PDAC User's Guide

Version 2.0

(Restricted Draft: Do Not Release or Cite)

T. Esposti Ongaro, C. Cavazzoni and A. Neri

January 2004

Istituto Nazionale di Geofisica e Vulcanologia
Pisa - Italy

Description

This *User's Guide* describes how to run and use the various features of the Pyroclastic Dispersal Analysis Code PDAC. This guide includes a description of the capabilities of the program, how to use these capabilities, the necessary input files and formats, and how to run the program on both uniprocessor and parallel machines.

PDAC Version 2.0
(Restricted Draft: Do Not Release or Cite)

Authors: T. Esposti Ongaro, C. Cavazzoni and A. Neri

Istituto Nazionale di Geofisica e Vulcanologia

PDAC (Pyroclastic Dispersal Analysis Code) Non-Exclusive, Non-Commercial Use License

Introduction

INGV, in cooperation wtih CNR and CINECA, has developed the Pyroclastic Dispersal Analysis Code PDAC to the purpose of scientific research. PDAC is then available free of charge for non-commercial use by individuals, academic or research institutions, upon completion and submission of the online registration form available from the PDAC web site http://www.pi.ingv.it/PDAC.

Commercial use of the PDAC software, or derivative works based thereon, REQUIRES A COMMERCIAL LICENSE. Commercial use includes: (1) integration of all or part of the Software into a product for sale, lease or license by or on behalf of Licensee to third parties, or (2) distribution of the Software to third parties that need it to commercialize product sold or licensed by or on behalf of Licensee. INGV will negotiate commercial-use licenses for PDAC upon request. These requests can be directed to pdac@pi.ingv.it

Registration

Individuals may register in their own name or with their institutional or corporate affiliations. Registration information must include name, title, and e-mail of a person with signature authority to authorize and commit the individuals, academic or research institution, or corporation as necessary to the terms and conditions of the license agreement.

All parts of the information must be understood and agreed to as part of completing the form. Completion of the form is required before software access is granted. Pay particular attention to the authorized requester requirements above, and be sure that the form submission is authorized by the duly responsible person.

Registration will be administered by the PDAC development team.

PDAC: Pyroclastic Dispersal Analysis Code LICENSE AGREEMENT

Contact Information

The best contact path for licensing issues is by e-mail to pdac@pi.ingv.it or send correspondence to:

PDAC Development Team
Istituto Nazionale di Geofisica e Vulcanologia
Centro di modellistica fisica e pericolosità dei processi vulcanici
Sezione Sismologia e tettonofisica
32, Via della Faggiola
56126, Pisa - Italy
FAX: +39-0508311942

Contents

List of Figures

1 Introduction

PDAC is a numerical code that has been developed to solve the multiphase flow equations for the simulation of pyroclastic atmospheric dispersal dynamics. It is specifically designed for high performance computing on UNIX platforms. This document describes how to use PDAC, its features, and the platforms on which it runs. The document is divided into six sections:

Section 1 gives an overview of PDAC.

Section 2 describes PDAC file formats.

Section 3 lists basic simulation options.

Section 4 provides sample configuration files.

Section 5 gives details on running PDAC.

Section 6 gives details on installing PDAC.

We have tried to make this document complete and easy to understand as well as PDAC itself easy to install and run. We welcome any suggestion for improving the documentation and code itself at pdac@pi.ingv.it

1.1 The PDAC main features

The PDAC code solves the multiphase flow transport equations for density, momentum and energy of a gas-pyroclast mixture in a steady standard atmosphere. The gas phase can be composed of several chemical components leaving the crater - such as water vapor, carbon dioxide, etc. - and atmospheric air (considered as a single chemical component). The pyroclasts are described by N phases of solid particles, each one characterized by a diameter, density, specific heat, thermal conductivity, and viscosity, and considered representative of a granulometric class commonly present in the eruptive mixture. Momentum and energy exchange between the gas and the different particulate phases are expressed through semi-empirical correlation equations.

PDAC has several important features that make it a useful tool for the simulation of volcanic columns and pyroclastic flows:

- **Robust over a wide range of flow regimes**
  - from weak to strong gas-particle coupling;
  - from laminar to turbulent regimes;
  - from quasi-isothermal to highly buoyant flows;
  - from subsonic (incompressible) to supersonic (compressible) regimes.
- **Unlimited number of particles classes**
  The number of particle classes used to approximate an observed distribution of grain sizes is limited only by the computational resources available. Up to 6 particulate phases have been used successfully. The size and density of particles is nevertheless constrained by the multiphase flow assumptions, that limit the diameter of grain particles from about $10\mu m$ to a few millimeters.

- **Up to 7 gas species**
  Up to 7 gas species can be considered as chemical components of the gas phase. The molecular composition allowed are: $O_2, N_2, CO_2, H_2, H_2O, Air, SO_2$. Atmospheric air is considered as a single component with averaged properties. Gas species are treated as tracers and no chemical reactions are allowed up to now. Future versions of PDAC will hopefully consider interphase mass transfer (such as water vapour condensation) and chemical reactions.

- **Atmospheric stratification**
  A standard stratified atmosphere can be assigned as initial condition. The temperature gradient within the different atmospheric layers are the Earth averaged values, whereas pressure is computed by assuming hydrostatic equilibrium. No vertical stratification of humidity is assumed at this time. Cross wind can be imposed in a Cartesian reference frame.

- **Control of simulation options**
  The simple textual input file allows PDAC to be started from prescribed initial conditions or from a binary restart file, representing the state of the main flow fields when the code stops for any reason. When the code is started from scratch, vent inlet flow conditions the volcano topography, and the atmospheric conditions need to be specified as input conditions. When the code is restarted, the final state saved in the restart file is used as initial condition. The computational mesh and all numerical parameters are easily specified in the input file.

1.2 New features in version 2.0

The numerical code PDAC is based on previous numerical codes developed by Dobran et al. (1993), Neri and Macedonio (1996) and Neri et al. (2003) for the multiphase flow simulation of collapsing volcanic columns. We based the present release 2.0 on the Neri et al. (2003) model (considered here as version 1.0). If you are interested in knowing more about the history of PDAC consult the PDAC web site at http://www.pi.ingv.it/PDAC or contact the development team at pdac@pi.ingv.it

New modular structure

The present release 2.0 of PDAC has been fully rewritten in Fortran90 by adopting a modular approach. Dynamic allocation of memory allows the use of a precompiled PDAC code. In addition the more readable source code will help the programmers to mantain and develop PDAC more easily. If you wish to implement a new feature in PDAC you are encouraged to contact the developers team at pdac@pi.ingv.it for guidance.

Parallel implementation and improved serial performance

The parallel implementation of PDAC includes load balancing sheme and ad-hoc communication routines that allow PDAC to scale efficiently on many processors, thus reducing the execution time almost linearly. The data distribution method can be chosen from input to improve the load-balancing and thus the scalability. All routines have been optimized in order to improve the efficiency of the memory access. Optimized subroutines have been implemented in 3D for three phases and first-order simulations, with considerable performance gain.

Portable on most architectures

Any machine with a message passing library compatible with the MPI standard should be able to run PDAC in parallel. In recent years, distributed memory parallel computers have been offering cost-effective computational power. PDAC was designed to run efficiently on such parallel machines using large number of cells. PDAC is particularly well suited also for the increasingly popular Beowulf-class PC clusters, which are quite similar to the workstation clusters for which is was originally designed. Future versions of PDAC will also make efficient use of clusters of multi-processor workstations or PCs.

Second-order spatial and third-order temporal discretizations

Second-order spatial discretization schemes have been introduced in order to improve the code accuracy and to reduce the so-called numerical dissipation which represents a limit in the simulation of turbulent flows. High-order schemes are implemented as corrections of the first-order scheme previously used. Few parameters have to be specified in the input file to control numerical accuracy. Up to third-order low-storage Runge-Kutta temporal discretizations can be selected by the input file.

Dynamic Smagorinsky model

Dynamic Smagorinsky sub-grid scale turbulence model (Germano et al., *Phys. Fluids*, 1991) has been implemented. This procedure allows to compute the Smagorinsky constant from the simulated fluid flow. An input flag allows the selection of the standard or dynamic model.

Improved control on convergence and errors

More controls on the convergence and timing are added to reduce and easily identify the source of convergence errors.

Possible fully implicit solution of model equations

Fully implicit solution or implicit pressure coupling of the energy equation can be selected by setting appropriate input parameters. This can be useful in the evaluation of the accuracy of the numerical solution but it comes out to be computationally very expensive.

Easy model settings

Running inviscid simulation, or testing different turbulence models or closure relations can be easily controlled from the input files.

More friendly I/O

A slef-explanatory Input file is now available. Pre-processing of OUTPUT files is also introduced. Binary or ascii formatted output files can be selected.

1.3 User feedback

If you have problems installing or running PDAC after reading this document, please send a complete description of the problem by email to pdac@pi.ingv.it. If you discover and fix a problem not described in this manual we would appreciate if you would tell us about this as well, so we can alert other users and incorporate the correction into the public distribution.

We are interested in making PDAC more useful to the volcanological community. Your suggestions are welcome at pdac@pi.ingv.it We also appreciate hearing from you about how you are using PDAC in your work.

1.4 Collaborations

The formulation of the high-order numerical schemes, for both spatial and temporal discretizations, has been developed in collaboration with Maria Vittoria Salvetti (Dip.to di Ingegneria Aerospaziale, Università di Pisa) and François Beux (Scuola Normale Superiore, Pisa). We also credit Maria Vittoria Salvetti for the formulation of the turbulence model implemented in PDAC in the framework of the Large Eddy Simulation (LES) approach.

1.5 Acknowledgments

The developement of the PDAC release 2.0 has been supported by European project EXPLORIS (EVR1-CT2002-40026), Gruppo Nazionale per la Vulcanologia (Italy) and Ministero dell'Istruzione, dell'Università e della Ricerca (Italy).

