1 Parallel User's Manual

1.1 The program

The parallel version of this shallow water circulation model is contained in four MATLAB functions. The 'main' function is contained in `shallowPar.m`, which calls two of the other functions; `initialize.m` sets up the initial square domain and `spOneStep.m` performs one time step of calculations for each individual processor. The last matlab function is `spPlot` which gathers data from a sucessful run other program and creates an animated plot.

In order to run this program, the user must have access to MATLAB version R2007b or later with the Parallel Computing Toolbox version 3.2 or later as well as at least four MATLAB parallel labs–obviously performance is much higher if each of these labs is running on a unique processor–and a MATLAB client. This program supports any number of labs, so long as it is a power of two; e.g. four, eight, sixteen, etc.

The main program must be run in parallel mode. A simple way to start this is to use the command

```
pmode start n
```

at the command line, where `n` is the number of labs available to the program. Note that the user must specify the number of labs available to the program in the initialization file; instructions for doing this are provided in the next section. After parallel mode is initialized, on the parallel command line, the user can run

```
shallowPar
```

to begin the program. In order to run this program, the user must navigate each of the labs to the directory in which the function files are stored. The starting conditions are set in the initialization file; again, instructions for doing this are provided in the next section.

Each parallel process runs a unique copy of the function `shallowPar.m`. These processes interact with each other to pass data and proceed with calculations at their own pace, which can be different depending on the relative speed of each processor. Each process will count out its progress in terms of number of times plotting data is stored to the lab's workspace. Plotting data is stored every 1/10 of a time step, with the total runtime defined by the user in the initialization file. Thus, for a program with a user-defined runtime of 10, each parallel process will count upwards to 100.

When all parallel processes have completed execution, the plotting function `spPlot.m` may be run in the client. Type the command

```
spPlot
```

in the client's command line and this function will gather data from each lab's workspace and create an animated plot from the plotting data that was stored at each counting iteration.

1.2 Setting user parameters

User defined parameters are set in the file `initialize.m`, which can be edited with any text editor. Please only modify data in the section that is labeled as user modifiable.

The most important user-defined parameter is the number of labs available to the program. This is defined by the variable `P` and is the first piece of data the user can set. If this number is set higher than the actual number of labs available, the program will crash. In addition, if this number is not set to be a power of 2, or is less than 4, the program will not function as intended, and may not run at all.

The next pieces of data set the initial conditions for the program. There is a brief description of each variable in the file.

`N` defines the number of grid lines in each direction. There are 2^N gridlines in each direction, which divides the domain into $2^N + 1 \times 2^N + 1$ elements. Suggested values are from 5 to 8. Values larger than 8 will run very slowly on all but the most powerful clusters.

`b` is the coefficient of rotation. This provides a rotational force analogous to the Coriolis effect to the initial fluid domain. Suggested values are on the order of 1.

`Tmax` is the runtime of the program. The program will plot 10 times per time unit of runtime. This value has an upper limit that depends on the memory available to each lab. If the runtime is too long, plot data will fill the lab's memory and slow the processors nearly to a halt. It is recommended to start at a lower value, such as 10, to ensure that there is enough memory to run.

`a` is the bottom friction coefficient. A value of 0 represents no friction, while 1 represents enough friction to immediately stop the motion of the bottom layer of the fluid.

`xc` and `xc` are the initial x- and y- coordinates of the fluid's center of mass. The bottom surfaces is a symetrical paraboloid with its lowest point at the origin. The starting coordinates should not be at the origin or there will be no "sloshing." Suggested values are between -1 and 1.

`mu` is the eddy viscocity. As discussed in the paper, this is a user-defined quantity. Suggested values are 0.05 or 0.1.

`a1` is the ratio of dt to dx. Suggested values are 0.05, 0.02 or 0.01. The smaller the value, the finer the time grid, since the size of dx is set by the grid size defined by `N`. For values greater than 0.05, there is a risk of negative element areas, where the shape of one element will become so distorted that it approaches zero, or becomes negative. This will cause the program to crash. If these crashes occur, choose a smaller value for `a1`.

1.3 Other program options

There is an alternate version of the function that does calculations included with the aforementioned files; it is titled `spOneStep2.m`. This version uses h^2 instead of h when calculating the terms s1, s2, T1, and T2. See equations 6.31 and 6.60 in the paper for further details. In order to use this version of the program, first back up the file `spOneStep.m` under a different file name. Then relabel the file `spOneStep2.m` to `spOneStep.m` and run the program as normal given the instructions above.