In-class Programming Homework, Nov. 3

Due date: Wednesday, Nov. 17 at midnight. You may work with up to one other person on the entirety of the assignment and turn in a joint program if you wish; in this case, each of you gets the same grade.

Introduction:

In this homework, you will parallelize a serial C, C++ or a Fortran code using OpenMP. The provided serial code solves the linearized shallow water equations in two dimension. After parallelizing the code, you will run it using 1, 2, 4, 6 and 8 threads to approximate the solution of the elevation h in the shallow equations at a specified time (see below) and use the provided "ncl" script (see below) to generate a 2D contour plot with your numerical solution. The files containing the procedures and the graphic script that you will use as a starting point are combined in "tar" files for_omp.tar (Fortran), c_omp.tar (C) and cc_omp.tar. (C++). You might practice with the three programming languages if you wish, but please select only one for the submission of your assignment.

If you are using a computer with unix or linux shell, you can transfer these tar files to Agave by using the following commands:

```
sftp username@agave.asu.edu
sftp> cd APM512
(or the name of the working directory you created for the class)
sftp> put for_omp.tar     (or c_omp.tar or cc_omp.tar)
sftp> quit
```

Login to Agave and extract the files, using the commands:

```
ssh username@agave.asu.edu
$ cd APM512 (or your working directory)
$ tar -xvf for_omp.tar     (or c_omp.tar or cc_omp.tar)
```

The following folder should be created be created:
Fortran: FOR_OMP: contains module_precision.f, module_bound.f module_flux.f, module_tendency.f, main.f and plot_h.ncl
C: C_OMP: contains bound.c, bound.h, flux.c, flux.h, tendency.c, tendency.h, main.c and plot_h.ncl

C++: CC_OMP: contains bound.cc, bound.h, flux.cc, flux.h, tendency.cc, tendency.h, main.cc and plot_h.ncl

To compile and run the code, use the following command lines:

```
$ interactive -n 10
```

- **Intel compiler:**

  ```
  $ module load intel/2018.x
  $ ifort -O3 -FR -qopenmp -o main module_precision.f module_bound.f
                module_flux.f module_tendency.f main.f (for Fortran)
  $ icc -O3 -std=c99 -qopenmp -o main bound.c flux.c tendency.c main.c
  (for C)
  $ icpc -O3 -qopenmp -o main bound.cc flux.cc tendency.cc main.cc
  (for C++)
  ```

Note: the code as it stands WILL compile and run serially without the flag -qopenmp
For parallel compilation and execution, you will need to modify the files main.f (.c or .cc) and module_tendency.f (tendency.c or tendency.cc) and compile the code with the flag -qopenmp. The places to modify are indicated in the program files.

After implementing the modifications, an executable file called **"main"** should be created. For parallel execution, type the following commands:

```
$ export OMP_NUM_THREADS=n (n=1, 2, 4, 6 or 8)
$ ./main
```

Assignment:

Consider the linearized shallow water equations in two dimension

$$\begin{aligned}
&\frac{\partial u}{\partial t} + g\frac{\partial h}{\partial x} = 0\\
&\frac{\partial v}{\partial t} + g\frac{\partial h}{\partial y} = 0\\
&\frac{\partial h}{\partial t} + H\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = Q(x,y,t)
\end{aligned}$$

Here, $h(x, y, t)$ is the perturbation of elevation, $u(x, y, t)$ and $v(x, y, t)$ are the perturbations of the $x-$ and $y-$velocities, $Q(x, y, t)$ is the forcing, t represents time, x is the position along the x-axis and y is the position along the y- axis. The parameters g and H represent the constant of gravity, $g = 9.81ms^{-2}$, and the elevation at rest, which is selected such that $c = \sqrt{gH} = 30ms^{-1}$.

The prescribed forcing Q has the form

$$Q = \frac{1}{\Delta t} Q_0 e^{-\frac{x^2+y^2}{2\Delta x^2}} \sin(2\pi\omega t)$$

where, $Q_0 = 10$, $\omega = 4 \times 10^{-3}$, Δt is the time step and Δx. Δt and Δx are defined below and in the program file main.f (main.c, main.cc).

The initial conditions are

$$h(x, y, 0) = 0; \quad u(x, y, 0) = 0; \; v(x, y, 0) = 0$$

The provided code (after modifications) integrates the shallow water equations in the domain defined by

$$-\frac{L}{2} \le x < \frac{L}{2}; \quad -\frac{L}{2} \le y < \frac{L}{2}$$

where $L = 60km$.

