Michigan Tech

UN5390: Scientific Computing I

Fall 2016

Assignment #04 (Due 11:59 am on Sunday, 9th Oct 2016; 10%)

Guidelines

1. Every problem in this assignment has a value beyond its numerical score. It is either a *swan kick* moment for what you already know OR a *wax on, wax off* training for something you will learn soon OR a combination thereof. Treating every problem as a tiny research project is very highly encouraged

2. You may seek help from anyone as long as you understand all such help and cite it appropriately. Write the programs and scripts in compliance with programming etiquette discussion as clarity and readability account for 25% of the assigned credit

3. Read through/Review the course material and additional references, if necessary. Give yourself enough time to complete the assignment on or ahead of time, and show continuous improvement. Check the submission for spelling and grammatical correctness. Contact the instructor if you didn't understand the material

4. Typeset your solutions/answers in `${USER}_04.tex`, `${USER}_04.pdf`, and other files as indicated throughout this assignment. Include imagery to explain your answers as and when necessary. Do not include the source code or data in the `${USER}_04.tex` document. There needs to be at least one commit per problem to the GitHub repository to show timely work, demonstrate your work ethic, and a final tagged version of the assignment for submission to earn any credit

5. Graduate students need to solve first six (worth 1.66% each) while undergraduate and non-traditional students need to solve first five problems (worth 2.00% each). Remaining problems represent an opportunity to *do a little more* and are worth 0.50% each. Deadline for seeking help from the instructor: 7:59 am on Friday, 7th Oct 2016

Assignment submission workflow

1. Replace `#1` with the appropriate problem number. Keep all files and folders under

   `${UN5390}/CourseWork/Week_04/${USER}_04/`

2. Before starting to work on a problem (or part thereof)

   ```
   cd ${UN5390}
   git pull
   ```

   Suggestions to correct/improve your answers, if any when requested by you, can be found in `${UN5390}/CourseWork/Week_04/suggested_corrections_04.txt`

3. After each problem (or a part thereof), commit to the repository

   ```
   cd ${UN5390}/LaTeXTemplates/Course/
   git pull
   git add ${USER}.bib
   cd ../../CourseWork/Week_04/
   git add ${USER}_04
   git commit -m "Submitting problem #1 in assignment #04"
   git push origin master
   ```

4. When the assignment in its entirety is ready for final submission

   ```
   cd ${UN5390}/CourseWork/Week_04/
   git pull
   git add ${USER}_04
   git tag -a a04 -m "Submitting assignment #04"
   git push origin a04
   ```

5. If you made edits to your submission after tagging but before the final due date, add a minor revision number to the tag

   ```
   cd ${UN5390}/CourseWork/Week_04/
   git pull
   git add ${USER}_04
   git tag -a a04.1 -m "Submitting assignment #04"
   git push origin a04.1
   ```

   and so on

Problem 1

Discuss with your research advisor and list the compliance requirements in effect for her/his research project(s) imposed by the federal and/or state governments, and/or the funding agency. Include in your list, using course or other material as reference, consequences of violations such requirements. Attempt to articulate – if possible in terms of time, finance and/or opportunities – the loss incurred by your advisor, the research group, the department and/or the university as a result of such violations.

If you do not have a research advisor, write one short paragraph (4-6 meaningful sentences) each for any three federal compliance regulations [1–9]. Keep it under one single-sided page.

Problem 2

What does the pseudo-code given below accomplish? Can the number of arithmetic operations be estimated in a functional form for a given value of d? Write an aptly named program with meaningful comments, etc., to verify your claim for various values of d. Discuss at least one potential way to improve the efficiency of this approach.

```
a, b, c, and d are non-negative integers; d is greater than one

OUTER LOOP BEGINS: a ranges from two through d in steps of one
   Set c to be zero
   INNER LOOP BEGINS: b ranges from 1 through a in steps of one
      If a is integer-divisible by b, then increment c by one
   INNER LOOP ENDS
   If c equals two, then print a
OUTER LOOP ENDS
```

