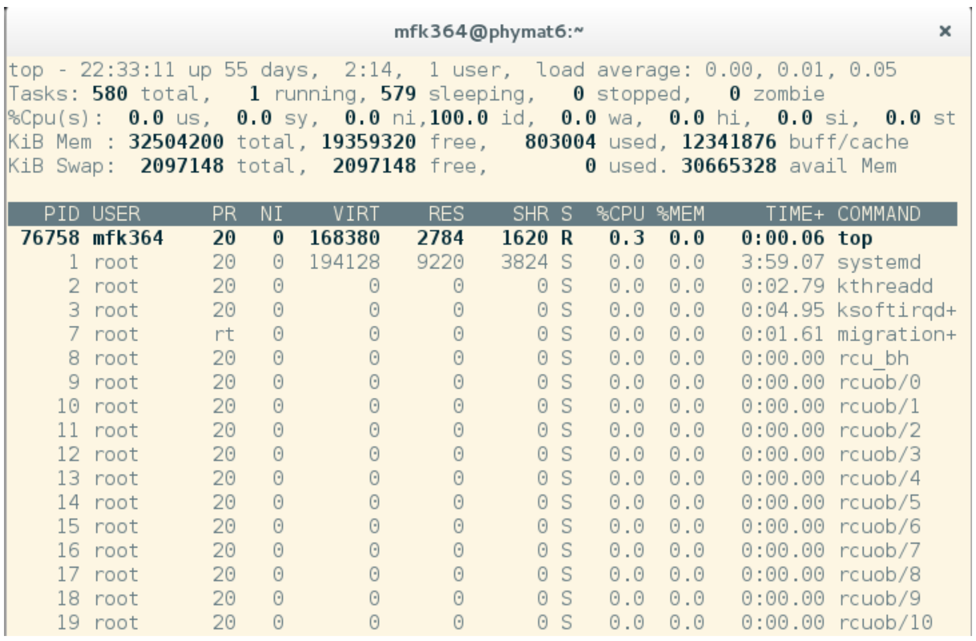
The output of the kill command is:

```
[running processes] Terminated\tab [process name]
```

k) `ps -u [username]`

As described above the `ps` command displays currently running processes. The -u option denotes that all the processes belonging to a user specified by [username] should be displayed. The default behaviour of texttps is to display the processes belonging to the current user running in the current TTY.

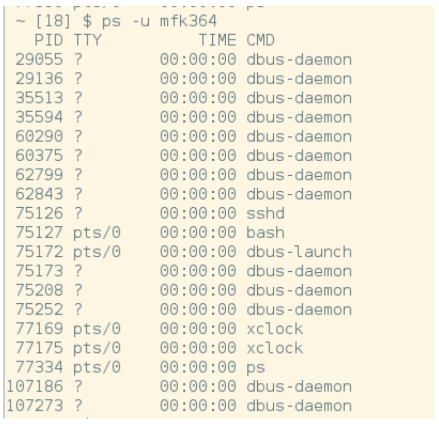
An example output of the `ps -u [username]` is shown in figure 4.



```
top - 22:33:11 up 55 days,  2:14,  1 user,  load average: 0.00, 0.01, 0.05
Tasks: 580 total,   1 running, 579 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32504200 total, 19359320 free,   803004 used, 12341876 buff/cache
KiB Swap:  2097148 total,  2097148 free,        0 used. 30665328 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 76758 mfk364    20   0  168380   2784   1620 R   0.3  0.0   0:00.06 top
     1 root      20   0  194128   9220   3824 S   0.0  0.0   3:59.07 systemd
     2 root      20   0       0      0      0 S   0.0  0.0   0:02.79 kthreadd
     3 root      20   0       0      0      0 S   0.0  0.0   0:04.95 ksoftirqd+
     7 root      rt   0       0      0      0 S   0.0  0.0   0:01.61 migration+
     8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
     9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
    10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/1
    11 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/2
    12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/3
    13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/4
    14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/5
    15 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/6
    16 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/7
    17 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/8
    18 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/9
    19 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/10
```

Figure 3: The `top` command in action



```
 ~ [18] $ ps -u mfk364
   PID TTY          TIME CMD
 29055 ?        00:00:00 dbus-daemon
 29136 ?        00:00:00 dbus-daemon
 35513 ?        00:00:00 dbus-daemon
 35594 ?        00:00:00 dbus-daemon
 60290 ?        00:00:00 dbus-daemon
 60375 ?        00:00:00 dbus-daemon
 62799 ?        00:00:00 dbus-daemon
 62843 ?        00:00:00 dbus-daemon
 75126 ?        00:00:00 sshd
 75127 pts/0    00:00:00 bash
 75172 pts/0    00:00:00 dbus-launch
 75173 ?        00:00:00 dbus-daemon
 75208 ?        00:00:00 dbus-daemon
 75252 ?        00:00:00 dbus-daemon
 77169 pts/0    00:00:00 xclock
 77175 pts/0    00:00:00 xclock
 77334 pts/0    00:00:00 ps
107186 ?        00:00:00 dbus-daemon
107273 ?        00:00:00 dbus-daemon
```