The authors would also like to thank the High Performance Computing staff of CINECA-Italy, for providing the computational resources and expertises needed for developing and running PDAC.

2 Input and Output Files

This section describes all the files that are red or written by PDAC. Release 2.0 does not have a post-processing tool integrated, so that all interaction you have with the program is through textual files. The I/O files are always red and written in the running directory, i.e. the directory from which PDAC has been run. The level of verbosity, controlling the amount of information written by the program during the execution, can be set before starting a run. Note that a too high level of verbosity for very long-lasting simulations could cause the program to stop.

2.1 PDAC input file formats

PDAC has two possible input files from which it gets simulation parameters and the initial conditions.

A textual parameters file *pdac.dat* should be compiled by the user as descirbed below. PDAC will stop if this file is not present in the directory where the executable is running. It contains the namelists and cards with the parameters of the simulation and the various option controlling the program execution.

The other Input file that can be required by the program is the restart file *pdac.res*. PDAC will stop if the restart mode is selected but this file is not present in the directory where the executable is running. This is a binary file written by the code itself and contains the dump of all independent fields that are saved when a simulation run is interrupted for any reason. Therefore it cannot be edited by the user. Depending on the size of the problem envisaged, the restart file can be very large, so that you must ensure that your file-system can manage it.

In a parallel execution all input files are opened, red, and closed by the root processor (the one with MPI id equal to 0), that is charged of broadcasting data to the other computing units.

2.1.1 The *pdac.dat* file

This file contains the parameters that define the physical system you want to simulate with PDAC. *pdac.dat* is not red from standard input but is connected explicitly to the Fortran unit 5. The file should be present in the running directory of PDAC and it is always red at the beginning of the simulation run even if the run is a continuation/restart of a previous simulation run. The parameters and options in the *pdac.dat* file are specified in a set of F90 namelists followed by cards for topography, initial and boundary conditions. The options and values specified determine the exact behavior of PDAC, what features are active or inactive, how long the simulation should last, etc. Sect. 3 lists the parameters required to run a simulation, their default values and the range of variability. Several sample PDAC parameter files are shown in Sect. 4.

The layout of *pdac.dat* file is as follow:

```
&control
  control_parameters1 = control_value1,
  control_parameters2 = control_value2,
  ...
/

&model
```

```
  model_parameters1 = model_value1,
  model_parameters2 = model_value2,
  ...
/

&mesh
  mesh_parameters1 = mesh_value1,
  mesh_parameters2 = mesh_value2,
  ...
/

&particles
  particles_parameters1 = particles_value1,
  particles_parameters2 = particles_value2,
  ...
/

&numeric
  numeric_parameters1 = numeric_value1,
  numeric_parameters2 = numeric_value2,
  ...
/

'ROUGHNESS'
roughness_parameters

'MESH'
nonuniform_mesh_distances

'FIXED_FLOWS'
fixed_flows_parameters

'INITIAL_CONDITIONS'
initial_conditions_parameters
```

Within a namelist, the parameters are specified as

```
keyword      =     value
```

Blank lines in the namelists are ignored. Comments are prefaced by a `!` and may appear after the keyword assignment:

```
keyword  = value          !  This is a comment
```

or may be at the beginning of a line:

```
! keyword  = value   This is a keyword commented out
```

Some keywords require several values, that are specified as an array of values separated by a comma:

```
keyword  = value_1, value_2, value_3,
```

Note that the namelists (&control, &model, &mesh, &particles and &numeric) and the cards (`'ROUGHNESS'`, `'MESH'`, `'FIXED_FLOWS'` and `'INITIAL_CONDITIONS'`) should appear in exactly this order. Parameters lines in the namelists are optional, if a parameter is omitted it will take a default values (see Sect. 3). On the contrary parameter lines in the cards are not optional and should always be present. Commet are allowed in the namelist using the "!" character to identify that what follows is a comment. Commet are not allowed within the cards parameter, but you could insert free text (not including cards and namelists names) between different namelists and cards. Parameters within cards are in general distributed on more lines depending on the simulation parameters. The layout of the input parameters within each cards is described in details in Sect. 3.

2.1.2 The *pdac.res* file

PDAC dumps all fields and parameters that are required to restart a simulation from exactly the same instant on the file *pdac.res*. The file is in double precision binary format to maintain the numerical accuracy of the internal representation of numbers and to save space. This file is not meant for post-processing and analysis of the fields, since its internal layout could vary easily from one version of the code to the other. The restart file is connected to the Fortran unit 9, it is opened in *READ* mode at the beginning of a restart simulation run, to read in the status of a previous simulation interrupted. Then *pdac.res* file is opened in *WRITE* mode at regular intervals of simulated time. The rate at which *pdac.res* is overwritten with a new dump of the simulation status is specified by the user through the control parameter "tdump" (see Sect. 3). All the appropriate measures should be taken in order not to lose this file. We suggest to make a copy of the file in a safe device (such as tapes) before every run, since any error occurring when the file is opened in write mode could make you losing it.

2.2 PDAC output file formats

PDAC writes several output files which can be subdivided into information and data files. Information files (*pdac.log*, *pdac.tst* and *pdac.err*) contain logging, testing and debugging information meant for the user to control the progress of the simulation. Data files (*OUTPUT.nnnn*) contain the values of all the independent physical fields saved at regular interval of time, in different files distinguished by an incremental index (the "nnnn" in the file name). Obviously the restart file described above can be considered also as an output data file.

Data output files are opened, written and closed by the root processor (the one having MPI rank equal to 0), that gathers data from the other computing units before writing. All other information output files are opened, written and closed by all processors, and can be distinguished by an extension indicating the rank of the processor.

2.2.1 The *pdac.log* file

The file *pdac.log* is used by PDAC to log information concerning the progress of the simulation. *pdac.log* is connected explicitly to the Fortran unit 6. *pdac.log* is a free text file with self-explanatory information. Note that PDAC does not use standard output stream but on many systems the unit 6 is associated by default to the standard output. Therefore, on these systems, *pdac.log* is actually a redirected standard output stream. It is important also to underline that when a new *pdac.log* is opened PDAC cancels any existing *pdac.log* file.

2.2.2 The *pdac.tst* file

The file *pdac.tst*, like *pdac.log*, is a free text file that contains debugging and testing informations. This is meant for developers only, therefore the information present in it is not particularly useful to the common user. The file is connected to the Fortran unit 7 and is cleared every new run.

2.2.3 The *pdac.err* file

Like *pdac.log* and *pdac.tst*, *pdac.err* is a free textual file. It is connected to the Fortran unit 8. In this file PDAC writes all messages related to severe errors that cause the program to stop. Note that this file is not cleared at every run but it is opened in *APPEND* mode.

2.2.4 Output file

All scalar and vector fields related to the evolution of the simulated system are saved into the *OUTPUT.nnnn* files at regular intervals of simulated time. The user can choose the interval of time through the control parameter `tpr` (see Sect. 3). The extension "nnnn" is a progressive number, ranging from 0000 to 9999, incremented each time the fields are written. The output files are associated to the Fortran unit 12. The format of these file can be selected by the user. The logical control parameter `formatted_output` controls if the output will be written in formatted textual format or in binary (single precision) format. In general, the output files are meant for post-processing analysis, it is therefore useful to know their layout. In both formatted and binary forms, the fields are saved in this order:

```
time            ! Simulated time

p               ! Thermodynamic pressure
ug              ! Gas velocity along x
vg              ! Gas velocity along y (only for 3D simulations)
wg              ! Gas velocity along z
tg              ! Gas temperature

xgc             ! Molar fraction of gas components
...             ! (for all gas components)

eps(s)          ! Volumetric fraction of particle class (s)
us(s)           ! Particle velocity along x
vs(s)           ! Particle velocity along y (only for 3D simulations)
ws(s)           ! Particle velocity along z
ts(s)           ! Particle temperature
...             ! (for all solid phases)
```

where the first line is a scalar real value representing the simulated time at which the fields are written and the oter lines represent the fields (of dimension `nx*ny*nz`) that are written using the

Fortran format specifier `FORMAT( 5(G14.6E3,1X) )`. The components of the gas and solid velocities in the y dimension (`vg` and `vs`) are present in the output file only if the simulation type is 3D.

When the file is written in the binary unformatted format, time and each variable is written in a separate record. In particular time is written in a record of lenght 1 while all other fields in records of lenght `nx*ny*nz`.

2.3 Post-processing files

PDAC release 2.0 provides the executable *pp.x* utility, that is intended to be a pre-processing tool for visualization of output files produced by PDAC. Even though this tool is still under development its usage is shortly described in this section in order to make it useful for the user, especially when the output of very large simulations are too big to be processed directly by a visualization tool. *pp.x* extracts from a series of *OUTPUT.nnnn* files the individual single fields, with the possibility of downsize, interpolate, and crop the data and stores them in separate files that can be easily plotted with various graphical application.

2.3.1 Post-processing input files

The pre-processing tool must read the input file *pdac.dat*, to which an additional namelist with the parameters that control the post-processing is added. The name of the additional namelist is `&pp`, and should be typed in between the namelist &numeric and the first card `'ROUGHNESS'` . The layout of the *pdac.dat* file for the post-processing is then:

```
...
...

&numeric
  numeric_parameters1 = numeric_value1,
  numeric_parameters2 = numeric_value2,
  ...
/

&pp
  pp_parameter1 = pp_value1,
  pp_parameter2 = pp_value2,
  ...
/

'ROUGHNESS'
roughness_parameters

...
...
```

Note that you can leave the &pp namelist in the *pdac.dat* file for the usage of main program too since it will be simply ignored.

