High Performance Computing Coursework

This project implements a parallel numerical code to solve the vorticity-streamfunction formulation of the incompressible Naiver-Stokes equation in 2D using the finite difference method. In particular, it implements a solver for the lid-driven cavity problem. The code is written in C++ and is parallelised through a combination of libraries including MPI and Scalapack.

Getting Started

These instructions are targetted at helping you get a copy of the code up and running.

Prerequisites

The running of the code requires the following to be preinstalled on the system. If the code is being run on either Hurricane or Spitfire, these libraries should already be present.

- Boost - in particular the program_options library to obtain program options from the command line
- MPI - message passing interface for parallel programming
- BLAS - linear algebra package for performing basic vector and matrix multiplication
- BLACS - linear algebra oriented message passing interface
- LAPACK - linear algebra package for solving systems of simultaneous linear equations in serial
- ScaLAPACK - linear algebra package for solving systems of simultaneous linear equations in parallel

Compiling

To compile the program, simply change into the base directory containing the Makefile and run the following command

```
make
```

The make file will compile the program and produce the executable **Solve**.

Running the tests

2 test cases, one serial and one parallel, have been included in the make file to test the validity of the output. This is done with parameters of:

- Lx = Ly = 1.0
- Nx = Ny = 5
- dt = 0.0001
- T = 1.0
- Re = 100

Serial test case

For the serial case, the code is run on a single process (--np 1) with Px = Py = 1. In order to run the serial test case, simply run the following command

```
make testSerial
```

Parallel test case

For the parallel case, the code is run on 4 processes (--np 4) with Px = Py = 2. In order to run the parallel test case, simply run the following command

```
make test
```

The output data of these tests can be found in the **Data/** folder.

Running for report output

An additional script named **./RunSolver** was implemented to run the solver with the given parameters required for the HPC Coursework report plots. To run the script, run the following command

```
./RunSolver.sh
```

This script will first compile the solver via the `make` command, then run the solver for the following parameters

- Lx = 1.0, Ly = 1.0, Nx = 161, Ny = 161, dt = 0.0005, T = 1.0, Re = 100
- Lx = 1.0, Ly = 1.0, Nx = 161, Ny = 161, dt = 0.0005, T = 1.0, Re = 400
- Lx = 1.0, Ly = 1.0, Nx = 161, Ny = 161, dt = 0.0005, T = 5.0, Re = 1000
- Lx = 1.0, Ly = 1.0, Nx = 161, Ny = 161, dt = 0.0005, T = 10.0, Re = 3200
- Lx = 1.0, Ly = 2.0, Nx = 161, Ny = 161, dt = 0.0005, T = 1.0, Re = 100
- Lx = 2.0, Ly = 1.0, Nx = 161, Ny = 161, dt = 0.0005, T = 1.0, Re = 100

All of these are run with 9 processes (--np 9) and Px = Py = 3. The data is saved in the **Data/** folder in the format `<Lx>_<Ly>_<Nx>_<Ny>_<Re>_data.txt`. Following this, the MATLAB script **GeneratePlots.m** is run, which generates the required plots and table data for the report. These plots are saved in the **Images/** folder.

Cleaning the folder

To clean the folder, i.e. remove the generated files, the following commands can be run.

Remove generated object file (*.o)

```
make clean
```

Remove generated plots (*.png)

```
make cleanImages
```

Remove generated data file (*.txt)

```
make cleanData
```

Remove all generated files

```
make cleanAll
```

Usage

Once the solver has been compiled, it can be used via the format

```
mpiexec --np <no. of procs> ./Solve [options] [args]
```

```
options:
--help        prints out the help message
--Lx <double> specifies length of domain in the x-direction
--Ly <double> specifies length of domain in the y-direction
--Nx <int>    specifies number of grid points in the x-direction
--Ny <int>    specifies number of grid points in the y-direction
--Px <int>    specifies number of partitions in the x-direction
--Py <int>    specifies number of partitions in the y-direction
--dt <double> specifies the timestep
--T <double>  specifies the final time
--Re <double> specifies the Reynolds number
```

Similarly, the data generated from running this will be saved in the **Data/** folder in the format `<Lx>_<Ly>_<Nx>_<Ny>_<Re>_data.txt` .

Data verification

A seperate MATLAB implementation of the lid-driven cavity solver is included in the distribution. This file is named **LidDrivenCavitySolver.m** and can be used to test the output against that produced by the C++ implementation.

Authors

- **Sean Chai** - *jsc4017* - jsc4017@ic.ac.uk