Setup of our DNS code

Tamer A. Zaki
Mechanical Engineering, Imperial College, London, SW7 2AZ, UK

(Received 11 August 2009)

This document provides a description of the main components of the DNS solver, and how to setup a sample calculation. The details are dependent on the platform, and the particular architecture. This guide is therefore intended to simply provide an overall view of the process, and the details must be tailored for the particular case of interest.

---

1. Introduction

You should have received a tar-ball of the code. When you untar this directory, you will recover the directory *DNS_Code_Blocks*, which contains:

- *DNS_Code_Blocks/lib*
- *DNS_Code_Blocks/Gen_Cart_Grid*
- *DNS_Code_Blocks/setup*
- *DNS_Code_Blocks/main_code*
- *DNS_Code_Blocks/Post_processing*

Each of the above items is required, and is described below.

Before you start playing with the code, remember that we're going to be using FORTRAN, and also MPI. Therefore, you will need to load some modules. I suggest you include the following modules in your *.bash_profile* file:

```
module load intel-suite
module load fftw/2.1.5-double
module load mpi
```

Don't forget to source your *.bash_profile* file before you proceed.

```
>>  source .bash_profile
```

1.1. *DNS_Code_Blocks/lib*

The sub-directory *DNS_Code_Blocks/lib* contains a set of routines that are commonly used by many applications, and therefore they are grouped in a library. These routines include items such as tri-diagonal solvers, penta-diagonal solvers, Fourier transforms,...etc.

Note: The files in this sub-directory must be compiled first, before any other codes since most of the other applications depend on lib. To compile the library, type the following command at the command prompt:

```
>> make
```

This will create a directory called *lib* in your home directory *($HOME/lib/)* , with the object files from compiling the above library. When you compile other applications that require linking to library, you will be linking to the object files in *$HOME/lib/*.o*.

1.2. *DNS_Code_Blocks/Gen_Cart_Grid*

The sub-directory *DNS_Code_Blocks/Gen_Cart_Grid* contains a set of routines that can create Cartesian grids. The grid is two-dimensional $x(i,j)$ and $y(i,j)$. The $x-$coordinate is uniform, and the $y-$coordinate can be uniform, stretched (for instance according to a hyperbolic tangent profile),..etc.

To compile the grid-generator, type the following command at the command prompt:

```
>>  ifort -r8 -o Gen_grid.data.exe main.f90 mesh_tanh_stretch.f90
```

The above command compiles the **.f90* files, and generates an executable, which is named *Gen_grid.data.exe*.

In order to generate a grid, you must run the executable, and also provide an input file. The input file is called *input.file*, and a sample is provided. *Note:* The format of the input file must be kept unchanged for the code to work properly. Now, to run the code, type the following at the command prompt,

```
>> ./Gen_grid.data.exe
```

This will create a *grid.data* file, which has the (x,y) coordinates of the grid.

1.3. *DNS_Code_Blocks/setup*

The sub-directory *DNS_Code_Blocks/setup* contains a set of routines that creates the initial field for the simulation, and also an archive. The notion of an *archive* file is important, and is discussed further in §2.

In order to compile the setup, type the following command at the command prompt:

```
>> make simple
```

The choice of *simple* is because the *makefile* can be used for many different initial fields. It was used for boundary layers, channels, jets,....etc. In the future, you can change it for your own purposes.

Once the code is compiled, it creates *a.out* file, which is the executable. You will also need an input file. An example input file is provided, and is named *input.setup*. *Note:* The format of the input file must be kept unchanged for the code to work properly. Now, to run the code, you must copy *grid.data* from the *../Gen_Cart_Grid/* directory into the current *setup* directory. Now you can type the following at the command prompt,

```
>> ./a.out
```

Running the code should create a number of files: contains:

- *ucvcwc.dble.000000*: Initial Cartesian velocity field
- *uuvvww.dble.000000*: Initial velocity fluxes
- *conold.dble.000000*: Initial convective terms
- *pressure_ph.000000*: Initial pressure in Fourier space, $\hat{p}$
- *archive*: A binary archive file
- *report*: An ASCI summary of *archive*. It has parameters of the run.

These files, as well as *grid.data*, must be copied to the location where you wish to carry out your simulations. *Note:* You should ***never*** run the code in your home directory. This will create large files, which will most certainly be beyond the storage quota for your $HOME. These circumstances can be avoided by submitting jobs from your $WORK - details of which can be found in §2 of this document.

