**The PENCIL CODE:** multi-purpose and multi-user maintained

[http://www.nordita.org/~brandenb/talks/misc/PencilCode04.htm](http://www.nordita.org/~brandenb/talks/misc/PencilCode04.htm)

*Figure 1:* Check-in patterns as a function of time for different subroutines. The different users are marked by different symbols and different colors.
### Contributors to the code

(in inverse alphabetical order according to their user name)

An up to date list of Pencil Code contributors can be found at [GitHub](https://github.com).

<table>
<thead>
<tr>
<th>User</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>wladimir.lyra</td>
<td>California State University/JPL</td>
</tr>
<tr>
<td>weezy</td>
<td>University of Newcastle</td>
</tr>
<tr>
<td>wdobler</td>
<td>Potsdam</td>
</tr>
<tr>
<td>vpariev</td>
<td>University of Rochester</td>
</tr>
<tr>
<td>torkel</td>
<td>Chalmers University</td>
</tr>
<tr>
<td>tavo.buk</td>
<td>Stanford University</td>
</tr>
<tr>
<td>thomas.gastine</td>
<td>MPI for Solar System Research</td>
</tr>
<tr>
<td>theine</td>
<td>IAS Princeton</td>
</tr>
<tr>
<td>tarek</td>
<td>University of Trondheim</td>
</tr>
<tr>
<td>sven.bingert</td>
<td>MPI for Solar System Research</td>
</tr>
<tr>
<td>steveb</td>
<td>UCLA</td>
</tr>
<tr>
<td>snod</td>
<td>University of Newcastle</td>
</tr>
<tr>
<td>pkapyla</td>
<td>University of Helsinki</td>
</tr>
<tr>
<td>nils.e.haugen</td>
<td>SINTEF</td>
</tr>
<tr>
<td>ngors</td>
<td>University of Newcastle</td>
</tr>
<tr>
<td>NBabkovskaia</td>
<td>University of Helsinki</td>
</tr>
<tr>
<td>mcmeinhardt</td>
<td>University of Helsinki</td>
</tr>
<tr>
<td>mkorpi</td>
<td>University of Helsinki</td>
</tr>
<tr>
<td>miikkavaisala</td>
<td>University of Helsinki</td>
</tr>
<tr>
<td>mee</td>
<td>University of Newcastle</td>
</tr>
<tr>
<td>mcmillan</td>
<td>York University, Toronto</td>
</tr>
<tr>
<td>mattias</td>
<td>formerly at Nordita</td>
</tr>
<tr>
<td>koenkemel</td>
<td>Nordita, Stockholm</td>
</tr>
<tr>
<td>karlsson</td>
<td>Nordita</td>
</tr>
<tr>
<td>joishi</td>
<td>Kavli Institute for Particle Physics, California</td>
</tr>
<tr>
<td>joern.warnecke</td>
<td>MPI for Solar System Research, Lund</td>
</tr>
<tr>
<td>Iomsn1</td>
<td>University of Dundee, Dundee</td>
</tr>
<tr>
<td>fadiesis</td>
<td>Nordita, Stockholm</td>
</tr>
<tr>
<td>dorch</td>
<td>University of Copenhagen</td>
</tr>
<tr>
<td>boris.dintrans</td>
<td>Observatoire Midi-Pyrénées, Toulouse</td>
</tr>
<tr>
<td>dhruba.mitra</td>
<td>Nordita, Stockholm</td>
</tr>
<tr>
<td>ccyang</td>
<td>Lund Observatory</td>
</tr>
<tr>
<td>christe</td>
<td>University of Uppsala</td>
</tr>
<tr>
<td>Bourdin.KIS</td>
<td>MPI for Solar System Research</td>
</tr>
<tr>
<td>AxelBrandenburg</td>
<td>Nordita</td>
</tr>
<tr>
<td>apichat</td>
<td>University of Newcastle</td>
</tr>
<tr>
<td>amjed</td>
<td>University of Oldenburg</td>
</tr>
<tr>
<td>alex.i.hubbard</td>
<td>Am. Museum Nat. History</td>
</tr>
<tr>
<td>michiel.lambrechts</td>
<td>Lund Observatory, Lund University</td>
</tr>
<tr>
<td>anders</td>
<td>Lund Observatory</td>
</tr>
<tr>
<td>mppiyali</td>
<td>University of Oslo</td>
</tr>
</tbody>
</table>

Copyright © 2001–2017 Wolfgang Dobler & Axel Brandenburg

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.
Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.
License agreement and giving credit

The content of all files under :pserver:$USER@svn.nordita.org:/var/cvs/brandenb are under the GNU General Public License (http://www.gnu.org/licenses/gpl.html).

We, the PENCIL CODE community, ask that in publications and presentations the use of the code (or parts of it) be acknowledged with reference to the web site http://www.nordita.org/software/pencil-code/ or (equivalently) to. https://github.com/pencil-code/pencil-code As a courtesy to the people involved in developing particularly important parts of the program (use svn annotate src/*.f90 to find out who did what!) we suggest to give appropriate reference to one or several of the following (or other appropriate) papers (listed here in temporal order):


This list is not always up-to-date. We therefore ask the developers to check in new relevant papers, avoiding however redundancies.
Foreword

This code was originally developed at the Turbulence Summer School of the Helmholtz Institute in Potsdam (2001). While some SPH and PPM codes for hydrodynamics and magnetohydrodynamics are publicly available, this does not generally seem to be the case for higher order finite-difference or spectral codes. Having been approached by people interested in using our code, we decided to make it as flexible as possible and as user-friendly as seems reasonable, and to put it onto a public CVS repository. Since 21 September 2008 it is distributed via https://github.com/pencil-code/pencil-code. The code can certainly not be treated as a black box (no code can), and in order to solve a new problem in an optimal way, users will need to find their own optimal set of parameters. In particular, you need to be careful in choosing the right values of viscosity, magnetic diffusivity, and radiative conductivity.

The PENCIL CODE is primarily designed to deal with weakly compressible turbulent flows, which is why we use high-order first and second derivatives. To achieve good parallelization, we use explicit (as opposed to compact) finite differences. Typical scientific targets include driven MHD turbulence in a periodic box, convection in a slab with non-periodic upper and lower boundaries, a convective star embedded in a fully nonperiodic box, accretion disc turbulence in the shearing sheet approximation, etc. Furthermore, nonlocal radiation transport, inertial particles, dust coagulation, self-gravity, chemical reaction networks, and several other physical components are installed, but this number increases steadily. In addition to Cartesian coordinates, the code can also deal with spherical and cylindrical polar coordinates.

Magnetic fields are implemented in terms of the magnetic vector potential to ensure that the field remains solenoidal (divergence-free). At the same time, having the magnetic vector potential readily available is a big advantage if one wants to monitor the magnetic helicity, for example. The code is therefore particularly well suited for all kinds of dynamo problems.

The code is normally non-conservative; thus, conserved quantities should only be conserved up to the discretization error of the scheme (not to machine accuracy). There is no guarantee that a conservative code is more accurate with respect to quantities that are not explicitly conserved, such as entropy. Another important quantity that is (to our knowledge) not strictly conserved by ordinary flux conserving schemes is magnetic helicity.

There are currently no plans to implement adaptive mesh refinement into the code, which would cause major technical complications. Given that turbulence is generically space-filling, local refinement to smaller scales would often not be very useful anyway. On the other hand, in some geometries turbulence may well be confined to certain regions in space, so one could indeed gain by solving the outer regions with fewer points.

In order to be cache-efficient, we solve the equations along pencils in the $x$ direction. One very convenient side-effect is that auxiliary and derived variables use very little memory, as they are only ever defined on one pencil. The domain can be tiled in the $y$ and $z$ directions. On multiprocessor computers, the code can use MPI (Message Passing Interface) calls to communicate between processors. An easy switching mechanism allows the user to run the code on a machine without MPI libraries (e.g. a notebook computer). Ghost zones are used to implement boundary conditions on physical and processor boundaries.
A high level of flexibility is achieved by encapsulating individual physical processes and variables in individual modules, which can be switched on or off in the file ‘Makefile.local’ in the local ‘src’ directory. This approach avoids the use of difficult-to-read preprocessor directives, at the price of requiring one dummy module for each physics module. For nonmagnetic hydrodynamics, for example, one will use the module ‘nomagnetic.f90’ and specifies

MAGNETIC = nomagnetic

in ‘Makefile.local’, while for MHD simulations, ‘magnetic.f90’ will be used:

MAGNETIC = magnetic

Note that the term module as used here is only loosely related to Fortran modules: both ‘magnetic.f90’ and ‘nomagnetic.f90’ define an F90 module named Magnetic — this is the basis of the switching mechanism we are using.

