A Quick Guide to PCUI

Gonçalo Gil
Stanford University, CA

18 September 2013*

*First drafted on 6 September 2013

Contents

Chapter 1

PCUI structure

This chapter deals with the structure of PCUI. First, all files are listed with their summaries, then the main file *ns.f* and initialization file *init.f* are briefly described. Next, we categorize and list the most important variables and provide a summary for each one.

1.1 File summary

1.1.1 Runtime files

Nr.	Filename	Summary
01	Makefile	Makefile to organize code compilation
02	cav	Resulting executable from running *make*

1.1.2 Fortran files

Here the first table shows the files that contain the subroutines called by the main subroutine (*ns.f*). The second table contains files with auxiliary subroutines.

1.1.3 Include files

1.1.4 Output files

The # symbol is a number that the file extension is given depending on the processor number that saved that file.

1.2 File Description

1.2.1 ns.f

This is the main file that runs the code. It does the following:

- Initialize MPI.

Table 1.1: Main files

Nr.	**Filename**	**Summary**
01	ns.f	Main file: run the code
02	mp.f	Initialize MPI
03	init.f	Initialize scalar and vector fields (ρ, $\mathbf{u}$)
04	cavity.f	Sets up domain size and grid for Example 2
05	isw.f	Sets up domain size and grid for Example 3
06	io.f	Deals with input and output
07	eddy.f	LES model
08	scal.f	Scalar solver
09	pred.f	Predictor step
10	pres.f	Pressure initialization
11	corr.f	Corrector step (velocity boundary conditions are set here)
12	grid_init.f	Initializes grid and outputs to file

Table 1.2: Auxiliary files

Nr.	**Filename**	**Summary**
11	trid.f	Tridiagonal solver
12	smad.f	ADI Smoothing
13	resd.f	Residual
14	fctr.f	Fine to coarse and coarse to fine
15	exch.f	Exchange boundary values
16	conv.f	Explicit convective terms

Table 1.3: Include files

Nr.	**Filename**	**Summary**
01	cavity.inc	Define domain variables
02	eddy.inc	Define LES model variables
03	metric.inc	Define metrics
04	mpi.inc	Define MPI topology variables
05	ns.inc	Define field variables
06	para.inc	Define auxiliary variables and parameters
07	size.inc	Define grid size and number of processors

- Initialize all variables.
- Initialize MPI topology.
- Initialize domain and set up grid.
- Iterate over the number of time steps.
  - Save all relevant data to binary files.
  - Call LES model.

Table 1.4: Output files

Nr.	**Filename**	**Summary**
01	xyz.###	Binary files containing grid
02	output_UVW.###	Binary files containing velocity vector field
03	output_S.###	Binary files containing density scalar field
04	qoutput	File that outputs run progress to assist when running in Bobstar

  - Compute the right hand side of the scalar field and update it.
  - Call predictor step, update intermediate velocity u*.
  - Call multigrid pressure solver.
  - Call corrector step, apply velocity boundary conditions.
  - Call scalar field solver, apply scalar field boundary conditions.
  - Call CFL number checker.

- Save last step.
- Output statistics to screen.

1.2.2 init.f

This is where the velocity and scalar fields are defined. This version of PCUI is set up to run the classic test case of a three-dimensional lid driven cavity, you can find details in Cui [1999]. The subroutine first initializes the lid velocity variable *u_lid*. Next, it initializes the scalar and vector fields including boundary conditions.

1.3 Variable summary

Only the most relevant variables are shown here.