2.3.2 Post-processing output files

The post-processing produces several output files containing all individual fields and grid information useful for visualization. In particular, post-processing *pp.x* reads a sequence of *OUTPUT.nnnn* files (the numbers indicating the first, the last, and the increment of "nnnn" are red in input) and writes out a sequence of files for each individual field. The name of new files are of the form *filter.[fieldname].nnnn*. The format of these file is the same of output, as specified above. A list of all files produced by the post-processing is the following:

- *pdac.xml*
  this file contains all the information contained in *pdac.dat* written using an XML compliant format
- *pdac.grx*
  this file contains the list of cell centers (excluding border cells) in x-direction
- *pdac.gry*
  this file contains the list of cell centers (excluding border cells) in y-direction
- *pdac.grz*
  this file contains the list of cell centers (excluding border cells) in z-direction
- *filter.pgas.nnnn*
  these files contain the gas pressure field
- *filter.ug.nnnn*
  these files contain the gas velocity field in x-direction for the gas phase
- *filter.vg.nnnn*
  these files contain the gas velocity field in y-direction for the gas phase This file is not present in 2D simulations
- *filter.wg.nnnn*
  these files contain the gas velocity field in z-direction for the gas phase
- *filter.tgas.nnnn*
  these files contain the gas temperature field
- *filter.xgGG.nnnn*
  these files contain the molar fraction fields for all the gas components (GG runs from 01 to ngas)
- *filter.epSS.nnnn*
  these files contain the particle volume fraction fields for all the solid phases (SS runs from 01 to nsolid)
- *filter.tsSS.nnnn*
  these files contain the particles temperature fields for all solid phases (SS run from 01 to nsolid)

- *filter.usSS.nnnn*
  these files contain the particles velocity fields in x-direction for all solid phases (SS run from 01 to nsolid)
- *filter.vsSS.nnnn*
  these files contain the particles velocity fields in y-direction for all solid phases (SS run from 01 to nsolid) These files are not present in 2D simulations
- *filter.wsSS.nnnn*
  these files contain the particles velocity fields in z-direction for all solid phases (SS run from 01 to nsolid)

Note that when running the post-processing all *OUTPUT.nnnn* files in the selected range should be present in the running directory of *pp.x*

3 Simulation Parameters

3.1 PDAC input namelists

3.1.1 `control` namelist

The `control` namelist contains the parameters that control the program flow, such as the start mode, the initial and final time, the output printing.

- `run_name` < run identificative >
  Acceptable Values: any string
  Default Value: `pdac_run_2d`
  Description: This string identifies the simulation. Its length must not exceed 80 characters. It must be enclosed by quotes.

- `job_type` < 2d or 3d >
  Acceptable Values: `'2d','2D','3d','3D'`
  Default Value: `'2D'`
  Description: When 2D is selected, the model equations are simplified by assuming some simmetry in the physical domain (translation in one cartesian direction or cylindrical symmetry)

- `restart_mode` < flag for restart >
  Acceptable Values: `'from_scratch'`, `'restart'`
  Default Value: `'from_scratch'`
  Description: If `restart` is selected, initial conditions are set up by using the fields dumped (in double precision) into the restart file `pdac.res` (see above). Otherwise, initial conditions are set up from input data in `pdac.dat`

- `time` < initial time >
  Acceptable Values: double real
  Default Value: 0.0
  Description: Inital simulation time (in seconds)

- `tstop` < end time >
  Acceptable Values: double real
  Default Value: 100.0
  Description: Time (in seconds) at which the simulation stops

- `dt` < time step >
  Acceptable Values: double real
  Default Value: 0.01
  Description: Time advancement step (in seconds). `dt` is constrained by the CFL condition $dt < C_{max} \frac{\Delta x}{V_{max}}$, where Δx is the size of the computational grid and V_{max} is the maximum velocity. The maximum CFL number $C_{max} \approx 0.2$ has been found empirically.

- `lpr` < level of verbosity >
  Acceptable Values: 1,2,3
  Default Value: 2
  Description: increases the level of verbosity in warning and error messages in `pdac.log`, `pdac.err`, `pdac.tst` files

- `tpr` < time interval for OUTPUT file >
  Acceptable Values: > dt
  Default Value: 1.0
  Description: OUTPUT files are printed every `tpr` seconds of simulated time

- `tdump` < time interval for restart file >
  Acceptable Values: > dt
  Default Value: 20.0
  Description: restart file is overwritten every `tdump` seconds of simulated time

- `max_seconds` < maximum CPU time >
  Acceptable Values: any real value
  Default Value: 20000.0
  Description: Maximum duration of a simulation (in seconds). If time exceeds this value a restart file is written before the simulation is stopped (useful for scheduling). Default time is set to 6 hours

- `nfil` < number of first OUTPUT file >
  Acceptable Values: any positive integer up to 9999
  Default Value: 0
  Description: OUTPUT files are written with 4 digits extension (`OUTPUT.XXXX`). The output time can be recovered as $(\texttt{XXXX} - \texttt{nfil}) * \texttt{tpr}$

- `formatted_output` < flag for OUTPUT format >
  Acceptable Values: T/F
  Default Value: T
  Description: Determine the format of output files: T - formatted ascii, F - binary 4-bytes

3.1.2 `model` namelist

The `model` namelist is intended for all switches that can be selected by the user to modify the model equations, either by neglecting some terms in the transport equations or by using different constitutive equations and submodels.

- `irex` < chemical reactions >
  Acceptable Values: 0,1
  Default Value: 0
  Description: The chemical reaction module has not yet been implemented

- `gas_viscosity` < gas diffusive transport >
  Acceptable Values: `T/F`
  Default Value: `T`
  Description: T to solve the full set of model equations. F switches the diffusive transport terms off for the gas phase (viscous and turbulent term in momentum equation, thermal diffusivity in enthalpy equation). Gas viscosity is computed anyway to include the gas-particle drag and energy exchange terms.

- `part_viscosity` < particle diffusive transport >
  Acceptable Values: `T/F`

Default Value: `T`
Description: Switches on/off the diffusive terms in the particle transport equations (viscous terms in the momentum equation and thermal conductivity).

- `iss` < turbulence model for particles >
  Acceptable Values: 0,1
  Default Value: 0
  Description: 1: computes a gas-analogous sub-grid stress (turbulent viscosity) for particles.

- `repulsive_model` < flag for Coulombic repulsive model >
  Acceptable Values: 0,1
  Default Value: 1
  Description: The Coulombic repulsive model add a contribution to the diagonal part of the solid viscous stress due to the repulsive interaction of particles at high concentrations. This term appears to be important for the solid equation to be well-posed.

- `iturb` < turbulence model for gas >
  Acceptable Values: 0,1,2
  Default Value: 1
  Description: 0 - no gas turbulence model; 1 - Smagorinsky sub-grid stress model; 2 - Smagorinsky sgs model with roughness closure at the walls.

- `modturbo` < Subgrid-scale model for gas turbulence >
  Acceptable Values: 1,2
  Default Value: 1
  Description: 1 - Classical Smagrinsky model; 2 - Dynamic Smagorinsky model

- `cmut` < Smagorinsky constant >
  Acceptable Values: usually between 0.1 and 0.4
  Default Value: 0.1
  Description: The exact value cannot be predicted *a priori*. The use of the dynamic Smagorinsky model (see above) makes the assignment of this constant non-necessary

- `rlim` < Multiphase limit >
  Acceptable Values: very small real value
  Default Value: 10^{-8}
  Description: It is related to minimum particle concentration for which multiphase flow equations are solved. Below this limit one-phase equations are solved (often critical for convergence on the cloud margins).

- `gravx` < acceleration along x >
  Acceptable Values: any real
  Default Value: 0.0
  Description: Body-force in x(r)-direction

- `gravy` < acceleration along y >
  Acceptable Values: any real
  Default Value: 0.0
  Description: Body-force in y-direction

- `gravz` < acceleration along z >
  Acceptable Values: any real
  Default Value: -9.81
  Description: Body-force in z-direction (usually the value of the gravitational acceleration). Values different from the default value could not be consistent with the atmospheric stratification, leading to instabilities. A value of 0.0 suppresses atmospheric stratification.

- `ngas` < number of gas components >
  Acceptable Values: 1 to 7
  Default Value: 2
  Description: Seven gas species are defined, specifically:
  $1)O_2, 2)N_2, 3), CO_2, 4)H_2, 5)H_2O, 6)Air, 7)SO_2$. Only gas species that are specified at the inlet or in the atmosphere are considered by the model. The total number of gas species specified by this flag must be therefore consistent with input conditions, otherwise the program stops.

- `density_specified` < flag for specified flow conditions >
  Acceptable Values: T/F
  Default Value: F
  Description: Setting this flag to T allows to specify gas density at the inlet instead of temperature. Gas temperature is then computed by using the perfect gas thermal equation of state.

3.1.3 `mesh` namelist

Here all parameters concerning the spatial discretization of the computational domain must be selected. In PDAC you are constrained to discretization on a rectilinear (non-)uniform mesh. Cylindrical coordinates can be selected only in 2D.

- `nx` < Number of cells in x(r)-direction >
  Acceptable Values: up to 512
  Default Value: 100
  Description: When 2D cylindrical coordinates are selected, "x" is used instead of "r". This number includes the boundary ghost cells. The maximum number can be increased by modifying the max_size parameter in the "dimensions" module

- `ny` < Number of cells in y-direction >
  Acceptable Values: up to 512
  Default Value: 1
  Description: Not used in 2D. It includes the boundary ghost cells.

- `nz` < Number of cells in z-direction >
  Acceptable Values: up to 512
  Default Value: 100
  Description: z is the second space coordinate in 2D. It includes the boundary ghost cells.

- `itc` < flag for cylindrical coordinates >
  Acceptable Values: 0,1
  Default Value: 0

Description: In 2D simulations, itc=1 sets the grid and the coordinates to cylindrical, by modifying the discretized equations.

- `iuni` < flag for uniform mesh >
  Acceptable Values: 0,1
  Default Value: 0
  Description: 0 - non uniform mesh; 1 - uniform mesh, takes `dx0, dy0, dz0` as the cells size in the two/three coordinate directions.