Input parameters (which are set in the files ‘start.in’, ‘run.in’) can be changed without recompilation. Furthermore, one can change the list of variables for monitoring (diagnostic) output on the fly, and there are mechanisms for making the code reload new parameters or exit gracefully at runtime. You may want to check for correctness of these files with the command pc_configtest.

The requirements for using the Pencil-MPI code are modest: you can use it on any Linux or Unix system with a F95 and C compiler suite, like GNU gcc and gfortran, together with the shell CSH, and the Perl interpreter are mandatory requirements.

Although the PENCIL CODE is mainly designed to run on supercomputers, more than 50% of the users run their code also on Macs, and the other half uses either directly Linux on their laptops or they use VirtualBox on their Windows machine on which they install Ubuntu Linux. If you have IDL as well, you will be able to visualize the results (a number of sample procedures are provided), but other tools such as Python, DX (OpenDX, data explorer) can also be used and some relevant tools and routines come with the PENCIL CODE.

If you want to make creative use of the code, this manual will contain far too little information. Its major aim is to give you an idea of the way the code is organized, so you can more efficiently read the source code, which contains a reasonable amount of comments. You might want to read through the various sample directories that are checked in. Choose one that is closest to your application and start modifying. For further enhancements that you may want to add to the code, you can take as an example the lines in the code that deal with related variables, functions, diagnostics, equations etc., which have already been implemented. Just remember: grep is one of your best friends when you want to understand how certain variables or functions are used in the code.

We will be happy to include user-supplied changes and updates to the code in future releases and welcome any feedback.
Acknowledgments

Many people have contributed in different ways to the development of this code. We thank first of all Åke Nordlund (Copenhagen Observatory) and Bob Stein (University of Michigan) who introduced us to the idea of using high-order schemes in compressible flows and who taught us a lot about simulations in general.

The calculation of the power spectra, structure functions, the remeshing procedures, routines for changing the number of processors, as well as the shearing sheet approximation and the flux-limited diffusion approximation for radiative transfer were implemented by Nils Erland L. Haugen (University of Trondheim). Tobi Heinemann added the long characteristics method for radiative transfer as well as hydrogen ionization. He also added and/or improved shock diffusion for other variables and improved the resulting timestep control. Anders Johansen, Wladimir (Wlad) Lyra, and Jeff Oishi contributed to the implementation of the dust equations (which now comprises an array of different components). Antony (Tony) Mee (University of Newcastle) implemented shock viscosity and added the interstellar module together with Graeme R. Sarson (also University of Newcastle), who also implemented the geodynamo set-up together with David McMillan (currently also at the University of Newcastle). Tony also included a method for outputting auxiliary variables and enhanced the overall functionality of the code and related idl and dx procedures. He also added, together with Andrew Snodin, the evolution equations for the cosmic ray energy density. Vladimir Pariev (University of Rochester) contributed to the development and testing of the potential field boundary condition at an early stage. The implementation of spherical and cylindrical coordinates is due to Dhruvabadiya (Dhruba) Mitra and Wladimir Lyra. Wlad also implemented the global set-up for protoplanetary disks (as opposed to the local shearing sheet formalism). He also added a $N$-body code (based on the particle module coded by Anders Johansen and Tony), and implemented the coupled evolution equations of neutrals and ions for two-fluid models of ambipolar diffusion. Boris Dintrans is in charge of implementing the anelastic and Boussinesq modules. Philippe-A. Bourdin implemented HDF5 support and wrote the optional IO-modules for high-performance computing featuring various communication strategies. He also contributed to the solar-corona module and worked on the IDL GUI, including the IDL routines for reading and working with large amounts of data. Again, this list contains other recent items that are not yet fully documented and acknowledged.

Use of the PPARC supported supercomputers in St Andrews (Mhd) and Leicester (Ukaff) is acknowledged. We also acknowledge the Danish Center for Scientific Computing for granting time on Horseshoe, which is a 512+140 processor Beowulf cluster in Odense (Horseshoe).
# Contents

1 Using the PENCIL CODE  

1.1 System requirements  

2 Obtaining the code  
2.1 Obtaining the code via git or svn  
2.2 Updating via svn or git  
2.3 Getting the last validated version  
2.4 Getting older versions  

3 Getting started  
3.1 Setup  
3.1.1 Environment settings  
3.1.2 Linking scripts and source files  
3.1.3 Adapting `Makefile.src`  
3.1.4 Running `make`  
3.1.5 Choosing a data directory  
3.1.6 Running the code  
3.2 Further tests  

4 Code structure  
4.1 Directory tree  
4.2 Basic concepts  
4.2.1 Data access in pencils  
4.2.2 Modularity  
4.3 Files in the run directories  
4.3.1 `start.in`, `run.in`, `print.in`  
4.3.2 `datadir.in`  
4.3.3 `reference.out`  
4.3.4 `start.csh`, `run.csh`, `getconf.csh` [obsolete; see Sect. 5.1]  
4.3.5 `src/`  
4.3.6 `data/`  

5 Using the code  
5.1 Configuring the code to compile and run on your computer  
5.1.1 Locating the configuration file  
5.1.2 Structure of configuration files  
5.1.3 Compiling the code  
5.1.4 Running the code  
5.1.5 Testing the code  
5.2 Adapting `Makefile.src` [obsolete; see Sect. 5.1]  
5.3 Changing the resolution  
5.4 Using a non-equidistant grid  
5.5 Diagnostic output  
5.6 Data files  
5.6.1 Snapshot files  
5.7 Video files and slices
5.8 Averages

5.8.1 One-dimensional output averaged in two dimensions

5.8.2 Two-dimensional output averaged in one dimension

5.8.3 Azimuthal averages

5.8.4 Time averages

5.9 Helper scripts

5.10 RELOAD and STOP files

5.11 RERUN and NEWDIR files

5.12 Start and run parameters

5.13 Physical units

5.14 Minimum amount of viscosity

5.15 The time step

5.15.1 The usual RK-2N time step

5.15.2 The Runge-Kutta-Fehlberg time step

5.16 Boundary conditions

5.16.1 Where to specify boundary conditions

5.16.2 How to specify boundary conditions

5.17 Restarting a simulation

5.18 One- and two-dimensional runs

5.19 Visualization

5.19.1 Gnuplot

5.19.2 Data explorer

5.19.3 GDL

5.19.4 IDL

5.19.5 Python

5.20 Running on multi-processor computers

5.20.1 How to run a sample problem in parallel

5.20.2 Hierarchical networks (e.g. on Beowulf clusters)

5.20.3 Extra workload caused by the ghost zones

5.21 Running in double-precision

5.22 Power spectrum

5.23 Structure functions

5.24 Particles

5.24.1 Particles in parallel

5.24.2 Large number of particles

5.24.3 Random number generator

5.25 Non-cartesian coordinate systems

6 The equations

6.1 Continuity equation

6.2 Equation of motion

6.3 Induction equation

6.4 Entropy equation

6.4.1 Viscous heating

6.4.2 Alternative description

6.5 Transport equation for a passive scalar

6.6 Bulk viscosity

6.6.1 Shock viscosity

6.7 Equation of state

6.8 Ionization
6.8.1 Ambipolar diffusion .................................................. 65
6.9 Radiative transfer .......................................................... 66
6.10 Self-gravity ................................................................. 67
6.11 Incompressible and anelastic equations .............................. 67
6.12 Dust equations ............................................................. 67
6.13 Cosmic ray pressure in diffusion approximation ................... 68
6.14 Particles ................................................................. 69
   6.14.1 Tracer particles ................................................... 69
   6.14.2 Dust particles ..................................................... 69
6.15 N-body solver ............................................................ 70
6.16 Test-field equations ..................................................... 71
6.17 Gravitational wave equations .......................................... 71