Table 1.5: Parameters

Nr.	**Filename**	**Summary**
01	dtime	Time step
02	case	Chooses different lid velocities depending on value of *case* (check *init.f*)
03	newrun	Used to start a new run or to start off where last run ended
04	periodic	Chooses if periodic boundary conditions are required
05	iscalar	Chooses if scalar solver is executed
06	mg_level	Multigrid level
07	nstep	Number of steps
08	nsave	Step interval to save results
09	vis	Molecular viscosity
10	ak	Eddy viscosity

Table 1.6: Field, domain and grid variables

Nr.	**Filename**	**Summary**
01	phi	Density scalar field
02	phi_init	Initial density scalar field
03	p	Pressure scalar field
04	u	Velocity vector field
05	u_lid	Lid velocity
06	xp	Grid definition
07	bx	Domain length
08	by	Domain height (note that y is the vertical coordinate and points upwards)
09	bz	Domain width
10	stretchx	Stretched grid in x-direction
11	stretchy	Stretched grid in y-direction
12	stretchz	Stretched grid in z-direction

Chapter 2

PCUI Examples

This chapter gets you started with a coding example whereby you will expand the functionality of PCUI. Then we go through the typical steps performed to set up two different kinds of runs.

2.1 Example 1: Periodic Boundary Conditions

This example may be performed in the FORTRAN source code from example 3, namely in folder '*./examples/pcui_internal_seiche/*'. Since, by default, periodic boundary conditions have only been implemented in the x-direction, the idea here is to get right into the code by modifying it to incorporate the periodic condition in the z-direction as well. Here, we briefly describe the required steps noting that, since it has been done for the x-direction, it is merely a question of performing a few copy and pastes and a few minor changes. To get started use the *grep* command on a terminal:

```
$ grep periodic *.f
```

The result is:

```
[1]  io.f:    periodic = 0
[2]  io.f:    call MPI_BCAST(periodic,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
[3]  pred.f:  if ( periodic .eq. 1 ) then
[4]  scal.f:  if ( periodic .eq. 1 ) then
[5]  smad.f:  if ( periodic .eq. 1 ) then
[6]  mp.f:    if ( periodic .eq. 1 ) periods(3) = .true.
```

Line [6] shows the file that needs to be modified in terms of code initialization. Lines [3], [4] and [5] show the files that need to be modified in terms of the code operation. Currently, the variable *periodic* defines whether the x-direction should have periodic boundary conditions and is specified in the *parameter* subroutine in *io.f*. Thus, to set up periodic boundary conditions in the z-direction you need to do the following:

1. Modify file *mp.f* to accept periodic boundary conditions in the z-direction.

2. Modify files *pred.f*, *scal.f* and *smad.f* to accept periodic boundary conditions in the z-direction.

2.2 Example 2: Lid driven cavity flow

In this example all the parameters are predefined, but, for completeness, here is a summary of the steps one would follow to successfully run the cavity example:

1. Navigate to example folder: '*./examples/pcui_cavity/*'.
2. Open file *io.f* and in subroutine *parameter* define the problem parameters such as time step, number of time steps, viscosity, etc.
3. Open file *size.inc* and define the grid size and processor number per direction.
4. Open file *cavity.f* and in subroutine *grid* define the domain properties.
5. Open file *init.f* and define the lid velocity, density and velocity fields in subroutine *initial*.
6. Open file *Makefile* to define the compiler and compiler options
7. Run command *make* in terminal to compile
8. Run command *mpirun -np x ./cav* to execute code

where x is given by the processor number $x = px * py * pz$ (defined in *size.inc*). If something is not working properly for some odd reason, run *make clean* and then proceed with steps 6 and 7. Check Cui [1999] for more details.

2.3 Example 3: Internal seiche

In the previous example the velocity field was directly defined in PCUI, alternatively, this example uses Matlab to initialize the velocity and density fields. This is useful in case you have a non analytic input to your simulation, such as a flow field that might result from a numerical solver. Therefore, the only difference from the previous example is that step 5 is substituted by running a Matlab script. Before running the main simulation, it is required that the grid be generated by PCUI before running the Matlab script to initialize the density and velocity fields. Therefore, in this example PCUI will be run twice. The first run will only generate the grid and write it to the file *xyz* so that it can be read by Matlab. The second run will perform the actual simulation. The following method assumes that you have an understanding of how to run PCUI (i.e. you can successfully complete Example 2).

1. Navigate to folder '*./examples/pcui_internal_seiche/*' and if desired change the problem properties as shown in the previous example. To use the default values skip to the next step.