1.4. *DNS_Code_Blocks/main_code*

The sub-directory *DNS_Code_Blocks/main_code* contains the main code. *Note:* Before you compile the code, you must edit the file *param.inc*, in order to make sure the data sizes match the grid and initial fields you have generated. In order to compile the code, you will need the usual command,

```
>> make
```

Now an executable, *a.out*, has been generated. We're almost there - only a few more files are needed before you can run the simulations: contains:

- *data*: ASCII file that contains the time-step. When starting from the initial field, *data* has only the value 0 (zero).
- *input*: Parameters for the run – a sample file is provided.

You are almost ready to run your first DNS. Before you do so, you will need 1 last file, the script that will tell the cluster what resources you need, the duration of the run,...etc. This file is provided in the tar-ball, in the parent directory *DNS_Code_Blocks*. In the next section, we will discuss how to submit a job to a parallel cluster.

1.5. *DNS_Code_Blocks/Post_processing*

The sub-directory *DNS_Code_Blocks/Post_processing* contains a simple visualization routine in MATLAB. You should instead try to use the visualization routines provided for Tecplot and Ensight. These are more powerful visualization packages, which you will use throughout your research.

2. The first simulation

In order to setup your first computation, you should first go through the steps outlined above in §1. These steps guide you through compiling the code, setting an initial field,...etc.

Now you are ready for your first job. However, before you setup the first computation, a word regarding directory structure. Your DNS code should reside in your $HOME directory. All simulations data should be stored in the $WORK directory.

For every new simulation, we create a new sub-directory in $WORK, and it will include a copy of the actual code that was used and the simulation results. This way, you have saved with the simulation the code that was used to generate a particular data-set, even if the version in your $HOME changes due to updates and developments.

First, create a folder in the $WORK directory where you will run your job:

```
>>  mkdir $WORK/my_first_dns
>>  cd $WORK/my_first_dns
```

We will make two sub-directories, one for the code and the second for the data:

```
>>  mkdir source_code
>>  mkdir data
```

Copy the source code from $HOME into the *source_code* directory:

```
>>  cp -r ~/DNS_Code_Blocks/main_code            source_code/.
>>  cp -r ~/DNS_Code_Blocks/Gen_Cart_Grid        source_code/.
>>  cp -r ~/DNS_Code_Blocks/setup                source_code/.
```

You could recompile the above codes in their $WORK location, just as we have in §1, and use them to generate the grid and initial field. Since we have already carried out this step in $HOME, we have already all the needed components. All we need to do is simply copy the necessary files from to the *data* directory, where we will carry out the actual simulation.

```
>>  cp  source_code/Gen_Cart_Grid/grid.data     data/.
>>  cp  source_code/setup/*000000               data/.
>>  cp  source_code/setup/archive               data/.
>>  cp  source_code/setup/report                data/.
>>  cp  source_code/main_code/a.out             data/.
```

As mentioned in §1, we need an input file, a data file, and a run-script:

```
>>  cp ~/DNS_Code_Blocks/input                  data/.
>>  cp ~/DNS_Code_Blocks/run_script.sh          data/.
```

Note that the files *archive* and *report* are overwritten after the job has finished running. We want to keep an original copy of *archive* and *report* and hence a renamed copy of these files (in the current directory) is made by executing the following command:

```
>> cp archive archive.000000
>> cp report   report.000000
```

Create a file called *data*, which only contains the number 0 (zero).

```
>>  vi data
    i TAB 0 ESC
    :wq
```

Now you can submit your first job:

```
>>  qsub run_script.sh
```

To check the status of the job (whether it's still running or it's done) use:

```
>>  qstat
```

Once the job has finished running you will get the following files:

- *ucvcwc.dble.002000*: Velocity field in Cartesian coordinates at final time.
- *uuvvww.dble.002000*: Velocity fluxes at final time.
- *conold.dble.002000*: Convective terms at final time.
- *pressure_ph.002000*: Pressure in Fourier space, $\hat{p}$ at final time.
- *pres_p.dble.002000*: Pressure in physical space at final time.
- *archive*: Binary archive file (overwritten).
- *report*: ASCI summary of *archive* (overwritten).

You can the visualize the velocity fields with the postprocessing routines.
A simple MATLAB version is given in $HOME/DNS_Code_Blocks/Post_processing. However, you would be substantially better off using the visualization packages which you have recently learned.