PHYS:5905 Homework #6a

Please submit your solutions as a single PDF file with answers to the questions asked.
Please complete required problems before lecture on Thursday, February 28, 2019.
Note that the only thing required to be returned is the plot generated by problem #5.

1. (Required) **Getting Online on the Argon Cluster**
   For this exercise, I will provide directions on how to connect to the University of Iowa's Argon cluster from the Windows workstations in 201 VAN. You are welcome to complete these steps from any computer of your choice (including personal laptops). The point of this exercise is to learn how to connect to the Argon cluster using secure shell `ssh` with `Xwindow` forwarding of windows from processes running on the remote server.

   (a) First, be sure that you have completed Homework #1 to request an account on the Argon cluster and that you have received an email from Research Services that your Argon account has been created.

   (b) Online documentation for the High Performance Computing (HPC) Cluster systems run by ITS Research Services at the University of Iowa can be found at
   `https://wiki.uiowa.edu/display/hpcdocs/Cluster+Systems+Documentation`
   This online documentation Wiki has a vast amount of information on accessing and using the Argon cluster, and we will refer to specific pages of this documentation frequently.

   (c) In order to access the Argon cluster using the directions here, it is necessary to have the Duo two-factor authentication set up for your HawkID. If you do not have this set up already, you can find information on the security feature relevant to the Argon cluster at
   `https://wiki.uiowa.edu/display/hpcdocs/Two-factor+authentication`
   including links to the ITS webpage for setting up the Duo service. If you have not already done this, please follow the directions here to set this up.

   (d) Log onto the workstation using your HawkID and password.

   (e) We will be using the FastX software to establish an `ssh` connection to the Argon cluster and to forward windows from the remote server to the local workstation using the following steps:

      i. Launch FastX 2, located in the FastX 2 folder.

      ii. When the FastX window comes up, click on the "+" sign in the upper right-hand corner and choose SSH.

      iii. Type in the following information for the connection:
      `Name:` `Argon`
      `Host:` `argon.hpc.uiowa.edu`
      `Port:` `22`
      `User:` *HawkID*
      `Sci:` leave as is

      Check the `Forward Agent Connections` box

      iv. If it gives you a warning that the host is not recognized, go ahead and click Continue.

      v. Enter your HawkID password

      vi. Respond on your secondary device (mobile phone) using the Duo app to complete the two-factor authentication.

   (f) A new window will appear with the title "Argon: (argon.hpc.uiowa.edu)." Click the "+" in the upper right-hand corner and choose xterm (or another choice if you prefer).

   (g) An xterm window will appear, and you will be successfully connected to Argon.

(h) If you start a process that brings up a window (*e.g.*, `emacs`, `gv`, or anything else), the window should appear on the terminal of your local workstation. This makes editing programs on the remote server and quick analysis of program results very simple.

(i) To terminate the xterm window, simply enter `exit` at the xterm prompt, which will close the xterm window.

(j) To close the `ssh` connection to Argon that has been established through FastX, simply click the "X" in the upper right-hand corner of the "Argon: (argon.hpc.uiowa.edu)" window.

2. (Required) **Setting Up Argon Shell**

(a) Argon is an HPC cluster system with a heterogeneous mix of compute node types, with a total of 612 compute nodes running CentOS-7.4 Linux. The compute nodes have a wide variety of configurations with 24-, 32-, 40-, 56-, or 64-cores and 64, 96, 128, 192, 256, or 512 GB of memory. The specific configuration for all compute nodes is detailed at
`https://wiki.uiowa.edu/display/hpcdocs/Argon+Cluster`

(b) In this heterogeneous mixtures of nodes, there are 21 machines with Nvidia P100 accelerators, 2 machines with Nvidia K80 accelerators, 11 machines with NVidia K20 accelerators, 2 machines with Nvidia P40 accelerators, 13 machines with 1080Ti accelerators, and 18 machines with Titan V accelerators. In addition, there are 38 Xeon Phi 5110P Accelerator Cards and 11 Nvidia Kepler K20 Accelerator Cards installed on certain nodes in the cluster.