2. To compile the executable that will save and generate the grid, run make as follows '*make grid*'. This operation creates an executable named *grid*.

3. Run PCUI with *'mpirun -np x ./grid'* to generate the grid and save it in files '*xyz.###*', where x is the number of processors defined in file *size.inc*.

4. Navigate to folder '*./examples/pcui_internal_seiche/mfiles/*' and run the matlab script *initialize_pcui.m*. This writes FORTRAN binary files for each processor, depending on what is specified in file *size.inc*, with the density and velocity fields which are read by PCUI (check file *init.f*).

5. Run '*make clean*' and '*make cleano*' to clean up the grid generation and any associated files. Then compile with '*make*'. Run PCUI with '*'mpirun -np x ./isw'*' to run the simulation.

Chapter 3

Matlab scripts

Matlab is used for initializing PCUI and post-processing the results. The Matlab files described in this chapter are found in folders: '*./examples/pcui_cavity/mfiles/*' corresponding to example 2 and '*./examples/pcui_internal_seiche/mfiles/*' corresponding to example 3. Example 3 contains more files than example 2 because some files are relative to initializing PCUI using Matlab scripts.

3.1 File summary

3.1.1 Main files

Nr.	**Filename**	**Summary**
01	post_process_pcui.m	Reads PCUI output and plots it
02	initialize_pcui	Writes density and velocity fields to PCUI

3.1.2 Auxiliary files

These files are required by the main files.

Nr.	**Filename**	**Summary**
01	getdata.m	Reads chunks of binary FORTRAN files
02	putdata.m	Writes chunks of binary FORTRAN files
03	read_binary_file_pcui.m	Read PCUI 3- or 4-dimensional arrays
04	write_binary_file_pcui.m	Write PCUI 3- or 4-dimensional arrays
05	variable_value_pcui.m	Searches through PCUI files for variable values
06	assemble_array.m	Converts from single Matlab arrays to PCUI readable files

3.2 Example 2: Lid driven cavity flow plots

Here we show results from the lid driven cavity example which are generated by the Matlab script file *post_process_pcui.m*. All components of velocity are shown. The grid is stretched near the boundaries in the x- and y-directions.

