PHYS 619 (Computational Physics) — Lab & Homework 01 (due 01/31)

Helmut G. Katzgraber and Amin Barzegar
Department of Physics and Astronomy, Texas A&M University

General Instructions — Homework is due one week after it was discussed in the lab. I ask you to please send a tarball of your version control directory using the TAMU filex system [https://filex.tamu.edu]. In the title please add the following string

```
LN_PHYS619_HWYY
```

where `LN` is your last name. To generate the tarball, simply issue the following commands:

```
cd dir_where_exercise_solution_is
tar cvzf LN_PHYS619_HWYY.tgz LN_PHYS619_HWYY
```

then upload the file

```
LN_PHYS619_HWYY.tgz
```

where `YY` is the weekly exercise index – in this case `YY=01`. Problems marked with `L` are meant to be completed during the lab time. You may finish these at home (because you might need part of it for subsequent exercise sets), but that will not count towards the lab grade. Problems marked with `H` are homework. However, if you have time, we encourage you to start with them during the lab time because you will have the unique opportunity to ask questions.

Problems marked with `E` are meant as extra credit problems. The numbers in parenthesis correspond to the percentage this question counts towards the grade. Example:

```
L1 [20]
```

means 'solve in lab, 20% of total grade.'

L1 [40] Set up a version control system using git — Browse the online documentation to understand how to correctly do this. This step is very important because you will need this version control system for all work you will do during the semester.

a) [10] Create a directory called `$HOME/Work` where you will do all your work in. In that directory, create a repository called

```
LN_PHYS619
```

for the course, where `LN` is your last name.

b) [10] Create one directory for each weekly assignment called

```
HWYY
```

where `YY` is the weekly index. In this case, `YY=01`. Inside this directory create directories for each problem, i.e.,

```
Z
```

where `Z` is the part/section of the problem.

c) [10] Using the version control system: Write a simple program that when run prints π up to 10 digits precision. Make sure you use proper indentation and comments in the text. Make sure this program is uploaded to your version control system. Explain in part (d) how π is being computed in your program.

d) [10] Write simple documentation explaining how to compile and run the program you developed in part (c). Store this information in a file called `README.txt` as part of this project in the version control system.

L2 [60] Computing the median of a data set with the bootstrap method — Produce a data set of random numbers in the range [0,9] with the following Perl code

```
perl -e 'srand(2018) ; printf("%d\n",int(rand(10))) for (1..10000)'
```

Make sure you store these numbers in a file that you will then analyze with your code. The function `srand(k)` seeds the random number generator. For this exercise, use `k=2018` and produce 10000 numbers.

Using these data, write a simple program that computes the median of the data set, as well as an error to the median using the bootstrap method. More information on the bootstrap method can be obtained at

`http://goo.gl/iMZIH`

Prof. A. Peter Young has written a nice document explaining the bootstrap method in detail. You can find details at

`http://goo.gl/tiITV9`

Again, make sure you use the version control system for this project.

H3 [60] Rounding errors — Consider the series

$$S_{up} = \sum_{n=1}^{N} \frac{1}{n} \qquad\qquad S_{dn} = \sum_{n=N}^{1} \frac{1}{n}$$

for finite N. Analytically, the result both S_{dn} and S_{up} should be the same and finite for finite N. You will have to make some figures. An example gnuplot plotting macro can be found at

`https://katzgraber.org/teaching/SS18/files/gnuplot.tgz`

Add PDF versions of the plot to your repository.

a) [10] Write a program to calculate S_{up} and S_{dn} in single precision for $N = 10^p$ with $p = 2, 3, \ldots 8$. Plot $|S_{up} - S_{dn}|$ as a function of p using gnuplot (PDF).

b) [10] Write a program to calculate S_{up} and S_{dn} in double precision for $N = 10^p$ with $p = 2, 3, \ldots 8$. Plot $|S_{up} - S_{dn}|$ as a function of p using gnuplot (PDF).

c) [10] Assuming that the double-precision result x is exact, show that the single-precision S_{up} is less accurate than the single-precision S_{dn}, i.e., plot $|S_{up} - x|$ and $|S_{dn} - x|$ as a function of p with gnuplot (PDF). Explain your result.

d) [30] Write a program to calculate S_{up} in single and double precision for $N = 10^p$ with $p = 2, 3, \ldots 9$. However, now compute the alternating sum, i.e.,

$$S_{up} = \sum_{n=1}^{N} \frac{(-1)^{n+1}}{n}$$

The exact limit for the alternating sum is $\ln(2)$. Plot $|S - \ln(2)|$ vs p with p up to 9 for both single and double precision using gnuplot (PDF). What do you observe?

H4 [40] Jackknife method — The goal is to compute the average of k random numbers in the interval $[0, 1]$ and estimate a statistical error bar using the jackknife method. For more details on the jackknife method, please see

`http://goo.gl/tiITV9`

First, produce a data set of random numbers in the range [0,1] with the following Perl code

```
perl -e 'srand(2018) ; printf("%f\n",rand()) for (1..1000)'
```

Store these numbers in a file that you will then analyze with your code. The function `srand(k)` seeds the random number generator. As before, use `k=2018` and produce 1000 numbers.

a) [10] Write a program that reads in the numbers and computes their average.

b) [10] Modify your program to also estimate the error of the average using the jackknife method.

c) [10] Increase the number of random numbers k to 10^7 and time your program for $k = p \cdot 10^7$ random numbers with $p = 1 \ldots 8$. Use gnuplot to make a plot of the timing, i.e., plot the wall clock time you determine with the Unix built-in function `time` vs k. To time your program, simply issue

```
time ./myprogram
```

and use the 'real' time. Add the PDF of the plot to your repository.

d) [10] What is the function that best describes the time complexity of your program? Add a text file to your repository with the answer.

E5 [5] — If your program in problem H4B took so long to run that you had to wait more than 15 minutes for the result, can you think of a way to re-write your program to make it more efficient?