- `dx0` < cell sizes in x(r)-direction >
  Acceptable Values: any real
  Default Value: 10.D0
  Description: Uniform cell sizes along x(r)

- `dy0` < cell sizes in y-direction >
  Acceptable Values: any real
  Default Value: 10.D0
  Description: Uniform cell sizes along y

- `dz0` < cell sizes in z-direction >
  Acceptable Values: any real
  Default Value: 10.D0
  Description: Uniform cell sizes along z

- `origin_x` < origin of the cartesian space >
  Acceptable Values: any real
  Default Value: 0.0
  Description: Origin of the x-axis

- `origin_y` < origin of the cartesian space >
  Acceptable Values: any real
  Default Value: 0.0
  Description: Origin of the y-axis

- `origin_z` < origin of the cartesian space >
  Acceptable Values: any real
  Default Value: 0.0
  Description: Origin of the z-axis

- `mesh_partition` < domain decomposition criterion >
  Acceptable Values: 1,2,3
  Default Value: 1
  Description: 1 - Layers decomposition, 2 - 2D Blocks decomposition, 3 - 3D Blocks decomposition

3.1.4 `particles` namelist

In this namelist the number of solid phases and the physical properties of the particles forming each phase are specified. Note that the order in which particle classes are specified must be the order of the solid-phases array storage, in order to be consistent with the initial conditions assignement.

- `nsolid` < number of solid phases >
  Acceptable Values: up to 10
  Default Value: 2
  Description: The number of solid phases considered in the multiphase equations. The maximum number can be increased by modifying the max_nsolid parameter in the "dimensions" module.

- `diameter` < effective diameter particles (in microns) >
  Acceptable Values: real
  Default Value: 100 microns
  Description: The model works well for particle diameters smaller than few millimeters

- `density` < microscopic particle density >
  Acceptable Values: real
  Default Value: 2700 Kg/m^3
  Description: Depends on materials and porosity

- `sphericity` < particle sphericity >
  Acceptable Values: 0.5 to 1.0
  Default Value: 1.0
  Description: Partially accounts for particle shapes

- `viscosity` < Empirical viscosity coefficient >
  Acceptable Values: 0.5 to 2.0
  Default Value: 0.5[Pa s]
  Description: Largest values apply to coarse particles

- `specific_heat` < solid specific heat >
  Acceptable Values: real
  Default Value: 1.2D3 [J/(K Kg)]
  Description: Depends on materials

- `thermal_conductivity` < solid thermal conductivity for Fourier law >
  Acceptable Values: real
  Default Value: 2.D0
  Description: Depends on materials

3.1.5 `numeric` namelist

By modifying these parameters, you can modify the way PDAC solves the model equations. We recommend non-expert users to avoid modifying the default values.

- `rungekut` < order of Runge-Kutta explicit integration >
  Acceptable Values: 1,2,3
  Default Value: 1
  Description: The low-storage Runge-Kutta algorithm is used for explicit time integration. The coefficients used in the RK integration are well suited only up to the third-order. High-order temporal integration is recommended for convergence and stability when high-order spatial discretization schemes are used.

- `beta` < degree of upwinding >
  Acceptable Values: 0.0 to 1.0
  Default Value: 0.25
  Description: When High Order spatial discretization schemes are used and the MUSCL beta-scheme (both limited or unlimited) is selected, the degree of upwinding can be chosen: `beta`=0.0 corresponds to a completely centered schemes: `beta`=1.0 corresponds to a completely upwinded method. The accuracy depends on the exact value: for `beta`=0.25 and `beta`=0.33 the formal third-order is achieved.

- `muscl` < flag for MUSCL technique >
  Acceptable Values: 0,1
  Default Value: 0
  Description: The MUSCL technique extends the first-order discretization scheme to high-orders by reconstructing the flux profile more accurately. 0 - uses first-order upwind; 1 - uses MUSCL technique.

- `lim_type` < limiter type >
  Acceptable Values: 0,1,2,3,4
  Default Value: 0
  Description: Select the type of MUSCL reconstruction and the limiter:
  0 - beta scheme unlimited; 1 - Van Leer limiter; 2 - Minmod limiter; 3 - Superbee limiter; 4 - beta limited.

- `inmax` < maximum number of inner iterations >
  Acceptable Values: integer (small)
  Default Value: 8
  Description: The exact value does not affect strongly the convergence but can slow down the simulation. Optimal value is set by default.

- `maxout` < maximum number of Gauss-Siedel iterations for convergence >
  Acceptable Values: integer (large)
  Default Value: 5000
  Description: Convergence is usually reached within less than 100 iterations. If maxout is reached the code CRASH!es

- `omega` < over/under-relaxation parameter >
  Acceptable Values: 0.0 to 2.0
  Default Value: 1.0
  Description: Values between 0.0 and 1.0 under-relax the iterative procedure, whereas values above 1.0 overrelax.

- `implicit_fluxes` < flag for implicit computation of fluxes >
  Acceptable Values: `T/F`
  Default Value: F
  Description: By selecting T, convective fluxes are computed (at the first or second-order, depending on flag `muscl`) within the iterative solver. This modification significantly slows down the computation but could relax the CFL constraint so that larger time-steps can be used.

- `implicit_enthalpy` < flag for implicit computation of enthalpies >
  Acceptable Values: `T/F`
  Default Value: F
  Description: Selecting the implicit solution of enthalpies enhances the level of coupling between the momentum and enthalpy equations. Nevertheless the convective part of the enthalpy equations can be left out from the iterative solver by opportunely selecting the flag `implicit_fluxes`. The performance loss in any case is significant.

3.2 PDAC input cards

3.2.1 `'ROUGHNESS'` card

The `'ROUGHNESS'` card is only valid for 2D simulations (in 3D it will be replaced by the roughness matrix). In this card three elements must be specified:

- `ir` < number of rough zones >
  Acceptable Values: 1,2
  Description: The code allows the specification of one or two roughness regions, characterized by different roughness lengths.

- `zrough(:)` < roughness length >
  Acceptable Values: real
  Description: Specify one roughness length for each rough region. Typical values for ground ranges from few centimeters to some metres.

- `roucha` < distance of roughness change >
  Acceptable Values: real
  Description: The distance from the left boundary at which the roughness changes.

3.2.2 `'MESH'` card

The `'MESH'` card contains the two (in 2D) or three (in 3D) arrays of cell sizes specified for a non-uniform rectilinear mesh.

- `dx(1:nx)` < Array of the cell sizes in x(r)-direction >
  Acceptable Values: real
  Description: Array of the cell sizes in x(r)-direction.

- `dy(1:ny)` < Array of the cell sizes in y-direction >
  Acceptable Values: real
  Description: Array of the cell sizes in y-direction. Not present in 2D.

- `dz(1:nz)` < Array of the cell sizes in z-direction >
  Acceptable Values: real
  Description: Array of the cell sizes in z-direction.

Array elements can be separated by commas, tabs, blanks or lay on different lines. When uniform mesh is selected by `iuni=1` parameter in the `mesh` namelist, the card can be empty (but the card name must always be present in the input file).

3.2.3 `'FIXED_FLOWS'` card

The `'FIXED_FLOWS'` card is designed to specify boundary conditions (b.cs.). Its name is due to the possibility to assign within this card any region with specified flow conditions (e.g. for inlet flow). B.cs. are imposed in mesh region determined by rectangular blocks. The number of blocks is the first parameter specified in the card

- `number_of_block` < number of blocks >
  Acceptable Values: integer
  Description: The number of blocks used to specify b.c.

Each block is characterized by a flag that specifies the kind of b.c., and by two (in 2D) or three (in 3D) couples of integer that specify the first and the last cell of the block in the two (three) directions. Each block is therefore identified by 5 (in 2D) or 7 (in 3D) integers:

- `block_type` < type of b.c. >
  Acceptable Values: 1 to 7
  Description: 1 - fluid cells: all equations are solved; 2 - free-slip; 3 - no-slip; 4 - free in/outflow; 5 - specified flow (inlet); 6 - specified pressure; 7 - specified input profile (only 2D).

- `block_bounds` < limits of blocks >
  Acceptable Values: integer
  Description: two or three couples of integers specifying $i_{min}, i_{max}, (j_{min}, j_{max}), k_{min}, k_{max}$

Blocks are used also to specify "blocking cells" (e.g. the volcano topography). When the block type is equal to 5 (specified flow), inlet conditions must be specified as follows:

First line:

- `fixed_vgas_x` < gas velocity component along x >
  Acceptable Values: real
  Description: Inlet gas velocity along x(r)-direction. Note that CFL condition (see above for `dt`) must be satisfied for every component.

- `fixed_vgas_y` < gas velocity component along y >
  Acceptable Values: real
  Description: Inlet gas velocity along x(r)-direction (not present in 2D!).

- `fixed_vgas_z` < gas velocity component along z >
  Acceptable Values: real
  Description: Inlet gas velocity along z-direction.

- `fixed_pressure` < pressure at inlet >
  Acceptable Values: real
  Description: The thermodynamic pressure of the gas phase.

- `fixed_gaseps` < inlet gas volumetric fraction >
  Acceptable Values: ≤ 1.0
  Description: This value must be consistent with the solid phases volumetric fractions. The closure relation imposes that the sum equals one.

- `fixed_gastemp` < inlet gas temperature >
  Acceptable Values: a positive real
  Description: No constraint is imposed on gas temperature, but the range of validity of the equation of state must be check.

Further `nsolid` lines:

- `fixed_vpart_x` < particle velocity along x >
  Acceptable Values: real
  Description: Inlet particle velocity along x(r)-direction. Note that CFL condition (see above for `tt`) must be satisfied for every component.

- `fixed_vpart_y` < particle velocity along y >
  Acceptable Values: real
  Description: Inlet particle velocity along y-direction(not present in 2D!).

- `fixed_vpart_z` < particle velocity along z >
  Acceptable Values: real
  Description: Inlet particle velocity along z-direction.