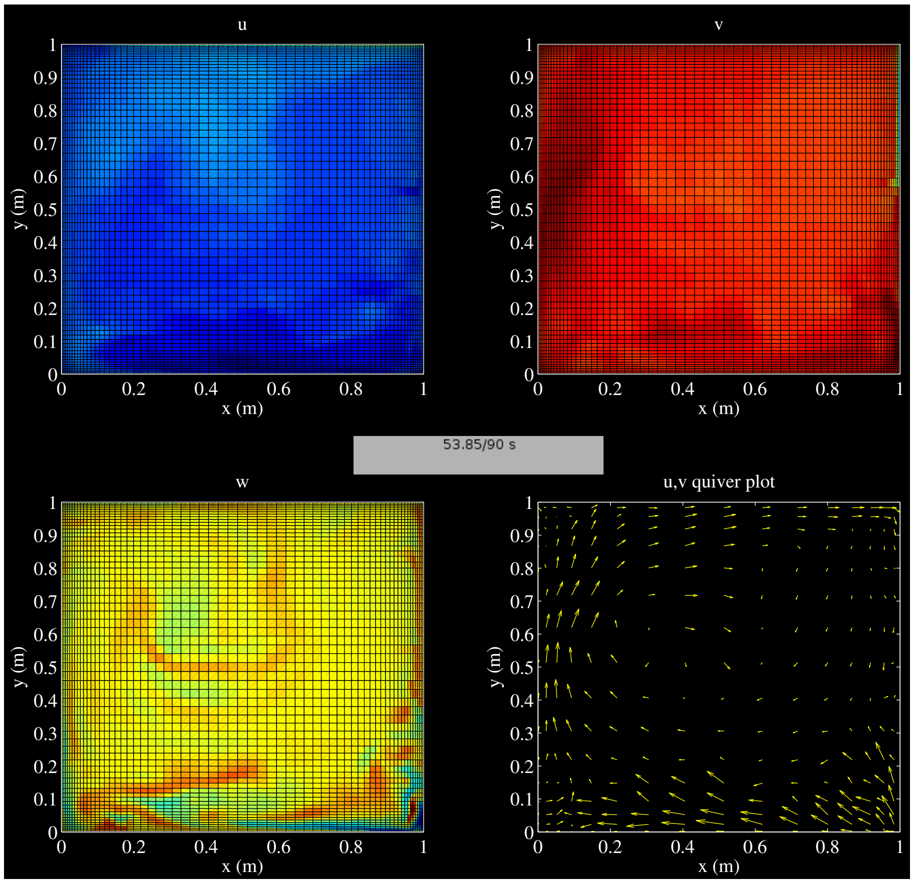

Figure 3.1: Velocity field for the lid driven cavity flow.

3.3 Example 3: Internal seiche plots

Here we show results from the internal seiche example which are generated by the Matlab script file *post_process_pcui.m*. All components of velocity are shown, as well as the density field.

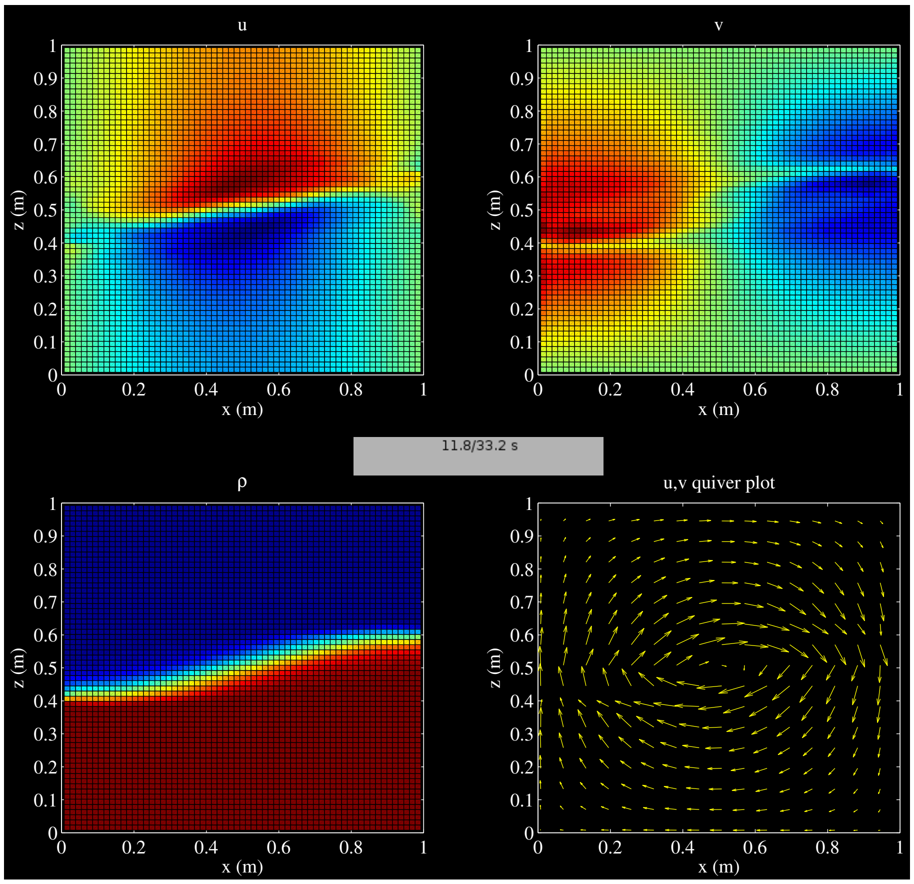

Figure 3.2: Velocity and density fields for an internal seiche.

Bibliography

A Cui. *On the parallel computation of turbulent rotating stratifed flows.* PhD thesis, Stanford University, 1999.