7 Troubleshooting / Frequently Asked Questions .......................... 75
7.1 Download and setup ...................................................... 75
   7.1.1 Download forbidden ................................................. 75
   7.1.2 Shell gives error message when sourcing 'sourceme.X' ....... 75
7.2 Compilation ............................................................... 76
   7.2.1 Problems compiling syscalls ..................................... 76
   7.2.2 Unable to open include file: chemistry.h ................... 76
   7.2.3 Compiling with ifc under Linux .................................. 76
   7.2.4 Segmentation fault with ifort 8.0 under Linux ............ 77
   7.2.5 The underscore problem: linking with MPI .................. 77
   7.2.6 Compilation stops with the cryptic error message .......... 77
   7.2.7 The code doesn’t compile ....................................... 78
   7.2.8 Some samples don’t even compile .............................. 78
   7.2.9 Internal compiler error with Compaq/Dec F90 ............... 79
   7.2.10 Assertion failure under SunOS .................................. 79
   7.2.11 After some dirty tricks I got pencil code to compile with MPI ...... 80
   7.2.12 Error: Symbol 'mpi_comm_world' at (1) has no IMPLICIT type .................. 80
   7.2.13 Error: Can’t open included file 'mpif.h' ................... 81
7.3 Pencil check ............................................................ 81
   7.3.1 The pencil check complains for no reason ................... 81
   7.3.2 The pencil check reports MISSING PENCILS and quits .... 81
   7.3.3 The pencil check reports unnecessary pencils ............. 81
   7.3.4 The pencil check reports that most or all pencils are missing .... 81
   7.3.5 Running the pencil check triggers mathematical errors in the code .......... 82
   7.3.6 The pencil check still complains .............................. 82
   7.3.7 The pencil check is annoying so I turned it off ......... 82
7.4 Running ............................................................... 82
   7.4.1 Periodic boundary conditions in ’start.x’ ................... 82
   7.4.2 csh problem? ...................................................... 82
   7.4.3 ‘run.csh’ doesn’t work ........................................... 83
   7.4.4 Code crashes after restarting .................................. 83
   7.4.5 auto-test gone mad...? .......................................... 83
   7.4.6 Can I restart with a different number of cpus? .......... 84
   7.4.7 Can I restart with a different number of cpus? .......... 84
   7.4.8 fft xyz.parallel.3D: nygrid needs to be an integer multiple... 84
   7.4.9 Unit-agnostic calculations? .................................... 85
7.5 Visualization .......................................................... 86
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.5.1</td>
<td>'start.pro' doesn't work</td>
<td>86</td>
</tr>
<tr>
<td>7.5.2</td>
<td>'start.pro' doesn't work</td>
<td>86</td>
</tr>
<tr>
<td>7.5.3</td>
<td>Something about tag name undefined</td>
<td>86</td>
</tr>
<tr>
<td>7.5.4</td>
<td>Something INC in start.pro</td>
<td>86</td>
</tr>
<tr>
<td>7.5.5</td>
<td>nl2idl problem when reading param2.nml</td>
<td>87</td>
</tr>
<tr>
<td>7.5.6</td>
<td>Spurious dots in the time series file</td>
<td>87</td>
</tr>
<tr>
<td>7.5.7</td>
<td>Problems with pc_varcontent.pro</td>
<td>87</td>
</tr>
<tr>
<td>7.6</td>
<td>General questions</td>
<td>88</td>
</tr>
<tr>
<td>7.6.1</td>
<td>“Installation” procedure</td>
<td>88</td>
</tr>
<tr>
<td>7.6.2</td>
<td>Small numbers in the code</td>
<td>88</td>
</tr>
<tr>
<td>7.6.3</td>
<td>Why do we need a /lphysics/ namelist in the first place?</td>
<td>89</td>
</tr>
<tr>
<td>7.6.4</td>
<td>Can I run the code on a Mac?</td>
<td>90</td>
</tr>
<tr>
<td>7.6.5</td>
<td>Pencil Code discussion forum</td>
<td>90</td>
</tr>
<tr>
<td>7.6.6</td>
<td>The manual</td>
<td>90</td>
</tr>
</tbody>
</table>

**II Programming the PENCIL CODE**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Understanding the code</td>
<td>95</td>
</tr>
<tr>
<td>8.1</td>
<td>Example: how is the continuity equation being solved?</td>
<td>95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Adapting the code</td>
<td>97</td>
</tr>
<tr>
<td>9.1</td>
<td>The PENCIL CODE coding standard</td>
<td>97</td>
</tr>
<tr>
<td>9.2</td>
<td>Adding new output diagnostics</td>
<td>98</td>
</tr>
<tr>
<td>9.3</td>
<td>The f-array</td>
<td>100</td>
</tr>
<tr>
<td>9.4</td>
<td>The df-array</td>
<td>100</td>
</tr>
<tr>
<td>9.5</td>
<td>The fp-array</td>
<td>101</td>
</tr>
<tr>
<td>9.6</td>
<td>The pencil case</td>
<td>101</td>
</tr>
<tr>
<td>9.6.1</td>
<td>Pencil check</td>
<td>102</td>
</tr>
<tr>
<td>9.6.2</td>
<td>Adding new pencils</td>
<td>103</td>
</tr>
<tr>
<td>9.7</td>
<td>Adding new physics: the Special module</td>
<td>103</td>
</tr>
<tr>
<td>9.8</td>
<td>Adding switchable modules</td>
<td>104</td>
</tr>
<tr>
<td>9.9</td>
<td>Adding your initial conditions: the InitialCondition module</td>
<td>104</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Testing the code</td>
<td>106</td>
</tr>
<tr>
<td>10.1</td>
<td>How to set up periodic tests</td>
<td>106</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Useful internals</td>
<td>108</td>
</tr>
<tr>
<td>11.1</td>
<td>Global variables</td>
<td>108</td>
</tr>
<tr>
<td>11.2</td>
<td>Subroutines and functions</td>
<td>108</td>
</tr>
</tbody>
</table>

**III Appendix**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Timings</td>
<td>111</td>
</tr>
<tr>
<td>A.1</td>
<td>Test case</td>
<td>118</td>
</tr>
<tr>
<td>A.2</td>
<td>Running the code</td>
<td>119</td>
</tr>
<tr>
<td>A.3</td>
<td>Triolith</td>
<td>119</td>
</tr>
<tr>
<td>A.4</td>
<td>Lindgren</td>
<td>119</td>
</tr>
<tr>
<td>B</td>
<td>Coding standard</td>
<td>122</td>
</tr>
</tbody>
</table>
Part I

Using the PENCIL Code

1 System requirements

To use the code, you will need the following:

1. Absolutely needed:
   - F95 compiler
   - C compiler

2. Used heavily (if you don’t have one of these, you will need to adjust many things manually):
   - a Unix/Linux-type system with make and csh
   - Perl (remember: if it doesn’t run Perl, it’s not a computer)

3. The following are dispensable, but enhance functionality in one way or the other:
   - an MPI implementation (for parallelization on multiprocessor systems)
   - DX alias OpenDX or data explorer (for 3-D visualization of results)
   - IDL (for visualization of results; the 7-minute demo license will do for many applications)
2 Obtaining the code

The code is now distributed via [https://github.com/pencil-code/pencil-code](https://github.com/pencil-code/pencil-code), where you can either download a tarball, or, preferably, download it via `svn` or `git`. In Iran and some other countries, GitHub is not currently available. To alleviate this problem, we have made a recent copy available on [http://www.nordita.org/software/pencil-code/](http://www.nordita.org/software/pencil-code/). If you want us to update this tarball, please contact us.

To ensure at least some level of stability of the `svn/git` versions, a set of test problems (listed in `$PENCIL_HOME/bin/auto-test`) are routinely tested. This includes all problems in `$PENCIL_HOME/samples`. See Sect. [10] for details.

### 2.1 Obtaining the code via `git` or `svn`

1. Many machines have `svn` installed (try `svn -v` or `which svn`). On Ubuntu, for example, `svn` comes under the package name `subversion`.

2. The code is now saved under Github, `git` can be obtained in Linux by typing `sudo apt-get install git`

3. Unless you are a privileged users with write access, you can download the code with the command

   ```
   git clone https://github.com/pencil-code/pencil-code.git
   ```

   or

   ```
   ```

   In order to push your changes to the repository, you have to ask the maintainer of pencil code for push access (to become a contributor), or put a pull request to the maintainer of the code.

   Be sure to run `auto-test` before you check anything back in again. It can be very annoying for someone else to figure out what's wrong, especially if you are just up to something else. At the very least, you should do

   ```
   pc_auto-test --level=0 --no-pencil-check -C
   ```

   This allows you to run just 2 of the most essential tests starting with all the no-modules and then most-modules.

### 2.2 Updating via `svn` or `git`

Independent of how you installed the code in the first place (from tarball or via `svn/git`), you can update your version using `svn/git`. If you have done nontrivial alterations to your version of the code, you ought to be careful about upgrading: although `svn/git` is an excellent tool for distributed programming, conflicts are quite possible, since many of us are going to touch many parts of the code while we develop it further. Thus, despite the
fact that the code is under svn/git, you should probably back up your important changes before upgrading.