- `fixed_parteps` < particle volumetric fraction >
  Acceptable Values: ≤ 1.0
  Description: Must satisfy closure relation of total volumetric fraction

- `fixed_parttemp` < particle temperature >
  Acceptable Values: a positive real
  Description: Particle temperature is not constrained by a thermal equation of state

Last line:

- `fixed_gasconc(1:7)` < concentration of each gas species at inlet >
  Acceptable Values: ≤ 1.0
  Description: The mass fraction (concentration) of each of the seven gas species must be specified. The closure relation for the total mass fraction must be satisfied. If this constrain is not satisfied, the `default_gas` mass fraction is automatically corrected.

3.2.4 `'INITIAL_CONDITIONS'` card

The `'INITIAL_CONDITIONS'` card specifies the ambient conditions at the beginning of the simulation in all the computational cells of type 1 (fluid cells).

First line:

- `initial_vgas_x` < gas velocity component along x >
  Acceptable Values: real
  Description: Initial gas velocity along x(r)-direction. Can be used for cross-winds. Please note that the CFL condition (see above for `dt`) must be satisfied for every component.

- `initial_vgas_y` < gas velocity component along y >
  Acceptable Values: real
  Description: Inlet gas velocity along y-direction (not present in 2D!).

- `initial_vgas_z` < gas velocity component along z >
  Acceptable Values: real
  Description: Inlet gas velocity along z-direction.

- `initial_pressure` < pressure at inlet >
  Acceptable Values: real
  Description: This corresponds to the thermodynamic ambient pressure (if gravity is switched off) or the pressure at the ground (if atmosphere is stratified, values different from 1 atmosphere lead to a WARNING message).

- `initial_void_fraction` < atmospheric gas volumetric fraction >
  Acceptable Values: ≤ 1.0
  Description: Solid phases volumetric fractions are computed to satisfy the closure relation

- `initial_temperature` < inlet gas temperature >
  Acceptable Values: a positive real
  Description: This corresponds to the ambient temperature (if gravity is switched off) or the temperature at the ground (if atmosphere is stratified, values different from 288.15K lead to a WARNING message). No constraint is imposed on gas temperature, but the range of validity of the equation of state must be check. Particles are supposed to be in thermal equilibrium with gas.

Further `nsolid` lines:

- `initial_vpart_x` < particle velocity along x >
  Acceptable Values: real
  Description: Initial particle velocity along x(r)-direction. Note that CFL condition (see above for `dt`) must be satisfied for every component.

- `initial_vpart_y` < particle velocity along y >
  Acceptable Values: real
  Description: Initial particle velocity along y-direction (not present in 2D!).

- `initial_vpart_z` < particle velocity along z >
  Acceptable Values: real
  Description: Initial particle velocity along z-direction.

Last line:

- `initial_gasconc(1:7)` < concentration of each gas species in atmosphere >
  Acceptable Values: ≤ 1.0
  Description: The mass fraction (concentration) of each of the seven gas species must be specified. The closure relation for the total mass fraction must be satisfied. If this constrain is not satisfied, the `default_gas` mass fraction is automatically corrected.

4 Sample configuration files

This section contains some examples of PDAC configuration files to be used as templates.

4.1 2D axisymmetric Plinian column

- one particle phase (50 μm diameter, $1500 Kg/m^3$ density)
- no topographic profile specified
- Smagorinsky turbulence model with $C_S = 0.33$
- high-order (Ultraquick scheme) discretization of convective fluxes
- non-uniform mesh with minimum cell size of 30m and maximum cell size of 200 m in both radial and vertical directions
- vent conditions specified in cells (2,1) and (3,1): gas and particle vertical velocity $w = 110 m/s$, Pressure balanced ($1 atm$), 0.1% particle fraction, equal gas and particle temperature ($T = 900K$). Water vapour is injected at the vent.
- initial conditions correspond to the standard atmosphere

```
&control
 run_name = 'Sub_Plinian_2D',
 job_type = '2D',
 restart_mode = 'from_scratch', ! ( from_scratch | restart )
 time =  0.00,
 tstop = 900.0,
 dt    = 0.01,
 tpr   = 10.0,
 tdump = 50.0
/

&model
  cmut = 0.33,
  iss  = 0,
  repulsive_model = 1,
  iturb = 1
/

&mesh
  itc = 1,
  nx = 100,
  nz = 200
/

&particles
  nsolid = 1,
  diameter = 50.
  density = 1500.
  sphericity = 1.0
  viscosity = 0.5
  specific_heat = 1.2D3
  thermal_conductivity = 2.0D0
/

&numeric
  rungekut = 1,
  muscl = 1,
  lim_type = 4,
  beta = 0.25,
  omega = 1.00
/

'ROUGHNESS'
 1, 1.D0, 5.D3

'MESH'

  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  32.95D0  35.90D0  38.85D0  41.80D0  44.74D0
  47.69D0  50.64D0  53.59D0  56.54D0  59.49D0
  62.44D0  65.39D0  68.34D0  71.29D0  74.23D0
  77.18D0  80.13D0  83.08D0  86.03D0  88.98D0
  91.93D0  94.88D0  97.83D0 100.78D0 103.72D0
 106.67D0 109.62D0 112.57D0 115.52D0 118.47D0
 121.42D0 124.37D0 127.32D0 130.27D0 133.21D0
 136.16D0 139.11D0 142.06D0 145.01D0 147.96D0
 150.91D0 153.86D0 156.81D0 159.76D0 162.70D0
 165.65D0 168.60D0 171.55D0 174.50D0 177.45D0
 180.40D0 183.35D0 186.30D0 189.24D0 192.19D0
 195.14D0 198.09D0 200.00D0 200.00D0 200.00D0
 200.00D0 200.00D0 200.00D0 200.00D0 200.00D0
 200.00D0 200.00D0 200.00D0 200.00D0 200.00D0

  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  31.09D0  32.19D0  33.28D0  34.38D0  35.47D0
  36.57D0  37.66D0  38.76D0  39.85D0  40.95D0
  42.04D0  43.14D0  44.23D0  45.33D0  46.42D0
  47.52D0  48.61D0  49.70D0  50.80D0  51.89D0
  52.99D0  54.08D0  55.18D0  56.27D0  57.37D0
  58.46D0  59.56D0  60.65D0  61.75D0  62.84D0
  63.94D0  65.03D0  66.13D0  67.22D0  68.31D0
  69.41D0  70.50D0  71.60D0  72.69D0  73.79D0
  74.88D0  75.98D0  77.07D0  78.17D0  79.26D0
  80.36D0  81.45D0  82.55D0  83.64D0  84.73D0
  85.83D0  86.92D0  88.02D0  89.11D0  90.21D0
  91.30D0  92.40D0  93.49D0  94.59D0  95.68D0
  96.78D0  97.87D0  98.97D0 100.06D0 101.16D0
 102.25D0 103.34D0 104.44D0 105.53D0 106.63D0
 107.72D0 108.82D0 109.91D0 111.01D0 112.10D0
 113.20D0 114.29D0 115.39D0 116.48D0 117.58D0
 118.67D0 119.77D0 120.86D0 121.95D0 123.05D0
 124.14D0 125.24D0 126.33D0 127.43D0 128.52D0
 129.62D0 130.71D0 131.81D0 132.90D0 134.00D0
 135.09D0 136.19D0 137.28D0 138.38D0 139.47D0
 140.56D0 141.66D0 142.75D0 143.85D0 144.94D0
 146.04D0 147.13D0 148.23D0 149.32D0 150.42D0
 151.51D0 152.61D0 153.70D0 154.80D0 155.89D0
 156.98D0 158.08D0 159.17D0 160.27D0 161.36D0
 162.46D0 163.55D0 164.65D0 165.74D0 166.84D0
 167.93D0 169.03D0 170.12D0 171.22D0 172.31D0
 173.41D0 174.50D0 175.59D0 176.69D0 177.78D0
 178.88D0 179.97D0 181.07D0 182.16D0 183.26D0
 184.35D0 185.45D0 186.54D0 187.64D0 188.73D0
 189.83D0 190.92D0 192.02D0 193.11D0 194.20D0
 195.30D0 196.39D0 197.49D0 198.58D0 199.68D0
 200.00D0 200.00D0 200.00D0 200.00D0 200.00D0

'FIXED_FLOWS'
5
5, 2  , 3  , 1  , 1
0.0, 110., 1.01325D5, 0.999, 900.0
0.0, 110., 0.001, 900.0
0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0
3, 4  , 100, 1  , 1
2, 1  , 1  , 1  , 200
6, 100, 100, 2  , 200
```

```
6, 2  , 100, 200, 200

'INITIAL_CONDITIONS'
0.0, 0.0, 1.01325D5, 1. 0.6413, 288.15
0.0, 0.0
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0
```

4.2 Classical Sod problem

The Sod problem is the numerical analogous to the laboratory shock-tube experiment, and it is used to check numerical discretization schemes on the standard 1D Riemann problem.