Figure 4: An example of `ps -u [username]` output

2 Question 4

For this question a C++ program was required to calculate different powers of ϕ (the silver ratio), given by $\phi = \frac{-1+\sqrt{5}}{2}$, and output the data to a file. The source code for this program is called "w1q4.cpp", and when run will output data to a file called "output". The code calculates and writes the power of phi by basic multiplication in lines 53-56.

The recursion relation

$$\phi^{n+1} = \phi^{n-1} - \phi^{n}$$

can be shown by noting that if we multiply both sides by ϕ we get

$$\phi^{n+2} = \phi^{n} - \phi^{n+1}$$

we can then substitute $m = n + 1$ to give us

$$\phi^{m+1} = \phi^{m-1} - \phi^{m}$$

This means the recursion relation is always valid for any value of $n \in \mathbb{Z}, n \geq n_0$ as long as the base case n_0 is defined for n (as we can see m must also be an integer greater than n_0).

The base case is relatively easy to show using the properties of the golden ratio, and by extension its conjugate which we are interested in. It is easiest to show by taking the case that $n_0 = 0$, especially when we note that a property of the golden ratio (see reference [2]) (Φ) is

$$\frac{1}{\Phi} = \Phi - 1$$

and therefore, because the silver ratio $\phi = \frac{1}{\Phi}$, that the silver ratio has a similar property

$$\frac{1}{\phi} = \phi + 1$$

If we now take our recursion relation with $n = 0$

$$\phi^{1} = \phi^{-1} - \phi^{0}$$

$$\phi = \frac{1}{\phi} - 1$$

we can see that the recursion relation is satisfied by considering the properties of the silver ratio as the conjugate of the golden ratio.

We can see by looking at the results that the two recursive functions both fail with varying degrees of severity. The function using floats begins to fluctuate relatively quickly, and produces very poor results. The double precision recursive function performs better, however it still produces fluctuations and innacuracies compared to our reference value calculated using direct multiplication. This happens in part because the error at each stage is compounded, so the relatively small error that occurs by using floating point numbers is amplified when larger values of N are used.

There is another number that satisfies this recursion relation, which corresponds to the negative golden ratio, $-\Phi = -1.61803$. This can be shown by substituting $-1/\phi$ into the recursion relation's base case where $n = 0$

$$-frac1\phi^1 = -frac1\phi^-1 - frac1phi^0$$

$$-frac1\phi = -\phi - 1$$

which we can see is equivalent to the recursion relation we already showed was valid. The fact that another number also satisfies the recursion relation means that fluctuations in our algorithm are not necessarily always flattened out by convergence, and can cause minor fluctuations and innacuracies to turn into vast errors.

3 Question 6

In this problem we consider a mass suspended by two springs from a bench, as shown in figure 5. We need to find the angle θ that the two spings make with the bench in terms of the known quantities: the length of the bench L, the mass m, and the spring constant k. The springs both have natural length $L/2$.

The equation for *theta* can be found by balancing forces. If we start with the equation

$$mg = 2T\sin\theta$$

where T is the tension in one spring, we can use the equation

$$T = k\Delta x$$

We then find Δx in terms of the L and θ

$$\Delta x = \frac{L}{2\cos\theta} - \frac{L}{2}$$

$$T = k\frac{L}{2}(\frac{1}{\cos\theta} - 1)$$

If we now substitute back into our first equation then we get

$$mg = kL(\tan\theta - \sin\theta)$$

We will use values $m = 5.5$kg, $L = 0.6$m and $k = 850$N. The code uses a recursive function to calculate and output the result using the bisection method of root finding (see chapter 03.03 in reference [1]). A precision of 6 significant figures is used, with our initial "bracket" around the root being 0 and $\pi/2$. By considering the physical properties of the system it is clear that these are sensible choices for out initial lower and upper bounds as θ must lie between these two points. The error at each iteration is given by half the difference between the upper and lower bounds, as we know the root must lie somewhere between this boundary and therefore cannot be more than half the difference from the centre of the boundary. Once the difference between the upper and lower bounds is less than our required precision we take the average of the upper and lower bound, and this gives us our answer to our chosen significant figures.

