Worksheet 1
Know Your Computer's Limitations

Josh Wainwright
UID:1079596

Contents

---

1 Know Your Computer's Limitations

1.1 Simple Unix Commands

These simple commands were entered into the command line and the resulting output recorded.

1.1.1 `mkdir compphys`

Creates a directory in the current folder called "compphys".

```
$ mkdir -v compphys
mkdir: created directory 'compphys'
```

1.1.2 `cd compphys`

Changes directory into the folder called "compphys".

```
$ cd compphys
~compphys $
```

1.1.3 `cat > file1.txt`

Writes the user input from the terminal to the file called "file1.txt".

```
$ cat > file1.txt
this is my first file

$
```

1.1.4 `ls`

Lists the non-hidden files and folders in the current directory.

```
$ls
file1.txt
```

1.1.5 `more text1.txt`

Using the file pager "more", view the contents of the file called "file1.txt".

```
$ more file1.txt
this is my first file
```

1.1.6 `xclock &`

Starts the program "xclock" to display a graphical clock face and forks the process from the terminal.

```
$ xclock &
$
```

1.1.7 `whoami`

Displays the username of the current user logged in.

```
$ whoami
jaw097
```

1.1.8 `man ls`

Displays the manual for the command named "ls" in the PATH directory

```
$ man ls
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.
```

```
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 ..
```

1.1.9 top

Displays linux tasks with information about them with ability to manage these processes.

```
$ top
top - 12:08:39 up 70 days,  3:45, 12 users,  load average: 0.00, 0.00, 0.03
Tasks: 230 total,   1 running, 229 sleeping,   0 stopped,   0 zombie
Cpu(s):  9.1%us,  0.1%sy,  0.5%ni, 90.2%id,  0.1%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   8161612k total,  8082896k used,    78716k free,  3748868k buffers
Swap:  1020024k total,      208k used,  1019816k free,   904568k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1267 root      15   0  767m 601m 100m S  1.9  7.6   2:14.83 winbindd
 7112 jaw097    15   0 14840 1140  752 R  1.9  0.0   0:00.01 top
    1 root      15   0 10364  636  544 S  0.0  0.0   0:02.95 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.03 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.09 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.14 migration/1
```

1.1.10 killall xcloock

The running process called xclock is stopped.

```
$ killall xclock
[2]+  Terminated              xclock
```

1.1.11 ps -u

Displays information about the currently running processes started bu the user jaw097.

```
$$ ps -u jaw097
  PID TTY          TIME CMD
 4412 ?        00:00:01 sshd
 4417 pts/0    00:00:00 bash
 6449 ?        00:00:00 sshd
 6472 pts/9    00:00:00 bash
```

1.2 The Silver Ratio

The Silver ratio, ϕ, is defined to be

$$\phi = \frac{-1+\sqrt{5}}{2}, \tag{1}$$

and is simply the reciprocal of the Golden ratio. The silver ratio satisfies the expression

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

as shown below.

$$\begin{aligned}
\phi^{n+1} &= \phi^{n-1} - \phi^{n} \\
\phi^{n} &= \phi^{n-1} - \phi^{n+1} \\
&= \phi^{n}\phi^{-1} - \phi^{n}\phi^{1} \\
&= \phi^{n}(\phi^{-1} - \phi) \\
1 &= \phi^{-1} - \phi \\
\phi &= 1 - \phi^{2} \\
\phi^{2} + \phi - 1 &= 0
\end{aligned}$$

This can be solved as a linear quadratic equation.

$$\phi = \frac{-1 \pm \sqrt{5}}{2}$$

This is the given value for the Silver Ration, ϕ, and so the equation,

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

is satisfied.

This means that the value of ϕ^n can be calculated by subtraction using equation 2. By specifying a value for $\phi^0 = 1$ and for $\phi^1 = \phi$, this equation can then be used as a recursion relation.