- Pseudo-one-dimensional (transverse direction has one computational cell and two boundary cells)
- Cartesian regular mesh
- One inviscid gas species
- Particle phase is dummy (volumetric fraction is set to zero). Please note that it is not allowed to set `nsolid = 0`, otherwise the program stops.
- Gravity force is neglected
- Density is specified as initial condition instead of temperature

```
&control
 run_name = 'shock_tube',
 job_type = '2D',
 restart_mode = 'from_scratch',
 time =  0.00,
 tstop = 0.20,
 dt    = 0.001,
 tpr   = 0.05,
 tdump = 5.0,
 nfil = 0
/

&model
 density_specified = T,
 gas_viscosity = F,
 iturb = 0,
 gravz = 0,
 ngas = 1
/

&mesh
  itc  = 0,
  iuni = 1,
  nx = 3,
  nz = 102,
  dx0 = 1.0d-2,
  dz0 = 1.0d-2
/
```

```
&particles
  nsolid = 1,
  diameter = 200.D0,
  density = 1500.D0,
  sphericity = 1.0,
  viscosity = 1.0,
  specific_heat = 1.2D3,
  thermal_conductivity = 2.0D0,
/

&numeric
  rungekut = 1,
  beta = 0.25,
  lim_type = 4,
  muscl = 0,
  omega = 1.00
/

'ROUGHNESS'

'MESH'

'FIXED_FLOWS'
6
2, 1, 3, 102, 102
2, 1, 1, 1, 102
2, 3, 3, 1, 102
1, 2, 2, 2, 31
0.0, 0.75, 1.D0, 1.0, 1.0  ! Last value is the density !
0.0, 0.0, 0.0, 0.0
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0
1, 2, 2, 32, 101
0.0, 0.0, 1.D-1, 1.0, 0.125  ! Last value is the density !
0.0, 0.0, 0.0, 0.0
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0
5, 2, 2, 1, 1
0.0, 0.75, 1.D0, 1.0, 1.0  ! Last value is the density !
0.0, 0.0, 0.0, 0.0
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0

'INITIAL_CONDITIONS'
0.0, 0.0, 1.0D0, 1.0 0.6413, 288.15
0.0, 0.0
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0
```

4.3 Simulation of pyroclastic flow at Vesuvius

- 2D axisymmetric simulation
- standard multiphase model
- volcano topography is introduced by specifying “blocking cells”. These are described by topographic blocks in the `FIXED FLOW` card.
- roughness at ground is specified

```
&control
 run_name = 'LL_w2_t950_A' ! ( Todesco et al. Bull.Volcanol. (2002) )
 job_type = '2D',
 restart_mode = 'from_scratch',
 time =  0.00,
 tstop = 800.0,
 dt    = 0.01,
 tpr   = 5.0,
 tdump = 10.0
/
&model
 iss = 0,
 repulsive_model = 1,
 iturb = 2
/

&mesh
  nx = 146,
  nz = 251,
  itc = 1,
  iuni = 0
/

&particles
  nsolid = 2,
  diameter = 30., 500.
  density = 2800., 1000.,
  sphericity = 1.0, 1.0,
  viscosity = 0.5,  1.0,
  specific_heat = 1.2D3, 1.2D3,
  thermal_conductivity = 2.0D0, 2.0D0
/

&numeric
  rungekut = 1,
  muscl = 0,
  omega = 1.00
```

```
/

'ROUGHNESS'
2, 1.D0, 2.D0, 5.D1

'MESH'

0.75D1, 0.75D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
2.D1, 2.D1, 2.D1, 2.D1, 2.D1, 3.D1, 3.D1, 3.D1, 3.D1, 3.D1,
4.D1, 4.D1, 4.D1, 4.D1, 4.D1, 5.D1, 5.D1, 5.D1, 5.D1, 5.D1,
6.D1, 6.D1, 6.D1, 6.D1, 6.D1, 7.D1, 7.D1, 7.D1, 7.D1, 7.D1,
8.D1, 8.D1, 8.D1, 8.D1, 8.D1, 9.D1, 9.D1, 9.D1, 9.D1, 9.D1,
1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2,
1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2,
1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2,
1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2,
1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.D2,
1.D2, 1.D2, 1.D2, 1.D2, 1.D2

1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1, 1.D1,
2.D1, 2.D1, 2.D1, 2.D1, 2.D1, 3.D1, 3.D1, 3.D1, 3.D1, 3.D1,
4.D1, 4.D1, 4.D1, 4.D1, 4.D1, 5.D1, 5.D1, 5.D1, 5.D1, 5.D1,
6.D1, 6.D1, 6.D1, 6.D1, 6.D1, 7.D1, 7.D1, 7.D1, 7.D1, 7.D1,
8.D1, 8.D1, 8.D1, 8.D1, 8.D1, 9.D1, 9.D1, 9.D1, 9.D1, 9.D1,
1.D2, 1.D2, 1.D2, 1.D2, 1.D2, 1.2D2, 1.2D2, 1.2D2, 1.2D2, 1.2D2,
1.4D2, 1.4D2, 1.4D2, 1.4D2, 1.4D2, 1.6D2, 1.6D2, 1.6D2, 1.6D2, 1.6D2,
1.8D2, 1.8D2, 1.8D2, 1.8D2, 1.8D2, 2.D2, 2.D2, 2.D2, 2.D2, 2.D2,
2.5D2, 2.5D2, 2.5D2, 2.5D2, 2.5D2, 3.D2, 3.D2, 3.D2, 3.D2, 3.D2,
3.5D2, 3.5D2, 3.5D2, 3.5D2, 3.5D2, 4.D2, 4.D2, 4.D2, 4.D2, 4.D2,
4.5D2, 4.5D2, 4.5D2, 4.5D2, 4.5D2, 5.D2, 5.D2, 5.D2, 5.D2, 5.D2,
6.D2, 6.D2, 6.D2, 6.D2, 6.D2, 7.D2, 7.D2, 7.D2, 7.D2, 7.D2,
8.D2, 8.D2, 8.D2, 8.D2, 8.D2, 9.D2, 9.D2, 9.D2, 9.D2, 9.D2,
1.D3, 1.D3, 1.D3, 1.D3, 1.D3, 1.D3, 1.D3, 1.D3, 1.D3, 1.D3

'FIXED_FLOWS'
116
5, 2, 7, 98, 98
0.0, 137.10, 17.0D5, 0.9069, 1223.0
0.0, 137.10, 0.0245, 1223.0
0.0, 137.10, 0.0686, 1223.0
0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0
3, 1, 7, 1, 97
3, 8, 8, 1, 99
3, 9, 9, 1, 100
3, 10, 10, 1, 101
3, 11, 11, 1, 102
3, 12, 12, 1, 103
3, 13, 13, 1, 104
3, 14, 14, 1, 105
3, 15, 15, 1, 106
3, 16, 16, 1, 107
3, 17, 17, 1, 108
3, 18, 18, 1, 109
3, 19, 19, 1, 110
3, 20, 20, 1, 111
3, 21, 21, 1, 112
3, 22, 22, 1, 113
3, 23, 23, 1, 114
3, 24, 24, 1, 115
3, 25, 25, 1, 116
3, 26, 26, 1, 117
3, 27, 27, 1, 118
3, 28, 29, 1, 119
3, 30, 30, 1, 118
3, 31, 31, 1, 117
3, 32, 32, 1, 116
3, 33, 34, 1, 115
3, 35, 35, 1, 114
3, 36, 36, 1, 113
3, 37, 37, 1, 112
3, 38, 39, 1, 111
3, 40, 40, 1, 110
3, 41, 41, 1, 109
3, 42, 42, 1, 108
3, 43, 44, 1, 107
3, 45, 45, 1, 106
3, 46, 46, 1, 105
```

```
3, 47, 47, 1, 104
3, 48, 49, 1, 103
3, 50, 50, 1, 102
3, 51, 51, 1, 101
3, 52, 52, 1, 100
3, 53, 53, 1, 98
3, 54, 54, 1, 96
3, 55, 55, 1, 94
3, 56, 56, 1, 92
3, 57, 57, 1, 91
3, 58, 58, 1, 90
3, 59, 59, 1, 89
3, 60, 60, 1, 87
3, 61, 61, 1, 86
3, 62, 62, 1, 85
3, 63, 63, 1, 84
3, 64, 64, 1, 82
3, 65, 65, 1, 81
3, 66, 66, 1, 79
3, 67, 67, 1, 78
3, 68, 68, 1, 76
3, 69, 69, 1, 75
3, 70, 70, 1, 73
3, 71, 71, 1, 71
3, 72, 72, 1, 70
3, 73, 73, 1, 68
3, 74, 74, 1, 67
3, 75, 75, 1, 66
3, 76, 76, 1, 65
3, 77, 77, 1, 64
3, 78, 78, 1, 62
3, 79, 79, 1, 60
3, 80, 80, 1, 59
3, 81, 81, 1, 58
3, 82, 82, 1, 56
3, 83, 83, 1, 55
3, 84, 84, 1, 53
3, 85, 85, 1, 52
3, 86, 86, 1, 50
3, 87, 87, 1, 49
3, 88, 88, 1, 47
3, 89, 89, 1, 46
3, 90, 90, 1, 44
3, 91, 91, 1, 43
3, 92, 92, 1, 41
3, 93, 93, 1, 40
```

```
3, 94, 94, 1, 38
3, 95, 95, 1, 36
3, 96, 96, 1, 34
3, 97, 97, 1, 32
3, 98, 98, 1, 30
3, 99, 99, 1, 28
3, 100, 100, 1, 26
3, 101, 101, 1, 24
3, 102, 103, 1, 22
3, 104, 104, 1, 21
3, 105, 106, 1, 20
3, 107, 107, 1, 19
3, 108, 109, 1, 18
3, 110, 110, 1, 17
3, 111, 112, 1, 16
3, 113, 113, 1, 15
3, 114, 115, 1, 14
3, 116, 116, 1, 13
3, 117, 118, 1, 12
3, 119, 119, 1, 11
3, 120, 121, 1, 10
3, 122, 122, 1, 9
3, 123, 124, 1, 8
3, 125, 125, 1, 7
3, 126, 127, 1, 6
3, 128, 128, 1, 5
3, 129, 130, 1, 4
3, 131, 131, 1, 3
3, 132, 133, 1, 2
3, 134, 146, 1, 1
2, 1, 1, 98, 251
4, 2, 146, 251, 251
4, 146, 146, 2, 251

'INITIAL_CONDITIONS'
0.0, 0.0, 1.01325D5, 1.0 0.6413, 288.15
0.0, 0.0
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0
```