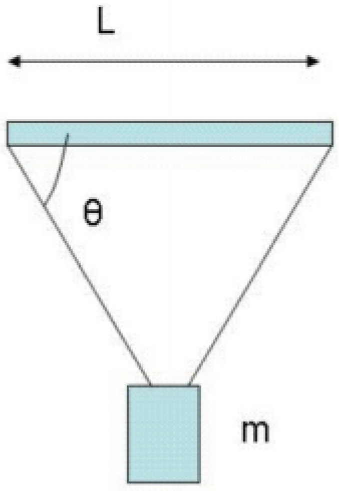


Figure 5: Mass suspended from a bench by two springs.

4 Question 7

Here we are considering the error propogation when using the bisection and Newton-Raphson methods for root finding. We can analyse the error when using the bisection method by considering that if at step n we have upper and lower bound u_n and l_n, middle value $m_n = \frac{u_n+l_n}{2}$ and error $\epsilon_n = \frac{u_n-l_n}{2}$. We can then say that our new boundaries will be one of u_n or l_n along with m_n, which has error $epsilon_n$ associated with it. If we assume that the error in the value of u_n/l_n is negligible compared to $epsilon_n$ then we can see that the middle on the $n+1$ iteration is

$$m_{n+1} = \frac{|m_n + \epsilon_n - (u_n\, or\, l_n)|}{2}$$

This shows us simply that the error is halved on each iteration, giving us:

$$\epsilon_{n+1} = \frac{\epsilon_n}{2}$$

Next let us consider the error given at each iteration by the Newton-Raphson method. If we perform a taylor expansion of the function about the root x_{root} with the value $x = x_n$ we get

$$f(x_n) = f(x_{root} + f'(x_{root})(x_n - x_{root}) + \frac{1}{2}f''(x_{root})(x_n - x_{root})^2 + ...$$

If we then note that the absolute error at each iteration is given by the difference between the value at each iteration and the actual root

$$\epsilon_n = x_n - x_{root}$$

and substitute this into our taylor expansion above (and also noting that $f(x_{root}) = 0$) we get

$$f(x_n) = f'(x_{root})\epsilon_n + \frac{1}{2}f''(x_{root})\epsilon^2$$

We can define the error at the $n+1$ step as the error at the n step plus the difference in the values at each step, i.e.

$$\epsilon_{n+1} = \epsilon_n + x_{n+1} - x_n$$

From the Newton-Raphson algorithm we can see that

$$x_{n+1} - x_n = fracf(x_n)f'(x_n)$$

and if we substitue this into our error relationship above we get

$$\epsilon_{n+1} = \epsilon_n + \frac{f(x_n)}{f'(x_n)}$$

We can then substitute in the taylor expansion above, and its derivative which is given by

$$f'(x_n) = f'(x_{root}) + f''(x_{root})\epsilon_n + ...$$

as $epsilon_n$ is linearly related to x_n. This then gives us

$$\epsilon_{n+1} = \epsilon_n + \frac{f'(x_{root})\epsilon_n + \frac{1}{2}f''(x_{root})\epsilon_n^2 + ...}{f'(x_{root}) + f''(x_{root})\epsilon_n + ...} \approx \epsilon_n + \frac{f'(x_{root})\epsilon_n + \frac{1}{2}f''(x_{root})\epsilon_n^2}{f'(x_{root}) + f''(x_{root})\epsilon_n}$$

Next we can simply rearrange this equation, to give us

$$\epsilon_{n+1} = \frac{f'(x_{root})\epsilon_n + f''(x_{root})\epsilon_n^2 - (f'(x_{root})\epsilon_n + \frac{1}{2}f''(x_{root})\epsilon_n^2}{f'(x_{root}) + f''(x_{root})\epsilon_n}$$

$$\epsilon_{n+1} = \frac{f''(x_{root})\epsilon_n^2}{2f'(x_{root})}$$

So we can see that the error at each iteration should decrease quadratically, a large improvement on the bisection method. This relationship above assumes that $f'(x_{root}) \neq 0$, and if the derivative is 0 then the algorithm will not converge on a root. For further details see reference [3].

It is possible for the Newton-Raphson method to not converge for other reasons as well. If the chosen initial value for x_0 is too close to a local maxima or minima, or too close to an asymptote the method may fail to converge on a root of the equation.

Figure 6 shows the absolute error over the iterations for three different runs, first using the bisection method and then two seperate runs of the Newton-Raphson method starting from different points. The bisection method starts with upper and lower bound 10 and -10 respectively. The first Newton-Raphson run starts with initial guess $n_0 = 10$, and the second starts with $n_0 = -10$. We can see from the graph that both the Newton-Raphson method runs converge on the root much quicker than the bisection method, which is as we expect as we have already seen that the Newton-Raphson method converges quadratically compared to the bisection methods linear convergance.