Problem 3

Rounding is a procedure for choosing the representation of a real number in a floating-point number system [10]. Machine epsilon (ϵ) imposes a fundamental limit on the accuracy of numerical computations, and is defined as the smallest floating-point number such that $1 + \epsilon > 1$. The following pseudo-code may be used to estimate the value of ϵ.

```
a is a floating-point number, and n is an integer
Set a to be one, and n to be zero

LOOP BEGINS: as long as one is less than one added to a
   Divide a by two, and increase n by one
LOOP ENDS

Multiply a by two, and decrease n by one
Print a and n
```

Convert the above pseudo-code into a program, `machine_epsilon.EXTN`. Compute the value ϵ for double-precision floating-point numbers and compare it with the language's built-in value, if available. The program must save the value of ϵ after each iteration along with iteration number, n, in a file named `machine_epsilon.dat`. Plot ϵ as a function of n (i.e., the data in `machine_epsilon.dat`) using Gnuplot [11–13]. Save the necessary Gnuplot instructions in `machine_epsilon.gnu`, and the plot in `machine_epsilon.eps`. Does the plot make sense? Is the computed value of ϵ in agreement with that found in literature?

Problem 4

The known value of Euler's constant, e, is 2.718281828459045. The following two expressions are well-known for computing the value of e^x.

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \qquad e^x = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n$$

Write a well-commented and modularized program, `exp_x.EXTN`, to implement these expressions. The program must be generic enough to compute e^x for any value of k, x, n, and for a specified error.

For the first expression, compute the value of e^x for a reasonably large k such that error is less than one part in a million. For the second expression, use $n = 10^{308}$. Use $x = 1$ in both cases. Explain the difference in computed value of e from these expressions, if any.

Problem 5

What is the probability of getting exactly h heads in n tosses for a given fairness factor, p? Suppose that a coin is tossed 10 times. Suppose that it results in seven heads and three tails. Can the coin be considered fair? Explain with a mathematical reasoning.

Write a program, `CoinFlip_single.EXTN`, to simulate the tossing a fair coin and record the the number heads. Run the program for varying number of tosses, 10^m (meaning, m ranges from 1 to 8 in steps of 1) and store the results in `CoinFlip_single.dat`. Plot the theoretical probability for heads obtained by tossing a fair coin, and the results from your computing experiment using Gnuplot. Save the Gnuplot instructions in `CoinFlip_single.gnu` and the plot in `CoinFlip_single.eps`. Interpret the plot. Does this change your answer – about the fairness of a coin when it results in seven heads in 10 tosses?

Write a program, `CoinFlip_multiple.EXTN`, that performs the following task: *toss a coin 10 times and record the number of heads*. The program must be capable of repeating this task 100 times for each of the following values of fairness factor, $p = 0.00 : 0.10 : 1.00$ (meaning, p ranges from 0.00 to 1.00 in steps of 0.10). Run the program and store the results in `CoinFlip_multiple.dat`. Save the Gnuplot instructions in `CoinFlip_multiple.gnu` and the plot in `CoinFlip_multiple.eps`. It should look like Fig. 1. Interpret the plot.