4.4 3D simulation of a volcanic column

- fully 3D simulation on a non-uniform cartesian mesh
- no topography specified

```
&control
 run_name = 'Sub_Plinian',
 job_type = '3D',
 restart_mode = 'from_scratch',
 time =  0.00,
 tstop = 600.0,
 dt    = 0.01,
 tpr   = 10.0,
 tdump = 50.0
/

&model
/

&mesh
  nx = 100,
  nx = 100,
  nz = 100
/

&particles
  nsolid = 1,
  diameter = 50.
  density = 1500.
  sphericity = 1.0
  viscosity = 0.5
  specific_heat = 1.2D3
  thermal_conductivity = 2.0D0
/

&numeric
  rungekut = 1,
  muscl = 0,
  omega = 1.00
/

'ROUGHNESS'

'MESH'

 200.00D0 200.00D0 196.39D0 192.02D0 187.64D0
```

```
183.26D0 178.88D0 174.50D0 170.12D0 165.74D0
161.36D0 156.98D0 152.61D0 148.23D0 143.85D0
139.47D0 135.09D0 130.71D0 126.33D0 121.95D0
117.58D0 113.20D0 108.82D0 104.44D0 100.06D0
 95.68D0  91.30D0  86.92D0  82.55D0  78.17D0
 73.79D0  69.41D0  65.03D0  60.65D0  56.27D0
 51.89D0  47.52D0  43.14D0  38.76D0  34.38D0
 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
 30.00D0  30.00D0  30.00D0  30.00D0  23.17D0
 23.17D0  30.00D0  30.00D0  30.00D0  30.00D0
 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
 34.38D0  38.76D0  43.14D0  47.52D0  51.89D0
 56.27D0  60.65D0  65.03D0  69.41D0  73.79D0
 78.17D0  82.55D0  86.92D0  91.30D0  95.68D0
100.06D0 104.44D0 108.82D0 113.20D0 117.58D0
121.95D0 126.33D0 130.71D0 135.09D0 139.47D0
143.85D0 148.23D0 152.61D0 156.98D0 161.36D0
165.74D0 170.12D0 174.50D0 178.88D0 183.26D0
187.64D0 192.02D0 196.39D0 200.00D0 200.00D0

200.00D0 200.00D0 196.39D0 192.02D0 187.64D0
183.26D0 178.88D0 174.50D0 170.12D0 165.74D0
161.36D0 156.98D0 152.61D0 148.23D0 143.85D0
139.47D0 135.09D0 130.71D0 126.33D0 121.95D0
117.58D0 113.20D0 108.82D0 104.44D0 100.06D0
 95.68D0  91.30D0  86.92D0  82.55D0  78.17D0
 73.79D0  69.41D0  65.03D0  60.65D0  56.27D0
 51.89D0  47.52D0  43.14D0  38.76D0  34.38D0
 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
 30.00D0  30.00D0  30.00D0  30.00D0  23.17D0
 23.17D0  30.00D0  30.00D0  30.00D0  30.00D0
 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
 34.38D0  38.76D0  43.14D0  47.52D0  51.89D0
 56.27D0  60.65D0  65.03D0  69.41D0  73.79D0
 78.17D0  82.55D0  86.92D0  91.30D0  95.68D0
100.06D0 104.44D0 108.82D0 113.20D0 117.58D0
121.95D0 126.33D0 130.71D0 135.09D0 139.47D0
143.85D0 148.23D0 152.61D0 156.98D0 161.36D0
165.74D0 170.12D0 174.50D0 178.88D0 183.26D0
187.64D0 192.02D0 196.39D0 200.00D0 200.00D0

 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
 30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
```

```
  30.00D0  30.00D0  30.00D0  30.00D0  30.00D0
  32.95D0  35.90D0  38.85D0  41.80D0  44.74D0
  47.69D0  50.64D0  53.59D0  56.54D0  59.49D0
  62.44D0  65.39D0  68.34D0  71.29D0  74.23D0
  77.18D0  80.13D0  83.08D0  86.03D0  88.98D0
  91.93D0  94.88D0  97.83D0 100.78D0 103.72D0
 106.67D0 109.62D0 112.57D0 115.52D0 118.47D0
 121.42D0 124.37D0 127.32D0 130.27D0 133.21D0
 136.16D0 139.11D0 142.06D0 145.01D0 147.96D0
 150.91D0 153.86D0 156.81D0 159.76D0 162.70D0
 165.65D0 168.60D0 171.55D0 174.50D0 177.45D0
 180.40D0 183.35D0 186.30D0 189.24D0 192.19D0
 195.14D0 198.09D0 200.00D0 200.00D0 200.00D0
 200.00D0 200.00D0 200.00D0 200.00D0 200.00D0
 200.00D0 200.00D0 200.00D0 200.00D0 200.00D0

'FIXED_FLOWS'
7
3, 1  , 100, 1 , 100, 1, 1
5, 49 , 52, 49, 52, 1, 1
0.0, 0.0, 110.00, 1.01325D5, 0.999, 900.0
0.0, 0.0, 110.00, 0.001, 900.0
0.D0, 0.D0, 0.D0, 0.D0, 1.D0, 0.D0, 0.D0
4, 1  , 1  , 1  , 100, 2  , 100
4, 100, 100, 1  , 100, 2  , 100
4, 1  , 100, 1  , 1  , 2  , 100
4, 1  , 100, 100, 100, 2  , 100
4, 1  , 100, 1  , 100, 100, 100

'INITIAL_CONDITIONS'
0.0, 0.0, 1.01325D5, 1. 0.6413, 288.15
0.0, 0.0
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0
```

5 Running PDAC

5.1 Getting started: what is needed

Before running PDAC the following elements are needed:

- The PDAC executable *pdac.x*
- The PDAC parameter file *pdac.dat*.

Here it is assumed that you either succeded in compiling the source code or that you got a precompiled executable suited for your architecture. Be sure to have the authorization for running the executable on the directory where it is installed. For further informations on compiling and on code portability, see Sect. 6.

PDAC runs on a variety of serial and parallel platforms. While it is trivial to launch a serial program, the execution of a parallel program depends on a platform-specific library, such as MPI, that allows to launch copies of the executable on other nodes and to provide access to a high performance network such as Myrinet (if available) or to a Ethernet connection.

On all parallel systems, either cluster of workstations or parallel supercomputers, PDAC execution is relied to the local implementation of MPI for data communications, and to standard system tools such as `mpirun` to launch jobs. Therefore a working installation of MPI is a prerequisite in order to run PDAC in parallel. The PDAC serial binaries can nevertheless be run directly (this is known as *standalone* mode) for single process runs on most platforms. Since different versions of MPI libraries are very often incompatible, you will likely need to recompile PDAC to run it in parallel. For serial runs the provided non-MPI binaries should still work.

5.2 Individual Unix Workstations

In this section we will explain how to run PDAC on your own workstation without a queuing system like OpenPBS or Loadleveler (see below on this chapter). On individual Unix workstations the use of the serial PDAC is quite easy, basically it behaves like any other Unix process running on your machine. To run PDAC then follow the steps:

- create a running directory

  ```
  > mkdir sample_run
  ```

- copy executable (*pdac.x*) and input file (*pdac.dat*) to the running directory

  ```
  > cp pdac.x pdac.dat sample_run
  ```

- run *pdac.x* (possibly in background)

  ```
  > pdac.x &
  ```

Nonetheless you are the only user of your workstation, we suggest you to breakup the simulation in shorter partial simulations (few days maximum each) and use the restart capability of PDAC (see Sect. 2.1.2). In this way you reduce the probability of data/work loss due to unforseenable events (system crash, blackout, ecc...). When restarting a simulation, remember to set the restart mode to “restart” and to save the *pdac.res* file, in case that something goes wrong in the new simulation step.

For multiprocessor workstations, if you have a working MPI library, you can compile PDAC in parallel mode to use all processors of the workstation. In this case, to run PDAC in parallel you should use an MPI loader (usually distributed with the MPI library), i.e. if your MPI library is MPICH you can run PDAC using the commad:

```
> mpirun -np 2 pdac.x
```

where the command parameter “-np 2” specifies that you want to run on two processors.

5.3 Server and Parallel Supercomputers

In this section we explain how to run PDAC on Unix servers and parallel supercomputers giving you some practical example for most popular architectures (at the time of writing). The main differences you will find, with respect to running on individual workstations, is represented by the presence of a queuing system that manage the user requests. Usually, you should also consider the limits the system administrator has set to your account, like disk space, memory or available running queues. If this limits are not taken into account, this could cause the simulation not to run or complete. Below, as anticipated, we will give you few examples on how to run on popular servers and supercomputers, but remember that these systems are highly customizable and what is presented here should be probably modified to run on other supercomputers.

5.3.1 IBM RS/6000 servers and parallel supercomputers

On most of IBM servers and RS/6000 supercomputers, to run a simulation you should write a job script to be submitted to the Loadleveler which is the IBM queuing system. Many systems allows you to run MPI program in interactive mode for testing and debugging, through the POE program. The options and environment variables for POE are various and arcane, so you should consult your local documentation for recommended settings. As an example for interactive mode runs on the RS/6000 supercomputer (Sp4) installed at CINECA at the shell prompt you should give the command:

```
> poe ./pdac.x -procs 4 -nodes 1
```

to run on a single node using 4 processors. For long runs it is required that you write a job script and submit it to the Loadleveler. Here is an example.

```
#!/bin/ksh
#@ job_type = parallel
#@ output = job.out
#@ error = job.err
#@ notification = never
#@ checkpoint = no
#@ restart = no
#@ wall_clock_limit =6:00:00
#@ resources = ConsumableCpus(1) ConsumableMemory(1024 mb)
#@ class = parallel
#@ network.MPI = csss,shared,US
#@ total_tasks = 8
#@ blocking = unlimited
#@ queue

cp pdac.dat /scratch/myrundir/pdac.dat
cp pdac.x /scratch/myrundir/pdac.x
cd /scratch/myrundir/
./pdac.x
```

The above script asks for 8 processors to run 8 MPI tasks with a limit of 1024Mbyte each for 6 hours on the parallel queue. Finally you should submit the job (suppose the file name is *myjobfile*):

```
> llsubmit myjobfile
```

5.3.2 Linux Clusters (Beowulf)

The most common way to run a program on a linux cluster is through the use of the combination of OpenPBS, MAUI scheduler and mpirun/mpiexec MPI loaders . For this kind of machines you could find an almost infinite number of environments and system softwares, therefore keep the example below as indicative, quite probably it would not work on other systems. As for other servers, you need to write a job script with the request for the queuing system and with the command you want to execute.

```
#!/bin/sh
#PBS -l nodes=16:ppn=2,walltime=6:00:00

cp pdac.dat /myrundir
cp pdac.x /myrundir
cd /myrundir
mpiexec -no-shmem -np 32 ./pdac.x
```

with this script we are asking for 16 nodes with 2 processors per node and 6 hours of execution time in the queue. Then you need to submit the job to the queuing system; for OpenPBS the command is:

```
> qsub myjobscript
```

where we have supposed that the filename of the script is *myjobscript*.