Equation 2 is used to calculate the values of the silver ratio raised to different powers, using the recursion relation demonstrated above. To demonstrate the limitations, the program is run in both float and double. Under these conditions, when the program is run, the output depends on the size of the address space available, i.e. whether the variable is assigned double or float precision.

When using double precision, there are approximately $\pm 1.7 \times 10^{\pm 308}$ values, whereas with float precision, since it assigns a smaller address space, there are only $\pm 3.4 \times 10^{\pm 38}$ possible values. The differences in address space mean that errors occur at different rates when performing itterative calculations.

The errors occur more quickly when using float. The iterations output values which are consistent with the directly calculated vales but errors build up so that by the 16th iteration, the float value is 3.6% different to the correct value as calculated by direct multiplication. The calculations using double precision however develop smaller errors so the values deviate more slowly. It takes until the 36th iteration to reach a comparable error, 2.7%.

As shown above, there are two possible values that satisfy the recursion relation used to calculate the value of ϕ^n. This is another cause of the deviation away from the accepted value. As the natural computational errors increase, the value of ϕ^n tends to ϕ_1^n where ϕ_1 is the alternative value, $\phi_1 = \frac{-1-\sqrt{5}}{2}$. After this, the value tends back towards the original value and so forth. The resulting pattern and the cumulative errors means that the value being calculated grows enormously, as can be seen in figure 1, getting further from the correct answer.

1.2.1 Results From Itterative Calculation of ϕ^n

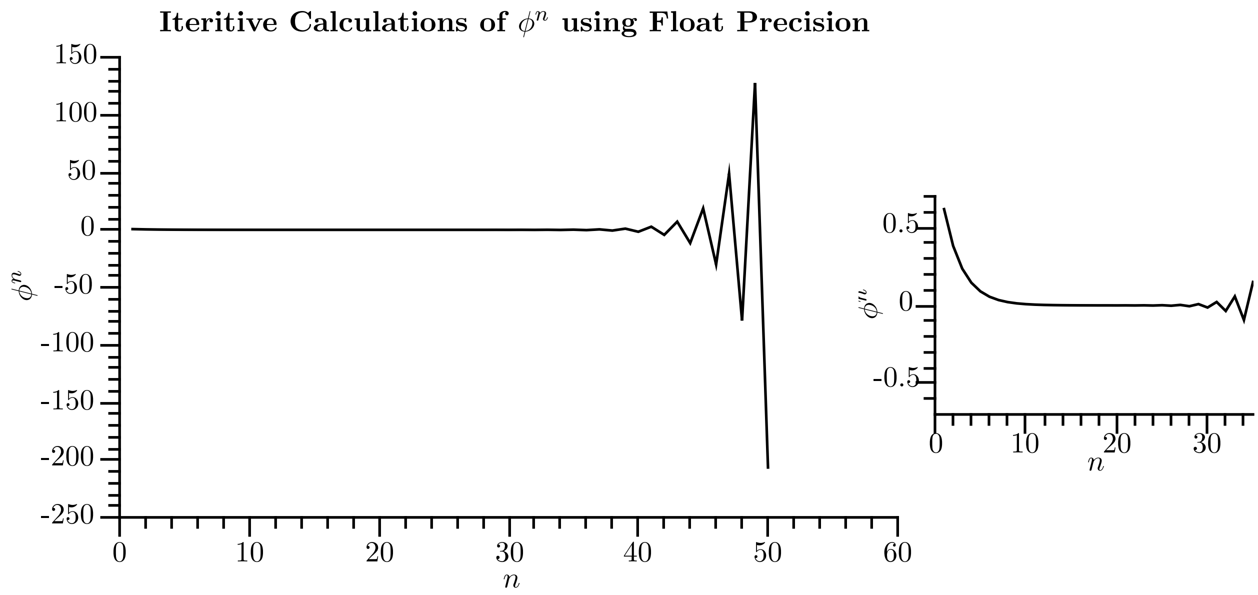

Figure 1: Graph showing the progression of the float iterative calculation as the errors increase, with insert showing initial detail while errors are small.