We can also see that the second Newton-Raphson run converges very quickly, within five iterations, which is to be expected as the initial guess input to the algorithm is quite close to the actual root. The first run for the method takes a little bit longer, and has some fluctuations in the error as we converge.

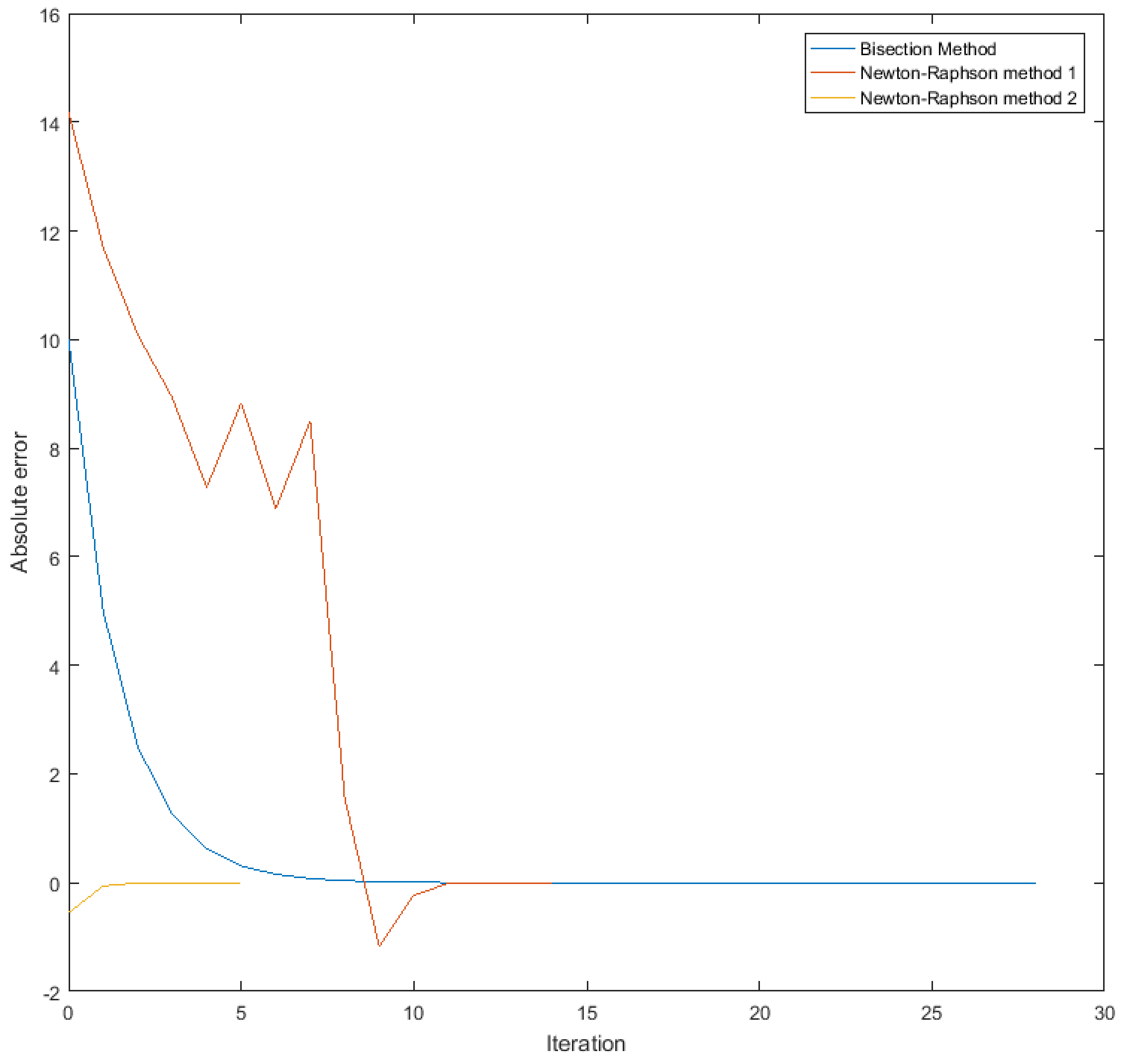


Figure 6: Graph showing the error at each iteration for the bisection method and two seperate runs of the Newton-Raphson method.

References

[1] Autar K. Kaw. *Holistic Numerical Methods.* URL: `http://nm.mathforcollege.com/topics/textbook_index.html`.

[2] Eric W. Weisstein. *Golden Ratio.* URL: `http://mathworld.wolfram.com/GoldenRatio.html`.

[3] Eric W. Weisstein. *Newton's Method.* URL: `http://mathworld.wolfram.com/NewtonsMethod.html`.