ME614 Spring 2017 - HPC Homework

Introduction to Parallel Computing

(Due February 22, 2017)

Please prepare this homework in the folder `/homeworkHPC/` of your git repository and submit it following the instructions in the last section of Homework 0. The main script should be named `homework.py` and placed in the `/homeworkHPC/code/` folder. The submitted code needs to run and create all of the required plots in the folder `/homeworkHPC/report/figures/` at once. The use of LaTeX for your report is strongly recommended (but not required) and you can start with a template from http://www.latextemplates.com/. Discussions and sharing of ideas are encouraged but individually prepared submissions (codes, figures, written reports, etc.) are required. A plagiarism detection algorithm will be run against all codes submitted. Do NOT include in your git repository files that are not required for homework submission (e.g. the syllabus, zip files with python libraries, other PDFs, sample python sessions etc). Follow exactly the suggested folder structure of your homework git repository as all of the grading tools on the instructor's end are scripted. Points will be deducted from late submissions at a rate of 20% of the overall homework value per day late. Homeworks are due at 11:59 PM of the due date.

Problem 1

The integral of any given univariate function that is discretely defined over N equispaced points between a and b (inclusive) with indices ranging from 0 to $N-1$ can be numerically approximated with Simpson's rule as,

$$\int_a^b f(x)dx \approx \hat{u}(x) = \sum_{i=0}^{N-2} \frac{\Delta x}{6}\left[f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right] \tag{1}$$

where, Δx is the grid spacing and x_j is the value of the x-coordinate at the subscripted index.
For a function $f(x) = \mathbf{e}^x \tanh(x)$ defined on [-10,10], write an MPI code that evaluates the integral of $f(x)$ with a variable number of points using equation (1) and times the calculation. Use `float64` precision for all numbers. If the total number of points is not divisible by the number of processes, assign the excess points to the last process. Use the `MPI` function `Reduce` with the operation `MPI.SUM` to sum up the calculations of all processes on process 0. To make sure that you are capturing the overall time taken by all processes from start to end, set up barriers *before you start your timer* and *before you stop your timer*. This will ensure that all processes are synchronized and the true overall execution time is reported.
The analytical anti-derivative of $f(x)$ for the given range is,

$$\int_{-10}^{+10} f(x)dx = u(x) = -\frac{1}{\mathbf{e}^{10}} + \mathbf{e}^{10} + 2\tan^{-1}\left(\frac{1}{\mathbf{e}^{10}}\right) - 2\tan^{-1}\left(\mathbf{e}^{10}\right). \tag{2}$$

(a) Using $N_{\text{procs}} = 1$, 2, 4, 8, and 16 processes, evaluate and plot in a single figure on a log-log scale the absolute truncation error, $\epsilon = |\hat{u} - u|$ against the inverse grid spacing Δx^{-1} by varying the total number of points, N between $10^1 + 1$ and $10^9 + 1$. What is the behavior of the truncation error? Comment on your results. [30%]

(b) Once again for all values of N_{procs} in part (a), perform weak and strong scaling analysis' of your MPI program. For definitions of weak and strong scaling, refer to https://www.sharcnet.ca/help/index.php/Measuring_Parallel_Scaling_Performance. Use a total number of points, $N = 10^8 + 1$ for strong scaling and points per processor, $N/N_{\text{procs}} = 10^7$ for weak scaling. Plot (separately) the weak and strong scaling efficiencies against the total number of processes, with horizontal dashed reference lines for 50%, 75%, and 100% efficiencies. [40%]

(c) With a fixed number of points, $N = 10^5 + 1$ and by varying the number of processes N_{procs} between 1 and 16 in increments of 1, evaluate the absolute truncation error for each case and plot against the corresponding number of processors. Also plot a dashed line corresponding to the average of all cases (use the function `axhline` from the module `pylab` or `matplotlib.pyplot` to plot a horizontal line at a given y-coordinate). Does the error vary for each case? Comment on your results. [30%]

Note: Some or all of these questions might require you to have your program output data into a text or binary file that you then postprocess with a secondary script after a certain number of runs. Look into the python module `pickle` to be able to store entire objects to disk as opposed to printing out single entries or lines into a text file.