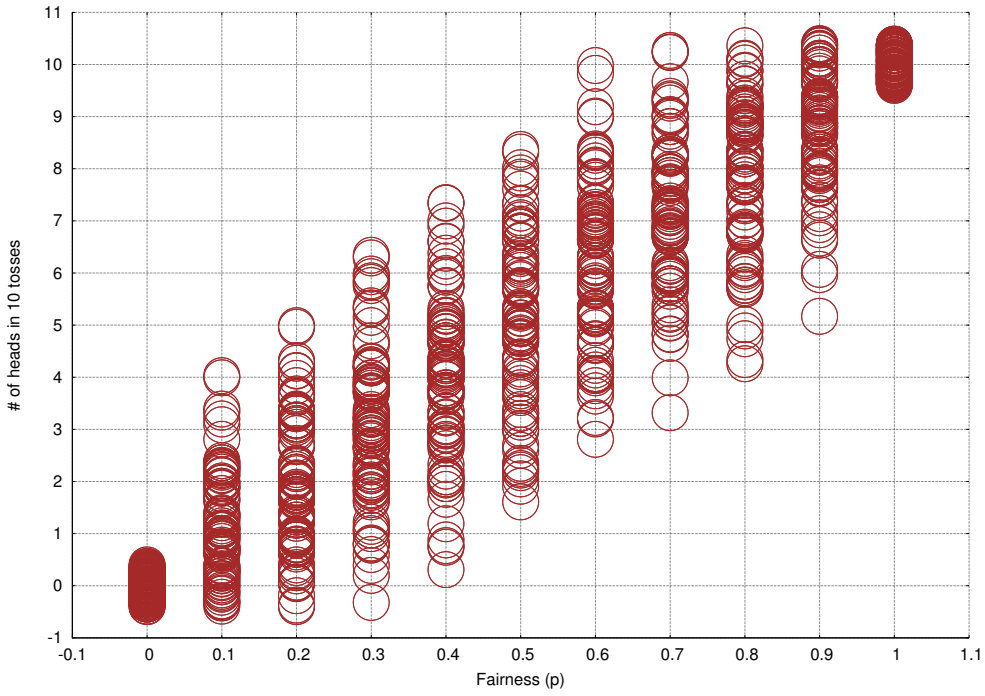

Figure 1: Jitter plot of data from coin flip (multiple) experiment

Problem 6

Given two locations described by their respective latitude (θ) and longitude (ϕ) coordinates, spherical law of cosines (SLC) and haversine formula (HF) are two of the commonly used approaches to compute the distance between them.

$$d_{\text{SLC}} = \arccos\left[\sin\theta_1 \sin\theta_2 + \cos\theta_1 \cos\theta_2 \cos\Delta\phi\right] \times 60 \times 1.1515$$

$$d_{\text{HF}} = 2 \times R_E \times \arctan 2\left[\sqrt{\sin^2(\Delta\theta/2) + \cos\theta_1 \cos\theta_2 \sin^2(\Delta\phi/2)},\right.$$
$$\left.\sqrt{1 - \left\{\sin^2(\Delta\theta/2) + \cos\theta_1 \cos\theta_2 \sin^2(\Delta\phi/2)\right\}}\right]$$

Write a program, `compute_distance.EXTN`, that computes the distance ($\overline{AB}$, $\overline{AC}$ and $\overline{BC}$) using both these approaches for locations described by A (47.1195 N, -88.5470 E), B (47.4688 N, -87.8884 E) and C (47.119509 N, -88.5470 E).

Does one approach fair better over the other? R_E is the radius of earth (i.e., 3960 miles), $\Delta\theta = |\theta_1 - \theta_2|$, and $\Delta\phi = |\phi_1 - \phi_2|$. The program should include at least two functions: `distance_slc(`θ_1`, `ϕ_1`, `θ_2`, `ϕ_2`)` and `distance_haversine(`θ_1`, `ϕ_1`, `θ_2`, `ϕ_2`)`. The following results may be used, if necessary, to simplify the computations.

$$\lim_{x\to 0} \frac{\ln(1 + x)}{x} = 1 \qquad \lim_{x\to 0} \cos(x) = 1 - \frac{x^2}{2} \qquad \lim_{x\to 0} \tan(x) = x$$

Problem 7

[Optional] Investigate the historical and computational significance of $\zeta(b)$. Write an aptly named, well-commented and modularized program to evaluate it for $b = 0, 1, 2, 3, ..., 39$. Extend b from 40 through 100 and identify the values of b that fit the prior pattern of $\zeta(b)$, if any.

$$\zeta(b) = b^2 + b + 41$$

Problem 8

[Optional] Write an appropriately named, well-commented and modularized program that computes the closest value of 0.10 as a sum of negative integer powers of two without exceeding it.

Problem 9

[Optional] A partition of a positive integer n, denoted by $p(n)$, is defined to be a sequence of positive integers whose sum is n [14, 15]. Although the order of the summands is irrelevant when writing partitions of n, it is a common practice to write the summands in a non-increasing order for consistency. Partitions of $n = 5$ are given below as an example.

$$5 = \begin{cases} 5 \\ 4+1 \\ 3+2 \\ 3+1+1 \\ 2+2+1 \\ 2+1+1+1 \\ 1+1+1+1+1 \end{cases}$$

As can be noted, there are seven unique summands for $n = 5$. Thus, partition of 5 is denoted by $p(5) = 7$. The value of $p(n)$ for $0 \leqslant n \leqslant 10$ and $n = 100$ are shown below.