Here is the upgrading procedure for git:

1. Perform a git update of the tree:
   
   \texttt{unix> git pull}

2. Fix any conflicts you encounter and make sure the examples in the directory `samples/` are still working.

Here is the upgrading procedure for svn:

1. Perform a svn update of the tree:

   \texttt{unix> pc_svnup}

2. Fix any conflicts you encounter and make sure the examples in the directory `samples/` are still working.

If you have made useful changes, please contact one of the (currently) 10 “Contributors” (listed under \url{https://github.com/pencil-code/pencil-code}) who can give you push or check-in permission. Be sure to have sufficient comments in the code and please follow our standard coding conventions explained in Section 9.1. There is also a script to check and fix the most common stylebreaks, pc_codingstyle.

### 2.3 Getting the last validated version

The script \texttt{pc_svnup} accepts arguments -val or -validated, which means that the current changes on a user's machine will be merged into the last working version. This way every user can be sure that any problems with the code must be due to the current changes done by this user since the last check-in.

Examples:

\texttt{unix> pc_svnup -src -s -validated}

brings all files in `$PENCIL_HOME/src` to the last validated status, and merges all your changes into this version. This allows you to work with this, but in order to check in your changes you have to update everything to the most recent status first, i.e.

\texttt{unix> pc_svnup -src}

Your own changes will be merged into this latest version as before.

NOTE: The functionality of the head of the trunk should be preserved at all times. However, accidents do happen. For the benefit of all other developers, any errors should be corrected within 1-2 hours. This is the reason why the code comes with a file `pencil-code/license/developers.txt`, which should contain contact details of all developers. The \texttt{pc_svnup -val} option allows all other people to stay away from any trouble.
2.4 Getting older versions

You may find that the latest \textit{svn} version of the code produces errors. If you have reasons to believe that this is due to changes introduced on 27 November 2008 (to give an example), you can check out the version prior to this by specifying a revision number with \texttt{svn update -r ####}. One reason why one cannot always reproduce exactly the same situation too far back in time is connected with the fact that processor architecture and the compiler were different, resulting e.g. in different rounding errors.
3 Getting started

To get yourself started, you should run one or several examples which are provided in one of the ‘samples/’ subdirectories. Note that you will only be able to fully assess the numerical solutions if you visualize them with IDL, DX or other tools (see Sect. 5.19).

3.1 Setup

3.1.1 Environment settings

The functionality of helper scripts and IDL routines relies on a few environment variables being set correctly. The simplest way to achieve this is to go to the top directory of the code and source one of the two scripts ‘sourceme.csh’ or ‘sourceme.sh’ (depending on the type of shell you are using):

```
csh> cd pencil-code
csh> source ./sourceme.csh
```
for tcsh or csh users; or

```
sh> cd pencil-code
sh> ./sourceme.sh
```
for users of bash, Bourne shell, or similar shells. You should get output similar to

```
PENCIL_HOME = </home/dobler/f90/pencil-code>
Adding /home/dobler/f90/pencil-code/bin to PATH
```

Apart from the PATH variable, the environment variable IDL_PATH is set to something like ./idl:../idl:+$PENCIL_HOME/idl:/data:<IDL_DEFAULT>.

**Note 1** The <IDL_DEFAULT> mechanism does not work for IDL versions 5.2 or older. In this case, you will have to edit the path manually, or adapt the ‘sourceme’ scripts.

**Note 2** If you don’t want to rely on the ‘sourceme’ scripts’ (quite heuristic) ability to correctly identify the code’s main directory, you can set the environment variable PENCIL_HOME explicitly before you run the source command.

**Note 3** Do not just source the ‘sourceme’ script from your shell startup file (‘~/cshrc’ or ‘~/bashrc’), because it outputs a few lines of diagnostics for each sub-shell, which will break many applications. To suppress all output, follow the instructions given in the header documentation of ‘sourceme.csh’ and ‘sourceme.sh’. Likewise, output from other files invoked by source should also be suppressed.

**Note 4** The second time you source ‘sourceme’, it will not add anything to your PATH variable. This is on purpose to avoid cluttering of your environment: you can source the file as often as you like (in your shell startup script, then manually and in addition in some script you have written), without thinking twice. If, however, at the first sourcing, the setting of PENCIL_HOME was wrong, this mechanism would keep you from ever adding
the right directory to your PATH. In this case, you need to first undefine the environment variable PENCIL_HOME:

```
csh> unsetenv PENCIL_HOME
```
```
csh> source ./sourceme.csh
```
or
```
sh> unset PENCIL_HOME
```
```
sh> . ./sourceme.sh
```

### 3.1.2 Linking scripts and source files

With your environment set up correctly, you can now go to the directory you want to work in and set up subdirectories and links. This is accomplished by the script `pc_setupsrc`, which is located in `$PENCIL_HOME/bin` and is thus now in your executable path.

For concreteness, let us assume you want to use `samples/conv-slab` as your run directory, i.e. you want to run a three-layer slab model of solar convection. You then do the following:

```
unix> cd samples/conv-slab
unix> pc_setupsrc
```

The script has linked a number of scripts from `$PENCIL_HOME/bin`, generated a directory `src` for the source code and linked the Fortran source files (plus a few more files) from `$PENCIL_HOME/src` to that directory:

```
unix> ls -F
reference.out src/
start.csh@ run.csh@ getconf.csh@
start.in run.in print.in
```

### 3.1.3 Adapting ‘Makefile.src’

This step requires some input from you, but you only have to do this once for each machine you want to run the code on. See Sect. 5.2 for a description of the steps you need to take here.

**Note:** If you are lucky and use compilers similar to the ones we have, you may be able to skip this step; but blame yourself if things don’t compile, then. If not, you can run `make` with explicit flags, see Sect. 5.2 and in particular Table 1.

### 3.1.4 Running make

Next, you run `make` in the ‘src’ subdirectory of your run directory. Since you are using one of the predefined test problems, the settings in ‘src/Makefile.local’ and ‘src/cparam.local’ are all reasonable, and you just do
If you have set up the compiler flags correctly, compilation should complete successfully.

### 3.1.5 Choosing a data directory

The code will by default write data like snapshot files to the subdirectory ‘data’ of the run directory. Since this will involve a large volume of IO-operations (at least for large grid sizes), one will normally try to avoid writing the data via NFS. The recommended way to set up a ‘data’ data directory is to generate a corresponding directory on the local disc of the computer you are running on and (soft-)link it to ‘./data’. Even if the link is part of an NFS directory, all the IO operations will be local. For example, if you have a local disc ‘/scratch’, you can do the following:

```bash
unix> mkdir -p /scratch/$USER/pencil-data/samples/conv-slab
unix> ln -s /scratch/$USER/pencil-data/samples/conv-slab .;/data
```

This is done automatically by the `pc_mkdatadir` command which, in turn, is invoked when making a new run directory with the `pc_newrun` command, for example.

Even if you don’t have an NFS-mounted directory (say, on your notebook computer), it is probably still a good idea to have code and data well separated by a scheme like the one described above.

An alternative to symbolic links, is to provide a file called ‘datadir.in’ in the root of the run directory. This file should contain one line of text specifying the absolute or relative data directory path to use. This facility is useful if one wishes to switch one run directory between different data directories. It is suggested that in such cases symbolic links are again made in the run directory to the various locations, then the ‘datadir.in’ need contain only a short relative path.

### 3.1.6 Running the code

You are now ready to start the code:

```bash
unix> start.csh
```

```bash
terminal:
Unix> start.csh
Linux cincinnatus 2.4.18-4GB #1 Wed Mar 27 13:57:05 UTC 2002 i686 unknown
Non-MPI version
datadir = data
Fri Aug 8 21:36:43 CEST 2003
src/start.x
CVS: io_dist.f90 v. 1.61 (brandenb ) 2003/08/03 09:26:55
[...]
CVS: start.in v. 1.4 (dobler ) 2002/10/02 20:11:14

```

```
xngrid,nygrid,nzgrid= 32 32 32
thermodynamics: assume cp=1
```

```
uu: up-down
piecewise polytropic vertical stratification (lnrho)
init_lnrho: cs2bot,cs2top= 1.450000 0.3333330
  e.g. for ionization runs: cs2bot,cs2top not yet set
piecewise polytropic vertical stratification (ss)
```
start.x has completed successfully

0.070u 0.020s 0:00.14 64.2% 0+0k 0+0io 180pf+0w

Fri Aug 8 21:36:43 CEST 2003

This runs ‘src/start.x’ to construct an initial condition based on the parameters set in ‘start.in’. This initial condition is stored in ‘data/proc0/var.dat’ (and in ‘data/proc1/var.dat’, etc. if you run the multiprocessor version). It is fair to say that this is now a rather primitive routine; see ‘pencil-code/idl/read’ for various reading routines. You can then visualize the data using standard idl language.