(c) The Argon cluster is physically split between two separate data centers: Information Technology Facility (ITF) and Lindquist Center (LC). Most of the nodes in the LC data center are connected with the OmniPath high speed interconnect fabric, while most of those in the ITF data center are connected with the InfiniPath fabric.

(d) Online documentation for using the all HPC clusters (including Argon) can be found at
`https://wiki.uiowa.edu/display/hpcdocs/Cluster+Systems+Documentation`

(e) Module Set Up: The first time, and only the first time, that we log onto Argon, we need to set up our `.bashrc` file so that the software modules that we need are loaded automatically.

i. To view the modules that are already loaded, use the command `module list`, and to see available modules, use `module avail`.

ii. To add the necessary lines to the `.bashrc` file, in your home directory, open the `.bashrc` file using `emacs` in the background using
`emacs .bashrc &`
A window will open showing the contents of your `.bashrc` file. Note that if the font size in emacs is too small to read comfortably, you may change the font size using `CTRL-X CTRL-+`
At the end of the file, after the line
`# User specific aliases and functions`
add the following two lines
```
export SYSTEM=argon
module load openmpi
```
Next hit `CTRL-x CTRL-s` to save the changes, and `CTRL-x CTRL-c` to quit `emacs`.

iii. Now you may exit your login shell and follow the previous instructions to log in via `ssh` again. Once you have logged on again, you may use `module list` to verify that the necessary modules are loaded.

3. (Required) **Compiling Parallel Programs on Argon**

(a) We are going to download the example parallel code, `HYDRO`, to Argon. Follow the instructions below:

i. Create a directory named `hydro` in your home directory on Argon
`mkdir hydro`

ii. Navigate into the hydro directory,
`cd hydro`

iii. Type the following command to download the tar archive of the `HYDRO` code

```
curl -o hyd190218.tar https://homepage.physics.uiowa.edu/~ghowes/teach/phys5905/codes/hyd190218.tar
```

iv. Note that the `HYDRO` code is also available at the PHYS:5905 course website at
`https://homepage.physics.uiowa.edu/~ghowes/teach/phys5905/codes5905.html`
in the Codes section.

v. Inside the `hydro` directory, unpack the tar file
`tar -xvf hyd190218.tar`

(b) Compiling the parallel code `HYDRO`:

i. The tar file for `HYDRO` contains a `Makefile` that allows for easy compilation of the code on different platforms (different computers). `HYDRO` is written in Fortran90 using MPI for parallelization.

ii. Compile the code by typing
`make`
This will produce an executable `hydro.e`

4. (Required) **Running Parallel Programs on Argon**

(a) Neon uses the Sun Grid Engine (SGE) as a scheduler for parallel jobs. More information can be found online at
`https://wiki.uiowa.edu/display/hpcdocs/Basic+Job+Submission`

(b) The examples below will use `HYDRO` with the sample input file `sample1.in`. The code requires the first argument after the executable to be the input file, thus the command will be
`hydro.e sample1.in`

(c) We will be running in batch mode, the preferable method for submitting jobs to shared computing resources. Submitting a job in batch mode puts the job into a queue to be run when resources become available.

(d) BATCH MODE: The usual method for running on a shared cluster or at a national supercomputing center is to run in batch mode, submitting your jobs using a script.

i. An example shell script for running on Argon is included in the tar archive, `sample1_argon.sh`, for submitting a batch job to SGE to run `HYDRO` (for the input file `sample1.in`) follows:

```
#!/bin/sh
# Job Submission script

#$ -q UI
#$ -l h_rt=00:10:00
#$ -pe 32cpn 32
#$ -N sample1
#$ -o sample1.log
#$ -j y
#$ -V
#$ -cwd

echo "Job begin:"`date`
echo "Run sample1 on Argon, 32 proc"
mpirun -n 32 hydro.e sample1.in
echo "Job end:"`date`
```

ii. Each of the options on the lines above specifies a different aspect of the run:
`-q UI` specifies the UI queue for the job (another option is `all.q`)
`-l h_rt=00:10:00` specifies the time limit in HH:MM:SS format
`-pe 32cpn 32` requests 32 cores using 32 cores per node
`-N sample1` specifies the name name of the job
`-o sample1.log` specifies the name of the file to send the standard output
`-j y` merges the stdout and stderr output,
`-V` imports environmental variables to the parallel SGE environment
`-cwd` places the output files in the current working directory.
The echo lines above simply write to the log file a few useful comments, but are not necessary.