$$\begin{aligned} p(0) &= 1 & p(6) &= 11 \\ p(1) &= 1 & p(7) &= 15 \\ p(2) &= 2 & p(8) &= 22 \\ p(3) &= 3 & p(9) &= 30 \\ p(4) &= 5 & p(10) &= 42 \\ p(5) &= 7 & p(100) &= 190{,}569{,}292 \end{aligned}$$

Write a program, `integer_partitions.EXTN`, that computes the number of partitions for any given n using brute force counting approach. A commonly used generating function, especially for computing larger values of n, is given by

$$p(n) \simeq \frac{\exp\left[\pi\sqrt{2\,n/3}\right]}{4\,n\sqrt{3}}$$

Compare the results from your brute force counting approach and the generating function for $n = 0, 1, 2, ..., 10$ and $n = 100$.

Problem 10

[Optional] Write a well-commented BASH script, `compile_execute_plot.sh`, with suitable error/exit codes to accomplish the following.

1. Compile the program (iff using a compiled programming language) and run it (iff the program was successfully compiled, as applicable). Save the data in a ASCII file
2. Plot the data in ASCII file using Gnuplot iff it's not empty. Save it as an EPS file
3. Print the time required for each phase in human readable format

It's usage should be as follows. `PROGRAM_BASENAME` is the base name of source code (e.g., `CoinFlip_single` or `CoinFlip_multiple`).

```
compile_execute_plot.sh PROGRAM_BASENAME
```

Suppose that you have used C programming language and the code is contained in `CoinFlip_single.c`. Then the script, when executed, should do the following.

1. Compile the program to produce `CoinFlip_single.x`, and run it. Save the data in a ASCII file, `CoinFlip_single.dat`
2. Plot the data in `CoinFlip_single.dat` using Gnuplot instructions in `CoinFlip_single.gnu` iff the data file is not empty. Save the image as `CoinFlip_single.eps`
3. Print the time required for each phase (i.e., compilation, execution, and plotting) in human readable format

Suppose that you have used MATLAB and the code is contained in `CoinFlip_single.m`. Then the script, when executed, should do the following.

1. Run `CoinFlip_single.m` using MATLAB. Save the data in a ASCII file, `CoinFlip_single.dat`
2. Plot the data in `CoinFlip_single.dat` using Gnuplot instructions in `CoinFlip_single.gnu` iff the data file is not empty. Save the image as `CoinFlip_single.eps`
3. Print the time required for execution and plotting phases in human readable format

References

[1] Export Administration Regulations (1979).
http://www.bis.doc.gov/index.php/regulations/export-administration-regulations-ear.

[2] Family Educational Rights and Privacy Act (2007).
http://www2.ed.gov/policy/gen/guid/fpco/ferpa/.

[3] Federal Information Security Management Act (2002).
http://www.dhs.gov/fisma.

[4] Gramm-Leach-Bliley Act (2003).
https://www.fdic.gov/regulations/compliance/manual/8/VIII-1.1.pdf.

[5] Health Information Portability and Accountability Act (1996).
http://www.hhs.gov/ocr/privacy/.

[6] Health IT for Economic and Clinical Health Act (2009).
http://www.healthit.gov/policy-researchers-implementers/health-it-legislation.

[7] International Traffic In Arms Regulations (1976).
https://www.pmddtc.state.gov/regulations_laws/itar.html.

[8] Payment Card Industry Data Security Standard (2007).
https://www.pcisecuritystandards.org/security_standards/.

[9] Red Flags Rule (2010).
http://www.ftc.gov/tips-advice/business-center/privacy-and-security/red-flags-rule.

[10] IEEE 754: Standard For Binary Floating-Point Arithmetic.
http://grouper.ieee.org/groups/754/.

[11] Gnuplot Official Website.
http://www.gnuplot.info/.

[12] Not So Frequently Asked Questions in Gnuplot.
http://lowrank.net/gnuplot/index-e.html.

[13] Scientific Plots Using Gnuplot.
http://www.gnuplotting.org/.

[14] Partition (Wikipedia).
https://en.wikipedia.org/wiki/Partition_(number_theory).

[15] Partition (Wolfram).
http://mathworld.wolfram.com/PartitionFunctionP.html.