If you visualize the profiles using IDL (see below), the result should bear some resemblance to Fig. [2] but with different values in the ghost zones (the correct values are set at run-time only) and a simpler velocity profile.

Now we run the code:

unix> run.csh

This executes ‘src/run.x’ and carries out nt time steps, where nt and other run-time parameters are specified in ‘run.in’. On a decent PC (1.7 GHz), 50 time steps take about 10 seconds.

The relevant part of the code’s output looks like

| --it----t-------dt-------urms----umax----rhom------ssm-----dtc----dtu---dtnu---dtchi- |
| 0 0.34  6.792E-03  0.0060  0.0452  14.4708  -0.4478  0.978  0.013  0.207  0.346 |
| 10 0.41  6.787E-03  0.0062  0.0440  14.4707  -0.4480  0.978  0.013  0.207  0.345 |
| 20 0.48  6.781E-03  0.0064  0.0429  14.4705  -0.4481  0.977  0.012  0.207  0.345 |
| 30 0.54  6.777E-03  0.0067  0.0408  14.4703  -0.4482  0.977  0.012  0.207  0.345 |
| 40 0.61  6.776E-03  0.0069  0.0381  14.4702  -0.4482  0.977  0.011  0.207  0.346 |

and lists

1. the number it of the current time step;
2. the time, t;
3. the time step, dt;
4. the rms velocity, urms = \sqrt{\langle u^2 \rangle};
5. the maximum velocity, umax = \max |u|;
6. the mean density, rhom = \langle \rho \rangle;
7. the mean entropy, ssm = \langle s \rangle / c_p;
8. the time step in units of the acoustic Courant step, dtc = \delta t / (c_{s0} \delta x_{min});
9. the time step in units of the advective time step, dtu = \delta t / (c_{\delta t} \delta x / \max |u|);
10. the time step in units of viscous time step, dtnu = \delta t / (c_{\delta t,v} \delta x^2 / \nu_{max});
11. the time step in units of the conductive time step, dtchi = \delta t / (c_{\delta t,v} \delta x^2 / \chi_{max}).

The entries in this list can be added, removed or reformatted in the file ‘print.in’, see Sects 5.5 and J.3. The output is also saved in ‘data/time_series.dat’ and should be identical to the content of ‘reference.out’.
3.2 Further tests

Figure 2: Stratification of the three-layer convection model in 'samples/conv-slab' after 50 timesteps ($t = 0.428$). Shown are (from left to right) density $\rho$, vertical velocity $u_z$, entropy $s/c_p$ and temperature $T$ as functions of the vertical coordinate $z$ for about ten different vertical lines in the computational box. The dashed lines denote domain boundaries: $z < -0.68$ is the lower ghost zone (points have no physical significance); $-0.68 < z < 0$ is a stably stratified layer ($ds/dz > 0$); $0 < z < 1$ is the unstable layer ($ds/dz < 0$); $1 < z < 1.32$ is the isothermal top layer; $z > 1.32$ is the upper ghost zone (points have no physical significance).

If you have IDL, you can visualize the stratification with (see Sect. 5.19.4 for details)

```
unix > idl
IDL > pc_read_var,obj=var,/trimall
IDL > tvscl,var,uu(*,*,0,0)
```

which shows $u_x$ in the $xy$ plane through the first meshpoint in the $z$ direction. There have been some now outdated specific routines that produce results like that shown in Fig. 2.

Note: If you want to run the code with MPI, you will probably need to adapt ‘getconf.csh’, which defines the commands and flags used to run MPI jobs (and which is sourced by the scripts ‘start.csh’ and ‘run.csh’). Try

```
csh -v getconf.csh
or
csh -x getconf.csh
```

to see how ‘getconf.csh’ makes its decisions. You would add a section for the host name of your machine with the particular settings. Since ‘getconf.csh’ is linked from the central directory ‘pencil-code/bin’, your changes will be useful for all your other runs too.

3.2 Further tests

There is a number of other tests in the ‘samples/’ directory. You can use the script ‘bin/auto-test’ to automatically run these tests and have the output compared to refer-
ence results.
4 Code structure

4.1 Directory tree

The overall directory structure of the code is shown in Fig. 3. Under ‘pencil-code’, there are currently the following files and directories:

```
bin/  config/  doc/  idl/  license/  perl/  samples/  sourceme.sh  utils/
bugs/  dx/  lib/  misc/  README  sourceme.csh  src/  www/
```

Almost all of the source code is contained in the directory ‘src/’, but in order to encapsulate individual applications, the code is compiled separately for each run in a local directory ‘src’ below the individual run directory, like e.g. ‘samples/conv-slab/src’.

It may be a good idea to keep your own runs also under svn or cvs (which is older than but similar to svn), but this would normally be a different repository. On the machine where you are running the code, you may want to check them out into a subdirectory of
For example, we have our own runs in a repository called ‘pencil-runs’, so we do

```sh
unix> cd $PENCIL_HOME
unix> svn co runs pencil-runs
```

In this case, ‘runs’ contains individual run directories, grouped in classes (like ‘spher’ for spherical calculations, or ‘kinematic’ for kinematic dynamo simulations). The current list of classes in our own ‘pencil-runs’ repository is

```
1d-tests/  disc/  kinematic/  rings/
2d-tests/  discont/  Misc/  slab_conv/
3d-tests/  discussion/  OLD/  test/
buoyTube/  forced/  pass_only/
convstar/  interstellar/  radiation/
```

The directory ‘forced/’ contains some forced turbulence runs (both magnetic and non-magnetic); ‘gravz/’ contains runs with vertical gravity; ‘rings/’ contains decaying MHD problems (interlocked flux rings as initial condition, for example); and ‘kinematic/’ contains kinematic dynamo problems where the hydrodynamics is turned off entirely. The file ‘samples/README’ should contain an up-to-date list and short description of the individual classes.

The subdirectory ‘src’ of each run directory contains a few local configuration files (currently these are ‘Makefile.local’ and ‘cparam.local’) and possibly ‘ctimeavg.local’. To compile the samples, links the files ‘.f90’, ‘.c’ and ‘Makefile.src’ need to be linked from the top file[src]/src directory to the local directory ‘./src’. These links are set up by the script pc_setupsrc when used in the root of a run directory.

General-purpose visualization routines for IDL or DX are in the directories ‘idl’ and ‘dx’, respectively. There are additional and more specialized IDL directories in the different branches under ‘pencil-runs’.

The directory ‘doc’ contains this manual; ‘bin’ contains a number of utility scripts (mostly written in csh and Perl), and in particular the ‘start.csh’, ‘run.csh’, and ‘getconf.csh’ scripts. The ‘.svn’ directory is used (you guessed it) by .svn, and is not normally directly accessed by the user; ‘bugs’, finally is used by us for internal purposes.

The files ‘sourceme.csh’ and ‘sourceme.sh’ will set up some environment variables — in particular PATH — and aliases/shell functions for your convenience. If you do not want to source one of these files, you need to make sure your IDL path is set appropriately (provided you want to use IDL) and you will need to address the scripts from ‘bin’ with their explicit path name, or adjust your PATH manually.

## 4.2 Basic concepts

### 4.2.1 Data access in pencils

Unlike the CRAY computers that dominated supercomputing in the 80s and early 90s, all modern computers have a cache that constitutes a significant bottleneck for many

---

1. Our ‘pencil-runs’ directory also contains runs that were done some time ago. Occasionally, we try to update these, especially if we have changed names or other input conventions.
codes. This is the case if large three-dimensional arrays are constantly used within each
time step, which has the obvious advantage of working on long arrays and allows vector-
ization of elementary machine operations. This approach also implies conceptual sim-
plicity of the code and allows extensive use of the intuitive F90 array syntax. However,
a more cache-efficient way of coding is to calculate an entire time step (or substep of a
multi-stage time-stepping scheme) only along a one-dimensional pencil of data within
the numerical grid. This technique is more efficient for modern RISC processors: on
Linux PCs and SGI workstations, for example, we have found a speed-up by about 60%
in some cases. An additional advantage is a drastic reduction in temporary storage for
auxiliary variables within each time step.

4.2.2 Modularity

Each run directory has a file ‘src/Makfile.local’ in which you choose certain modules\(^2\),
which tell the code whether or not entropy, magnetic fields, hydrodynamics, forcing, etc.
should be invoked. For example, the settings for forced turbulent MHD simulations are

```
HYDRO = hydro
DENSITY = density
ENTROPY = noentropy
MAGNETIC = magnetic
GRAVITY = nogravity
FORCING = forcing

MPICOMM = nompicomm
GLOBAL = noglobal
IO = io_dist
FOURIER = nofourier
```

This file will be processed by make and the settings are thus assignments of make
variables. Apart from the physics modules (equation of motion: yes, density [pressure]:
yes, entropy equation: no, magnetic fields: yes, gravity: no, forcing: yes), a few technical
modules can also be used or deactivated; in the example above, these are MPI (switched
off), additional global variables (none), input/output (distributed), and FFT (not used).
The table in Sect. I in the Appendix lists all currently available modules.