5.3.3 Compaq AlphaServer SC

If your machine is a Quadrics interconnect you should use the Elan communication subsystem commands to run your parallel applications or jobs. The main command is the *prun* command, that is similar to mpirun of MPICH, but with prun you could run both job scripts and interactive commands. Here is an example :

```
> prun -n 8 ./pdac.x
```

this command will run *pdac.x* on 8 processors, as soon as they will be free, infact if there are other job (or commands) already running your command will be queued. There are several additional options taht should be consulted on the specific documentation.

5.3.4 SGI Origin systems

SGI Origin systems are shared memory parallel machines with real single system image framework. In other words the user can see or access all the processors directly from the login environment. Differently from other architectures, there is no distinction between login and execution nodes. This has some benefits and drawbacks: benefits come from the possibility to use the whole machine as a multiprocessors workstation with high number of processors, the drawbacks are related to the possibility that a different job running at the same time on the machine interferes with your job. In general, you can run parallel applications in two different ways, using a shell command like in your own workstation and using a queuing system, which is usually NQS. To run from the shell prompt use mpirun, i.e.:

```
> mpirun -np 8 ./pdac.x
```

This will execute *pdac.x* with 8 MPI task, that not necessarily goes to different processors, it depends on the system load. Although this way of executing parallel applications is very friendly, it could cause a lot of interferences with system activities and other applications, with the result of slowing down the system. Then, quite often, the system administrator set severe limits to this way of running, mainly on the number of real available processors (no matter how many MPI tasks you are asking for), execution time and memory usage. Therefore on many systems to run large applications you need to interface with the NQS queuing system, so that you have to write a script with the requests for processors and memory and then submit the script to NQS. An example of job script for NQS is the following:

```
#!/bin/sh
#QSUB -l mpp_p=8
#QSUB -l p_mpp_t=2:00:00

cp pdac.dat /myrundir
cp pdac.x /myrundir
cd /myrundir
mpirun -np 8 ./pdac.x
```

with this script you ask for 8 processors for 2 hours, then you could submit the script with the command (suppose its filename is *myjobfile*):

```
> qsub myjobfile
```

5.4 Grid environment

Few simulations on a Grid environment (such as Condor) have been performed. Some attention must be payed to the settings of MPI environment in Condor, to the need of static compiling and to possible errors relied to the read-write opening of the restart file. Further study will be devoted in the next future to the possibility of running PDAC on a distributed system.

5.5 Memory usage considerations

The use of parallel supercomputers or PC clusters is mandatory not only for the purpose of decreasing simulation time, but also since memory occupation can be too high for a single workstation. As a rough estimate, consider that, for a “standard” 3D simulation (with 2 solid phases) PDAC requires a memory occupation of about 1.5 KBytes/cell.

5.6 Monitoring parallel speedup and scaling

Although PDAC is designed to be a scalable program, particularly for large simulations (100,000 cells or more), increasing the number of processors above a certain threshold will provide little or no extra performance. This is known as “speedup saturation” and is due to the increase of the ratio between the communication and computation times. If you have access to a parallel machine you should measure PDAC s parallel speedup for a variety of processor counts when running your particular simulation. The easiest and most accurate way to do this is to look at the information contained in the timing report in the Log files for the different routines. On many parallel architectures fine-tuned performance monitoring can be performed through appropriate programs.

6 PDAC Availability and Installation

PDAC distribution is governed by the License reported at the beginning of this manual.

PDAC 2.0 is based on the MPI message passing interface (`http://www.mpi-forum.org/`) and on a customized communication layer which have been ported to a wide variety of parallel platforms. This section describes how to obtain and install PDAC 2.0.

6.1 How to obtain PDAC

PDAC may be downloaded from http://www.pi.ingv.it/PDAC. You will be required to provide minimal registration information and agree to a license before receiving access to the software. Both source and binary distributions are available.

6.2 Platforms on which PDAC will currently run

PDAC should be portable to any serial platform with a Fortran90 and C compilers and to any parallel platform with the above compilers and the MPI library. Precompiled PDAC 2.0 binaries are available for the following platforms:

- Linux on Intel
- Mac OS X (also called Darwin) on PowerPC processors
- AIX on RS/6000 processors
- HP-UX on PA-RISC processors
- Solaris on Sparc processors (with and without MPI)
- Tru64 Unix on Alpha processors (with and without MPI)
- IBM RS/6000 SP (using MPI)
- Compaq AlphaServer SC (using the Quadrics Elan library)
- SGI Origin 2000 (with and without MPI)

6.3 Compiling PDAC

As mentionead above, we provide binaries for all platforms to which PDAC has been ported. However, it is recommended to recompile the code on your system, since hardware is changing rapidly and most probably your native Fortran90 compiler will optimize the code further. On Linux parallel platform with MPI, recompiling is in practice mandatory, since there are too many possible combinations of MPI libraries and networking hardware for which the executables are incompatible. Recompiling is also required if you wish to add or modify features of PDAC.

6.3.1 Directory structure

The PDAC source code is distributed with a directory structure with two levels, a single main directory named <*EXPLORIS*> (in the official distribution), and several sub-directories. In what follows the different directories are listed and their content commented.

- directory <*EXPLORIS*>: this is the main directory and contains the Fortran90 source files of the PDAC main subroutines. This directory contains also the *Makefile* and the *Machine* file (see below).
- directory <*EXPLORIS/comm*>: this subdirectory contains the PDAC communication layer, interfaced to MPI and few wrappers for system dependent features. In general, all low level subroutines that need preprocessing are in this directory.
- directory <*EXPLORIS/doc*>: this directory contains the PDAC documentation and this manual.
- directory <*EXPLORIS/examples*>: this directory contains several input files examples.
- directory <*EXPLORIS/utility*>: this directory contains few utility scripts and programs to help compiling and porting the code.

6.3.2 *Machine* file and *Makefile*

PDAC source code comes with a makefile (named *Makefile* in the distribution) that takes care of the compilation of PDAC itself. The *Makefile* file contains general commands that should be quite general and platform independent. There are other *Makefile* files in the subdirectories to compile libraries, utilities and documentations. All *Makefile* files include a configuration file named *Machine*, which contains all the system dependent parameters for the compilation, like the compiler name and the compilations flags. In the main directory there are several *Machine.arch* files already prepared for a number of existing architectures. The *Machine.arch* files distributed with the source code are configured and tested on the corresponding architecture, although your system could have some particular feature which requires some change in the *Machine.arch* file, in this case you should edit the file to make the proper changes. The main *Makefile* file includes also a hidden file (*.dependencies*) containing all the dependencies among the source Fortran90 files. The PDAC source code is infact written using Fortran90 modules and source files should be compiled in the proper order, determined by the dependencies among modules. The file *.dependencies* is generated by the *shdep* script distributed togheter with the code (see next section for more details).

6.3.3 Making dependencies and compiling

All steps required to compile the code are now listed and commented.

- Edit *Machine* file

  The very first thing you should do to compile PDAC is to choose the *Machine* file for your architecture among those distributed with the source code and named *Machine.arch*, where the "arch" suffix vary on the list of the supported architectures. For example, if your machine is an IBM Sp4, you should copy the file *Machine.sp4* onto the *Machine* file. If there isn't the *Machine.arch* for your architecture, choose the most close among those present and edit it to set the compiling parameters to the proper values. Once you have find your *Machine* file you should copy it into the file named *Machine* with the following command:

```
>
> cp Machine.arch Machine
>
```

Then you can edit *Machine* to change compiling parameters, such as the compiler options or the executable type (serial or parallel). For a serial executable you should unset the -D__MPI precompiler macro subsituting it with -D__SERIAL, and substitute the parallel compilers with the scalar one, changing the variables FC, MPIFC and LINKER in the *Machine* file.

- Build the *.dependencies* file

  Before compiling PDAC executable you shold build the dependencies file (.dependencies). To do this, you should move to the directory EXPLORIS/utility and type

  ```
  > make all
  ```

  then, supposing that you are in the EXPLORIS directory, the exact sequence of command is:

  ```
  > cd utility
  > make all
  > cd ..
  ```

- Compiling *pdac.x* and *pp.x*

  At this point everything is ready to compile the PDAC executable (*pdac.x*) and the post processing executable (*pp.x*), from the EXPLORIS directory type:

  ```
  > make pdac.x
  >
  > make pp.x
  ```

- Compiling this manual

  To compile the manual, you must have installed a Tex packages with standard LaTeX extensions support, then type the following commands:

  ```
  >
  > cd doc
  > latex pdac_ug.tex
  > latex pdac_rm.tex
  > dvips pdac_ug.dvi -o pdac_ug.ps
  > dvips pdac_rm.dvi -o pdac_rm.ps
  ```

6.4 Documentation

All available PDAC documentation is available for downloading without registration via the PDAC web site http://www.pi.ingv.it/PDAC.