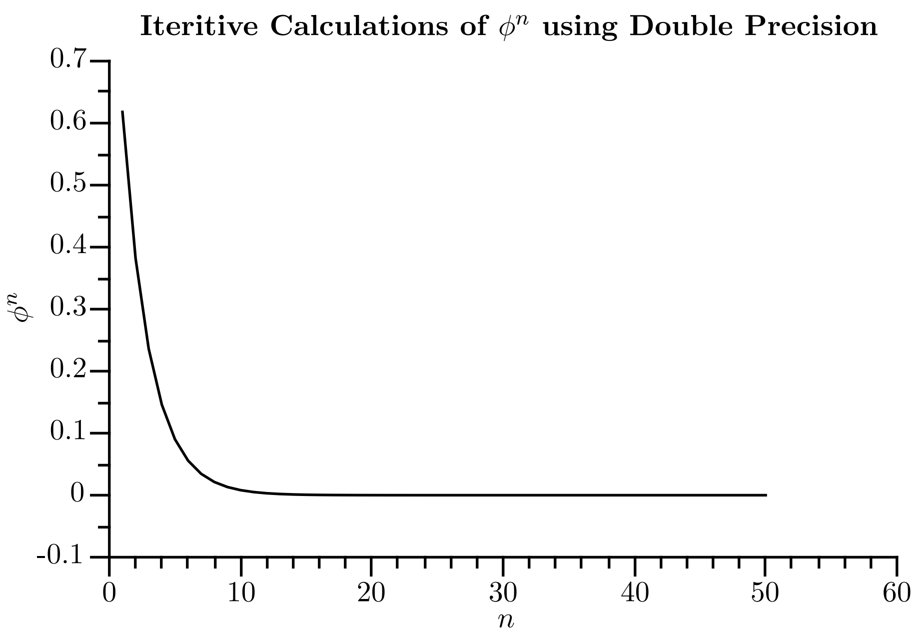

Figure 2: Graph showing the progression of the double iterative calculation. The errors are now reduced so that the value tends to zero.

n	Double	Float	n	Double	Float
1	0.618034	0.618034	26	3.68E-06	-0.00198901
2	0.381966	0.381966	27	2.28E-06	0.00322652
3	0.236068	0.236068	28	1.41E-06	-0.00521553
4	0.145898	0.145898	29	8.70E-07	0.00844204
5	0.0901699	0.09017	30	5.37E-07	-0.0136576
6	0.0557281	0.055728	31	3.32E-07	0.0220996
7	0.0344419	0.0344421	32	2.05E-07	-0.0357572
8	0.0212862	0.0212859	33	1.27E-07	0.0578568
9	0.0131556	0.0131562	34	7.81E-08	-0.093614
10	0.00813062	0.00812972	35	4.90E-08	0.151471
11	0.005025	0.00502646	36	2.91E-08	-0.245085
12	0.00310562	0.00310326	37	1.98E-08	0.396556
13	0.00191938	0.0019232	38	9.32E-09	-0.64164
14	0.00118624	0.00118005	39	1.05E-08	1.0382
15	0.000733137	0.000743151	40	-1.19E-09	-1.67984
16	0.000453104	0.000436902	41	1.17E-08	2.71803
17	0.000280034	0.000306249	42	-1.29E-08	-4.39787
18	0.00017307	0.000130653	43	2.46E-08	7.1159
19	0.000106963	0.000175595	44	-3.75E-08	-11.5138
20	6.61E-05	-4.49E-05	45	6.20E-08	18.6297
21	4.09E-05	0.000220537	46	-9.95E-08	-30.1434
22	2.53E-05	-0.000265479	47	1.62E-07	48.7731
23	1.56E-05	0.000486016	48	-2.61E-07	-78.9165
24	9.64E-06	-0.000751495	49	4.23E-07	127.69
25	5.96E-06	0.00123751	50	-6.84E-07	-206.606

Table 1: This table shows the values as calculated by the recursion relation when using float and double precision.