Note that most of these make variables must be set, but they will normally obtain rea-
sonable default values in ‘Makfile’ (so you only need to set the non-standard ones in
‘Makfile.local’). It is by using this switching mechanism through make that we achieve
high flexibility without resorting to excessive amounts of cryptic preprocessor directives
or other switches within the code.

Many possible combinations of modules have already been tested and examples are part
of the distribution, but you may be interested in a combination which was never tried
before and which may not work yet, since the modules are not fully orthogonal. In such
cases, we depend on user feedback for fixing problems and documenting the changes for
others.

\(^2\)We stress once more that we are not talking about F90 modules here, although there is some connec-
tion, as most of our modules define F90 modules: For example each of the modules gravity_simple, grav_r
and nogravity defines a Fortran module Gravity.
4.3 Files in the run directories

4.3.1 ‘start.in’, ‘run.in’, ‘print.in’

These files specify the startup and runtime parameters (see Sects. 5.12 and 5.2), and the list of diagnostic variables to print (see 5.5). They specify the setup of a given simulation and are kept under svn in the individual ‘samples’ directories.

You may want to check for the correctness of these configuration files by issuing the command `pc_configtest`.

4.3.2 ‘datadir.in’

If this file exists, it must contain the name of an existing directory, which will be used as data directory, i.e. the directory where all results are written. If ‘datadir.in’ does not exist, the data directory is ‘data/’.

4.3.3 ‘reference.out’

If present, ‘reference.out’ contains the output you should obtain in the given run directory, provided you have not changed any parameters. To see whether the results of your run are OK, compare ‘time_series.dat’ to ‘reference.out’:

```
unix> diff data/time_series.dat reference.out
```

4.3.4 ‘start.csh’, ‘run.csh’, ‘getconf.csh’ [obsolete; see Sect. 5.1]

These are links to ‘$PENCIL_HOME/bin’. You will be constantly using the scripts ‘start.csh’ and ‘run.csh’ to initialize the code. Things that are needed by both (like the name of the `mpirun` executable, `MPI` options, or the number of processors) are located in ‘getconf.csh’, which is never directly invoked.

4.3.5 ‘src/’

The ‘src’ directory contains two local files, ‘src/Makefile.local’ and ‘src/cparam.local’, which allow the user to choose individual modules (see 4.2.2) and to set parameters like the grid size and the number of processors for each direction. These two files are part of the setup of a given simulation and are kept under svn in the individual ‘samples’ directories.

The file ‘src/cparam.inc’ is automatically generated by the script ‘mkcparam’ and contains the number of fundamental variables for a given setup.

All other files in ‘src/’ are either links to source files (and ‘Makefile.src’) in the ‘$PENCIL_HOME/src’ directory, or object and module files generated by the compiler.
4.3 Files in the run directories

4.3.6 ‘data/’

This directory (the name of which will actually be overwritten by the first line of ‘datadir.in’, if that file is present; see §4.3.2) contains the output from the code:

‘data/dim.dat’ The global array dimensions.

‘data/legend.dat’ The header line specifying the names of the diagnostic variables in ‘time_series.dat’.

‘data/time_series.dat’ Time series of diagnostic variables (also printed to stdout). You can use this file directly for plotting with Gnuplot, IDL, Xmgrace or similar tools (see also §5.19).

‘data/tsnap.dat’, ‘data/tvid.dat’ Time when the next snapshot ‘VAR.N’ or animation slice should be taken.

‘data/params.log’ Keeps a log of all your parameters: ‘start.x’ writes the startup parameters to this file, ‘run.x’ appends the runtime parameters and appends them anew, each time you use the ‘RELLOAD’ mechanism (see §5.10).

‘data/param.nml’ Complete set of startup parameters, printed as Fortran namelist. This file is read in by ‘run.x’ (this is how values of startup parameters are propagated to ‘run.x’) and by IDL (if you use it).

‘data/param2.nml’ Complete set of runtime parameters, printed as Fortran namelist. This file is read by IDL (if you use it).

‘data/index.pro’ Can be used as include file in IDL and contains the column in which certain variables appear in the diagnostics file (‘time_series.dat’). It also contains the positions of variables in the ‘VAR.N’ files. These positions depend on whether entropy or noentropy, etc, are invoked. This is a temporary solution and the file may disappear in future releases.

‘data/interstellar.dat’ Unformatted file containing the time at which the next supernova event will occur, under certain supernova schemes. (Only needed by the interstellar module.)

‘data/proc0’, ‘data/proc1’, … These are the directories containing data from the individual processors. So after running an MPI job on two processors, you will have the two directories ‘data/proc0’ and ‘data/proc1’. Each of the directories can contain the following files:

‘var.dat’ binary file containing the latest snapshot;
‘VAR.N’ binary file containing individual snapshot number N;
‘dim.dat’ ASCII file containing the array dimensions as seen by the given processor;
‘time.dat’ ASCII file containing the time corresponding to ‘var.dat’ (not actually used by the code, unless you use the io_mpiiodist.f90 module);
‘grid.dat’ binary file containing the part of the grid seen by the given processor;
‘seed.dat’ the random seed for the next time step (saved for reasons of reproducibility). For multi-processor runs with velocity forcing, the files ‘procN/seed.dat’ must all contain the same numbers, because globally coherent waves of given wavenumber are used;
‘X.xy’, ‘X.xz’, ‘X.yz’ two-dimensional sections of variable X, where X stands for the corresponding variable. The current list includes

<table>
<thead>
<tr>
<th>bx.xy</th>
<th>bx.xz</th>
<th>by.xy</th>
<th>by.xz</th>
<th>bz.xy</th>
<th>bz.xz</th>
<th>divu.xy</th>
<th>lnrho.xz</th>
</tr>
</thead>
<tbody>
<tr>
<td>ss.xz</td>
<td>ux.xy</td>
<td>ux.xz</td>
<td>uz.xy</td>
<td>uz.xz</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each processor writes its own slice, so these need to be reassembled if one wants to plot a full slice.
5 Using the code

5.1 Configuring the code to compile and run on your computer

Note: We recommend to use the procedure described here, rather than the old method described in Sects. 5.2 and 4.3.4.

Quick instructions: You may compile with a default compiler-specific configuration:

1. Single-processor using the GNU compiler collection:
   unix> pc_build -f GNU-GCC

2. Multi-processor using GNU with MPI support:
   unix> pc_build -f GNU-GCC_MPI

Many compilers are supported already, please refer to the available config files in ‘$PENCIL_HOME/config/compilers/*.conf’, e.g. ‘Intel.conf’ and ‘Intel_MPI.conf’.

If you have to set up some compiler options specific to a certain host system you work on, or if you like to create a host-specific configuration file so that you can simply execute pc_build without any options, you can clone an existing host-file, just include an existing compiler configuration, and simply only add the options you need. A good example of a host-file is ‘$PENCIL_HOME/config/hosts/IWF/host-andromeda-GNU_Linux-Linux.conf’. You may save a clone under ‘$PENCIL_HOME/config/hosts/<ID>.conf’, where ‘<ID>’ is to be replaced by the output of pc_build -i. This will be the new default for pc_build.

If you don’t know what this was all about, read on.

In essence, configuration, compiling and running the code work like this:

1. Create a configuration file for your computer’s host ID.
2. Compile the code using pc_build.
3. Run the code using pc_run

In the following, we will discuss the essentials of this scheme. Exhaustive documentation is available with the commands perldoc Pencil::ConfigFinder and perldoc PENCIL::ConfigParser.

5.1.1 Locating the configuration file

Commands like pc_build and pc_run use the Perl module ‘Pencil::ConfigFinder’ to locate an appropriate configuration file and ‘Pencil::ConfigParser’ to read and interpret it. When you use ‘ConfigFinder’ on a given computer, it constructs a host ID for the system it is running on, and looks for a file ‘host_ID.conf’ in any subdirectory of ‘$PENCIL_HOME/config/hosts’.

E.g., if the host ID is “workhorse.pencil.org”, ‘ConfigFinder’ would consider the file ‘$PENCIL_HOME/config/hosts/pencil.org/workhorse.pencil.org.conf’.
Note 1: The location in the tree under ‘hosts/’ is irrelevant, which allows you to group related hosts by institution, owner, hardware, etc.