iii. To submit the job, use
`qsub sample1_argon.sh`

iv. You can check the status of all of your submitted jobs using `qstat -u` *HawkID*
Be aware that, if your job runs immediately (and it often does), the job may be complete before you even complete issuing the `qstat` command.

v. If the job has run successfully, it will generate a number of output files in the present directory:
`sample1.log` is a log file with output information
`sample1.time` is a timing output (which currently does not work)
Also, it will output data files in the `data` directory of the form `sample1.out.####`.

5. (Required) **Plotting Result of `HYDRO` run Sample1.in**

(a) Full details on the `HYDRO` Example Parallel Hydrodynamics Code can be found in the documentation available on the course website at
`https://homepage.physics.uiowa.edu/~ghowes/teach/phys5905/codes5905.html`

(b) The output each time the code saves the output is put into a file *runname*`.out.####`. This is an ASCII text file with data in the following columns:

Column	Value
1	time
2	ix
3	iy
4	x
5	y
6	ρ
7	u_x
8	u_y
9	p
10	iproc

Note that, for the current scheme collecting all data on proc0 for output, the last column iproc=0 always.

The output from `HYDRO` can easily be viewed using `gnuplot`. The following steps will explain how to plot the output of the `HYDRO` code with the input file `sample1.in` and make postscript figures from this output.

i. Run `HYDRO` with the input file `sample1.in`.

ii. Navigate into the `data` directory of `HYDRO`. A listing of the directory will show
```
sample1.out.0000 sample1.out.0003 sample1.out.0006 sample1.out.0009
sample1.out.0001 sample1.out.0004 sample1.out.0007 sample1.out.0010
sample1.out.0002 sample1.out.0005 sample1.out.0008
```

iii. Start `gnuplot` with the command
`gnuplot`

In `gnuplot`, you can get help for any *command* by using
`help` *command*

iv. Set the data style to plot lines rather than points
`set style data lines`

v. Plot out the final values of u_x vs. x using the command
`plot 'sample1.out.0010' u 4:7`
If Xforwarding has been enabled when you established your `ssh` connection to log onto the remote machine, an Xwindow will pop up with your plot (this may take a little time for the secure connection to be established).

vi. To add another curve, such as the initial condition to the same plot, use
`replot 'sample1.out.0000' u 4:7`
These two curves should be very close to each other, as the last plot is exactly one period of the wave after the first plot. You can plot two or more curves together with a single command
`plot 'sample1.out.0000' u 4:7, 'sample1.out.0010' u 4:7`

vii. To see how the wave evolves in time over the course of the simulations, plot every other snapshot in time as follows: `plot 'sample1.out.0000' u 4:7`
`replot 'sample1.out.0002' u 4:7`
`replot 'sample1.out.0004' u 4:7`
`replot 'sample1.out.0006' u 4:7`
`replot 'sample1.out.0008' u 4:7`
`replot 'sample1.out.0010' u 4:7`
You can also label the axes using
`set xlabel "x"`
`set ylabel "u_x"`

viii. To output a plot to a Postscript file:
`set terminal postscript enhanced linewidth 2.0 set output "sample1.ps"` Then issue all of your plotting commands again. Note that the plot on screen will not update with these commands.
Finally, return the output to the screen by using
`set output`
`set terminal x11`

ix. You can use `ghostview` to view the resulting postscript using `gv sample1.ps &`

x. (Return) Please turn in only the figure from the `gnuplot` output of `sample1.in`