ASTR615 Fall 2015 Problem Set #1

Due Sept 23rd, 2015

This problem set is designed to familiarize you with computer programming, shell scripting, and simple visualization. You will also learn about benchmarking and optimization.

HINT: If you plan to use C for this assignment, and you're new to the language or need a refresher, be sure to read the *Intro to C* document first. It provides information on, for example, using internal clocks to measure code performance. You should also read the other introductory documents on Unix (especially scripting) and visualization.

1. Write a program to time the CPU expense of various mathematical operations, as listed in the table below. Do each operation n times, where $0 < n \leq 10^9$ is supplied by the user. Your program should output the final value obtained after n cumulative iterations of each operation, and the CPU time required to complete them. For the moment, do not make use of compiler optimizations.

Operand Type	Starting Value	Operation
integer	0	add 5
integer	0	subtract 5
integer	1	multiply by 1
integer	1	divide by 1
double-precision	0.0	add 5.0
double-precision	0.0	subtract 5.0
double-precision	1.0	multiply by 1.000001
double-precision	1.0	divide by 1.000001
double-precision	1.1	take square root (`sqrt(x)`)
double-precision	1.1	raise to the 0.5 power (`pow(x,0.5)`)

(a) Write a shell script to construct a table of execution times for double-precision addition and multiplication as a function of n for $n = 10^6$, 3×10^6, 10^7, 3×10^7, 10^8, 3×10^8, and 10^9. Do the square-root and raise-to-the-power functions give much different execution times? Comment.

(b) Plot the data in the table, execution time as a function of n, using any plotting package you like. The axes should use a logarithmic scale. Estimate the number of floating-point additions and multiplications (MFLOPs) your code carried out on average per second.

(c) Redo one or more plots in question 1b after recompiling your code with optimizations turned on (e.g., try the "`-O2`" compiler flag) and plot the results. Is there a difference? Also try a different compiler if possible, with and without optimizations (e.g., if you used `gcc`, try `icc` next.[1]) Comment on the results and suggest why differences exist.

[1] On the department Linux machines, type `source /astromake/astromake_start` followed by `astroload intel` in order to get `icc` to work. The FORTRAN equivalent is `ifort`.

2. How long did it take you to finish this problem set?

When submitting this assignment, package together the code, script(s), output, and your answers to specific questions into a single file (ideally a tar ball or zip file that unpacks into a directory whose name is your last name) and e-mail it to me (`ricotti@astro.umd.edu`). Include the Makefile or README file with instructions on how to compile and run your code/scripts(s). Be sure to make your code and script "bomb proof" as much as possible.

<ins>CHECKLIST</ins>

Your submitted assignment should contain at least the following:

- Source file for benchmark code.
- Scripts for generating runtimes as a function of n and plot the results.
- Document providing written answers to questions in PDF format.
- Plots showing runtimes and optionally MFLOPs as a function of n; these can be included in the written document. Feel free to include multiple plots on a page.
- A Makefile and/or README file written in plain text describing all the files being submitted, how to run the source code, and how to run the script. Optionally include information on how to generate the plots from the script output, but this is not required.