Note 2: ‘ConfigFinder’ actually uses the following search path:

1. ‘./config’
2. ‘$PENCIL_HOME/config-local’
3. ‘$HOME/.pencil/config-local’
4. ‘$PENCIL_HOME/config’

This allows you to override part of the ‘config/’ tree globally on the given file system, or locally for a particular run directory, or for one given copy of the PENCIL CODE. This search path is used both, for locating the configuration file for your host ID, and for locating included files (see below).

The host ID is constructed based on information that is easily available for your system. The algorithm is as follows:

1. Most commands using ‘ConfigFinder’ have a ‘--host-id’ (sometimes abbreviated as ‘-H’) option to explicitly set the host ID.
2. If the environment variable PENCIL_HOST_ID is set, its value is used.
3. If any of the files ‘./host-ID’, ‘$PENCIL_HOME/host-ID’, ‘$HOME/.pencil/host-ID’, exists, its first line is used.
4. If ‘ConfigFinder’ can get hold of a fully qualified host name, that is used as host ID.
5. Else, a combination of host name, operating system name and possibly some other information characterizing the system is used.
6. If no config file for the host ID is found, the current operating system name is tried as fallback host ID.

To see which host IDs are tried (up to the first one for which a configuration file is found), run

`unix> pc_build --debug-config`

This command will tell you the host-ID of the system that you are using:

`unix> pc_build -i`

5.1.2 Structure of configuration files

It is strongly recommended to include in a users configuration file one of the preset compiler suite configuration files. Then, only minor options need to be set by a user, e.g. the optimization flags. One of those user configuration files looks rather simple:

```
# Simple config file. Most files don't need more.
%include compilers/GNU-GCC
```

or if you prefer a different compiler:
# Simple Intel compiler suite config file.
#include compilers/Intel

A more complex file (using MPI with additional options) would look like this:

# More complex config file.
#include compilers/GNU-GCC_MPI

%section Makefile
    MAKE_VAR1 = -j4  # joined compilation with four threads
    FFLAGS += -O3 -Wall -fbacktrace  # don’t redefine, but append with ‘+=’
%endsection Makefile

%section runtime
    mpiexec = mpirun  # some MPI backends need a redefinition of mpiexec
%endsection runtime

%section environment
    SCRATCH_DIR=/var/tmp/$USER
%endsection environment

Adding "_MPI" to a compiler suite’s name is usually sufficient to use MPI.

**Note 3:** We strongly advise *not* to mix Fortran- and C-compilers from different manufacturers or versions by manually including multiple separate compiler configurations.

**Note 4:** We strongly advise to use *at maximum* the optimization levels ‘-O2’ for the Intel compiler and ‘-O3’ for all other compilers. Higher optimization levels implicate an inadequate loss of precision.

The ‘.conf’ files consist of the following elements:

**Comments:** A # sign and any text following it on the same line are ignored.

**Sections:** There are three sections:

- **Makefile** for setting make parameters
- **runtime** for adding compiler flags used by pc_run
- **environment** shell environment variables for compilation and running

**Include statements:** An %include ... statement is recursively replaced by the contents of the files it points to.

The included path gets a .conf suffix appended. Included paths are typically “absolute”, e.g.

```
#include os/Unix
```

will include the file ‘os/Unix.conf’ in the search path listed above (typically from ‘$PENCIL_HOME/config’). However, if the included path starts with a dot, it is a relative path, so

---

3However, if the include statement is inside a section, only the file’s contents inside that section are inserted.
%include ./*Unix

will only search in the directory where the including file is located.

Assignments: Statements like FFLAGS += -O3 or mpiexec=mpirun are assignments and will set parameters that are used by pc_build/make or pc_run.

Lines ending with a backslash `\' are continuation lines.

If possible, one should always use incremental assignments, indicated by using a `+=` sign instead of `=`, instead of redefining certain flags.

Thus,

CFLAGS += -O3
CFLAGS += -I../include -Wall

is the same as

CFLAGS = $(CFLAGS) -O3 -I../include -Wall

5.1.3 Compiling the code

Use the pc_build command to compile the code:

unix> pc_build # use default compiler suite
unix> pc_build -f Intel_MPI # specify a compiler suite
unix> pc_build -f os/GNU_Linux,mpi/open-mpi # explicitly specify config files
unix> pc_build VAR=something # set variables for the makefile
unix> pc_build --cleanall # remove generated files

The third example circumvents the whole host ID mechanism by explicitly instructing pc_build which configuration files to use. The fourth example shows how to define extra variables (VAR=something) for the execution of the Makefile.

See pc_build --help for a complete list of options.

5.1.4 Running the code

Use the pc_run command to run the code:

unix> pc_run # start if necessary, then run
unix> pc_run start
unix> pc_run run

unix> pc_run start run^3 # start, then run 3 times
unix> pc_run start run run run # start, then run 3 times
unix> pc_run ^3 # start if necessary, then run 3 times

See pc_run --help for a complete list of options.

5.1.5 Testing the code

The pc_auto-test command uses pc_build for compiling and pc_run for running the tests. Here are a few useful options:
Adapting ‘Makefile.src’ [obsolete; see Sect. 5.1]

By default, one should use the above described configuration mechanism for compilation. If for whatever reason one needs to work with a modified ‘Makefile’, there is a mechanism for picking the right set of compiler flags based on the hostname.

To give you an idea of how to add your own machines, let us assume you have several Linux boxes running a compiler f95 that needs the options ‘-O2 -u’, while one of them, Janus, runs a compiler f90 which needs the flags ‘-O3’ and requires the additional options ‘-lmpi -llam’ for MPI.

The ‘Makefile.src’ you need will have the following section:

```bash
## Linux:
[...] (leave everything from original Makefile and add:)
#FC=f95
#FFLAGS=-O2 -u
#FC=f90    #(Janus)
#FFLAGS=-O3    #(Janus)
#LDMPI=-lmpi -llam    #(Janus)
```

See pencil-test --help for a complete list of options.

5.2 Adapting ‘Makefile.src’ [obsolete; see Sect. 5.1]

unix> pc_auto-test
unix> pc_auto-test --no-pencil-check  # suppress pencil consistency check
unix> pc_auto-test --max-level=1      # run only tests in level 0 and 1
unix> pc_auto-test --time-limit=2m    # kill each test after 2 minutes

See pc_auto-test --help for a complete list of options.
Table 1: Compiler flags for common compilers. Note that some combinations of OS and compiler require much more elaborate settings; also, if you use MPI, you will have to set LD_MPI.

<table>
<thead>
<tr>
<th>Compiler</th>
<th>FC</th>
<th>FFLAGS</th>
<th>CC</th>
<th>CFLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix/POSIX:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNU</td>
<td>gfortran</td>
<td>-O3</td>
<td>gcc</td>
<td>-O3 -DFUNDERSC=1</td>
</tr>
<tr>
<td>Intel</td>
<td>ifort</td>
<td>-O2</td>
<td>icc</td>
<td>-O3 -DFUNDERSC=1</td>
</tr>
<tr>
<td>PGI</td>
<td>pgf95</td>
<td>-O3</td>
<td>pgcc</td>
<td>-O3 -DFUNDERSC=1</td>
</tr>
<tr>
<td>G95</td>
<td>g95</td>
<td>-O3 -fno-second-underscore</td>
<td>gcc</td>
<td>-O3 -DFUNDERSC=1</td>
</tr>
<tr>
<td>Absoft</td>
<td>f95</td>
<td>-O3 -N15</td>
<td>gcc</td>
<td>-O3 -DFUNDERSC=1</td>
</tr>
<tr>
<td>IBM XL</td>
<td>xlf95</td>
<td>-q suffix=f=f90 -O3</td>
<td>xlc</td>
<td>-O3 -DFUNDERSC=1</td>
</tr>
</tbody>
</table>

outdated:

|                   |      |                 |       |                |
| IRIX Mips         | f90  | -64 -O3 -mips4  | cc    | -O3 -64 -DFUNDERSC=1 |
| Compaq            | f90  | -fast -O3       | cc    | -O3 -DFUNDERSC=1 |

### End machine-dependent

**Note 1** There is a script for adapting the Makefile: ‘adapt-mkfile’. In the example above, #(Janus) is not a comment, but marks this line to be activated (uncommented) by adapt-mkfile if your hostname (‘uname -n’) matches ‘Janus’ or ‘janus’ (capitalization is irrelevant). You can combine machine names with a vertical bar: a line containing #(onsager|Janus) will be activated on both, Janus and Onsager.