The domain uses a staggered grid with periodic boundary conditions

$$\psi(x + L, y, t) = \psi(x, t) \;\text{ and }\; \psi(x, y + L, t) = \psi(x, y, t)$$

where ψ represents h, u or v. The code uses a total number of $N_x \times N_y$ grid points,

where $N_x = N_y = M = 600$, with a grid spacing of $\Delta x = \Delta y = 100m$

The time integration uses the third-order Runge-Kutta (RK3) time stepping scheme; and the space derivatives are approximated by fourth-order finite differences.

The numerical solver integrates the equations from $t = 0$ to $t = n_t\Delta t$, where Δt is time step and $n_t = 1000$ is the total number of steps. The time step is determined from $\Delta t = r\Delta x/(c\sqrt{2})$, where $r = 0.4$ is the Courant number.

Algorithm for the solver:

- Begin the time step loop for $it = 1, 2, \cdots, n_t$

    1. Begin the RK3 substep loop for $ir = 1, 2, 3$

    (a) RK3 substep1: $ir = 1 \implies \Delta\tau = \Delta t/3, \ s = n, \quad s+1 = *$
    (b) RK3 substep2: $ir = 2 \implies \Delta\tau = \Delta t/2, \ s = *, \quad s+1 = **$
    (c) RK3 substep3: $ir = 3 \implies \Delta\tau = \Delta t, \ s = **, \quad s+1 = n+1$
    (d) Compute the tendency terms $F^s_{i,q}$, $G^s_{p,j}$ and $R^s_{p,q}$
    (e) Advance u, v and h for the substep $s+1$:
$$u^{s+1}_{i,q} = u^n_{i,q} - \Delta\tau F^s_{i,q}$$
$$v^{s+1}_{p,j} = v^n_{p,j} - \Delta\tau G^s_{p,j}$$
$$h^{s+1}_{p,q} = h^n_{p,q} - \Delta\tau G^s_{p,q}$$
    (f) Repeat steps a, b, c, d and e for the next substep $ir = ir + 1$

2. End the RK3 substep loop
3. Apply the forcing for h
$$h^{n+1}_{p,q} = h^{n+1}_{p,q} + Q_{p,q}$$
4. Overwrite u^n, v^n and v^n:
$$u^n_{i,q} = u^{n+1}_{i,q}$$
$$v^n_{p,j} = v^{n+1}_{p,j}$$
$$h^n_{p,q} = h^{n+1}_{p,q}$$
5. Repeat steps 1-5 for the next time step $it = it + 1$

- End the time step loop

1) Write directives or pragmas to parallelize the serial code provided. This code integrates the shallow water equations described above. The places where you will include your modifications are indicated in the program files. The files that need modifications are: main.f (.c, .cc), module_tendency.f, tendency.c (.h). Use the procedures provided in module_flux.f (flux.c, flux.cc) as a guide to parallelize your procedures. The routine that computes a load-balanced decomposition of the outer (y) loop is provided in module_bound.f (bound.c or .cc). Solve the equations using the parameters $(c, H, g, L, \cdots)$ described above for a total number of time steps $n_t = 1000$. A binary file named "shallow.bin" should be produced after successful execution of ./main. This file should archive two dimensional fields of the simulated elevation for each 100 time steps. You plot should include the archived field at $n_t = 1000$.

2) Use the NCAR Command Language (NCL) (https://www.ncl.ucar.edu) graphic software, which is available in AGAVE, to generate a contour plot for the two dimensional field of elevation simulated at $n_t = 1000$. Use the following ranges for the axes: x-axis [-30km, 30km] and y-axis [-30km, 30km].
A graphic script called plot_h.ncl is provided for you. To run the script use the following command lines:

- Generating the plot using NCL:

  ```
  $ module load ncl/6.3.0
  $ ncl plot.ncl
  ```

  A pdf file called shallow.pdf should be generated

Grading rubric:

- Upload the following files to Canvas for grading:
  - A tar file hw5.tar containing your program files.
  - A pdf plot showing the elevation field simulated with 8 threads at $n_t = 1000$ in pdf format. Please don't upload the binary file to canvas.
- This assignment is worth 20 points, as follows:
  - (2 points) The program structure contains the files and the functions mentioned above with the appropriate naming
  - (3 points) Your program compiles without warnings or errors with the flag options specified above.
  - (10 points) When compiled and executed, your program produces accurate approximation for h and speeds up by a factor of at least 6 on 8 threads.
  - (5 points) The program is adequately (but not excessively) commented.