A Quick Guide to PCUI

Gonçalo Gil
Stanford University, CA

10 September 2013*

*First drafted on 6 September 2013

Contents

Contents **i**

1 PCUI structure **1**
1.1 File summary 1
1.1.1 Runtime files 1
1.1.2 Fortran files 1
1.1.3 Include files 2
1.1.4 Output files 2
1.2 File Description 2
1.2.1 ns.f 2
1.2.2 init.f 3
1.3 Variable summary 3
1.3.1 Parameters 3
1.3.2 Field, domain and grid variables 4

2 PCUI Examples **5**
2.1 Example 1: Periodic Boundary Conditions 5
2.2 Example 2: Run the cavity test case on 4 processors 6

Bibliography **7**

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.

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
05	io.f	Deals with output
06	eddy.f	LES model
07	scal.f	Scalar solver
08	pred.f	Predictor step
09	pres.f	Pressure solver
10	corr.f	Corrector step

Nr.	Filename	Summary
11	trid.f	Tridiagonal solver
12	smad.f	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

1.1.3 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

1.1.4 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

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.
- 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.
  - 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 driven lid 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.

1.3.1 Parameters

Nr.	Filename	Summary
01	dtime	Time step
02	case	Chooses different lid velocities depending on value of *case* (check *init.f*)
03	newrun	Initialize scalar and vector fields
04	periodic	Sets up domain size and grid
05	iscalar	Deals with output
06	ieddy.f	LES model
07	mg_level	Scalar solver
08	nstep	Predictor step
09	nsave	Pressure solver
10	vis	Molecular viscosity
11	ak	Eddy viscosity

1.3.2 Field, domain and grid variables

Nr.	Filename	Summary
01	phi	Perturbation 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 vector
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 in setting up a run.

2.1 Example 1: Periodic Boundary Conditions

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. I will 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. The variable *periodic* is specified in the *parameter* subroutine in *io.f*. If it is equal to 1 it currently sets up the x-direction to be periodic, thus, to set up periodic boundary conditions in the z-direction:

- File *mp.f* requires modification to set up z-direction.
- Files *pred.f*, *scal.f* and *smad.f* require modification to set up z-direction.

2.2 Example 2: Run the cavity test case on 4 processors

For this example, everything is predefined. But for completeness, this is more or less the steps one would perform to set everything up and run:

1. Set up run properties such as time step, number of time steps, viscosity, etc in *parameter* subroutine in file *io.f*
2. Define the grid size and processor number per direction in *size.inc*
3. Define the domain and grid properties in subroutine *grid* in file *cavity.f*
4. Define the lid velocity, density and velocity fields in subroutine *initial* in file *init.f*
5. Define compiler and compiler option in file *Makefile*
6. Run command *make* in terminal to compile
7. 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.

Bibliography

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