**Note 2** If you want to experiment with compiler flags, or if you want to get things running without setting up the machine-dependent section of the ‘Makefile’, you can set make variables at the command line in the usual manner:

```
src> make FC=f90 FFLAGS='-fast -u'
```

will use the compiler f90 and the flags ‘-fast -u’ for both compilation and linking. Table 1 summarizes flags we use for common compilers.

### 5.3 Changing the resolution

It is advisable to produce a new run directory each time you run a new case. (This does not include restarts from an old run, of course.) If you have a 32^3 run in some directory ‘runA_32a’, then go to its parent directory, i.e.

```
runA_32a> cd ..
forced> pc_newrun runA_32a runA_64a
forced> cd runA_64a/src
forced> vi cparam.local
```

and edit the ‘cparam.local’ for the new resolution.

If you have ever wondered why we don’t do dynamic allocation of the main variable (f) array, the main reason is that with static allocation the compiler can check whether we are out of bounds.
5.4 Using a non-equidistant grid

We introduce a non-equidistant grid \( z_i \) (by now, this is also implemented for the other directions) as a function \( z(\zeta) \) of an equidistant grid \( \zeta \), with grid spacing \( \Delta \zeta = 1 \).

The way the parameters are handled, the box size and position are *not* changed when you switch to a non-equidistant grid, i.e. they are still determined by \( xyz0 \) and \( Lxyz \).

The first and second derivatives can be calculated by

\[
\frac{df}{dz} = \frac{df}{d\zeta} \frac{d\zeta}{dz} = \frac{1}{z'} f'(\zeta), \quad \frac{d^2 f}{dz^2} = \frac{1}{z'^2} f''(\zeta) - \frac{z''}{z'^3} f'(\zeta),
\]

which can be written somewhat more compactly using the inverse function \( \zeta(z) \):

\[
\frac{df}{dz} = \zeta'(z) f'(\zeta), \quad \frac{d^2 f}{dz^2} = \zeta''(z) f''(\zeta) + \zeta''(z) \zeta'(z) f'(\zeta).
\]

Internally, the code uses the quantities \( dz1 \equiv 1/z' = \zeta'(z) \) and \( dz_tilde \equiv -z''/z'^2 = \zeta''/\zeta' \), and stores them in ‘data/procN/grid.dat’.

The parameters \( \text{lequidist} \) (a 3-element logical array), \( \text{grid_func} \), \( \text{coeff_grid} \) (a \( \geq 2 \)-element real array) are used to choose a non-equidistant grid and define the function \( z(\zeta) \). So far, one can choose between \( \text{grid_func}='\text{sinh}' \), \( \text{grid_func}='\text{linear}' \) (which produces an equidistant grid for testing purposes), and \( \text{grid_func}='\text{step-linear}' \).

**The sinh profile:** For \( \text{grid_func}='\text{sinh}' \), the function \( z(\zeta) \) is given by

\[
z(\zeta) = z_0 + L_z \frac{\sinh a(\zeta-\zeta_1) + \sinh a(\zeta-\zeta_2)}{\sinh a(\zeta_2-\zeta_1) + \sinh a(\zeta-\zeta_1)},
\]

where \( z_0 \) and \( z_0+L_z \) are the lowest and uppermost levels, \( \zeta_1, \zeta_2 \) are the \( \zeta \) values representing those levels (normally \( \zeta_1 = 0, \zeta_2 = N_z - 1 \) for a grid of \( N_z \) vertical layers [excluding ghost layers]), and \( \zeta_s \) is the \( \zeta \) value of the inflection point of the sinh function. The \( z \) coordinate and \( \zeta \) value of the inflection point are related via

\[
z_* = z_0 + L_z \frac{\sinh a(\zeta_2-\zeta_1)}{\sinh a(\zeta_2-\zeta_1) + \sinh a(\zeta_2-\zeta_1)},
\]

which can be inverted (“after some algebra”) to

\[
\zeta_* = \frac{\zeta_1+\zeta_2}{2} + \frac{1}{a} \text{artanh} \left( \frac{2(z_*-z_0)}{L_z} - 1 \right) \tanh \frac{\zeta_2-\zeta_1}{2}.
\]

**General profile:** For a general monotonic function \( \psi() \) instead of \( \sinh \) we get,

\[
z(\zeta) = z_0 + L_z \frac{\psi[a(\zeta-\zeta_1)] + \psi[a(\zeta-\zeta_2)]}{\psi[a(\zeta_2-\zeta_1)] + \psi[a(\zeta_2-\zeta_1)]},
\]

and the reference point \( \zeta_* \) is found by inverting

\[
z_* = z_0 + L_z \frac{\psi[a(\zeta_2-\zeta_1)]}{\psi[a(\zeta_2-\zeta_1)] + \psi[a(\zeta_2-\zeta_1)]},
\]

numerically.
**Duct flow:** The profile function `grid_function='duct'` generates a grid profile for turbulent flow in ducts. For a duct flow, most gradients are steepest close to the walls, and hence very fine resolution is required near the walls. Here we follow the method of [21] and use a Chebyshev-type grid with a cosine distribution of the grid points such that in the $y$ direction they are located at

$$y_j = h \cos \theta_j,$$

where

$$\theta_j = \frac{(N_y - j) \pi}{N_y - 1}, \quad j = 1, 2, \ldots, N_y$$

and $h = L_y/2$ is the duct half-width.

Currently this method is only adapted for ducts where $x$ is the stream-wise direction, $z$ is in the span-wise direction and the walls are at $y = y_0$ and $y = y_0 + L_y$.

In order to have fine enough resolution, the first grid point should be a distance $\delta = 0.05 l_w$ from the wall, where

$$l_w = \frac{\nu}{u_r}, \quad u_r = \sqrt{\frac{\tau_w}{\rho}},$$

and $\tau_w$ is the shear wall stress. This is accomplished by using at least

$$N_y \geq N_y^* = \frac{\pi}{\arccos \left(1 - \frac{\delta}{h}\right)} + 1$$

$$= \pi \sqrt{\frac{h}{2\delta}} + 1 - \frac{\pi}{24} \frac{2\delta}{h} + O \left(\frac{\delta}{h}^{3/2}\right)$$

grid points in the $y$-direction. After rounding up to the next integer value, we find that the truncated condition

$$N_y \geq \left\lceil \pi \sqrt{\frac{h}{2\delta}} \right\rceil + 1$$

(wher $[x]$ denotes the ceiling function, i.e. the smallest integer equal to, or larger than, $x$) gives practically identical results.

**Example:** To apply the sinh profile, you can set the following in ‘start.in’ (this example is from ‘samples/sound-spherical-noequi’):

```plaintext
&init_pars
/```


The parameter array \texttt{coeff.grid} represents \( z^* \) and \( a \); the bottom height \( z_0 \) and the total box height \( L_z \) are taken from \texttt{xyz0} and \texttt{Lxyz} as in the equidistant case. The inflection point of the sinh profile (the part where it is linear) is not in the middle of the box, because we have set \texttt{xyz-star(3)} (i.e. \( z^* \)) to \(-2\).

### 5.5 Diagnostic output

Every \texttt{it1} time steps (\texttt{it1} is a runtime parameter, see Sect. J.2), the code writes monitoring output to \texttt{stdout} and, parallel to this, to the file `data/time_series.dat`. The variables that appear in this listing and the output format are defined in the file `print.in` and can be changed without touching the code (even while the code is running). A simple example of `print.in` may look like this:

\[
\begin{align*}
\text{t}(F10.3) \\
\text{urms}(E13.4) \\
\text{rhom}(F10.5) \\
\text{oum}
\end{align*}
\]

which means that the output table will contain time \( t \) in the first column formatted as (F10.3), followed by the mean squared velocity, \texttt{urms} (i.e. \( \langle u^2 \rangle^{1/2} \)) in the second column with format (E13.4), the average density \texttt{rhom} (\( \langle \rho \rangle \), which allows to monitor mass conservation) formatted (F10.5) and the kinetic helicity \texttt{oum} (that is \( \langle \omega \cdot u \rangle \)) in the last column with the default format (E10.2).4

The corresponding diagnostic output will look like

\[
\begin{align*}
\text{----t---------urms--------rhom------oum----} \\
0.000 & 4.9643E-03 & 14.42457 & -8.62E-06 \\
0.032 & 3.9423E-03 & 14.42446 & -5.25E-06 \\
0.063 & 6.8399E-03 & 14.42449 & -3.50E-06 \\
0.095 & 1.1437E-02 & 14.42455 & -2.58E-06 \\
0.126 & 1.6980E-02 & 14.42457 & -1.93E-06
\end{align*}
\]

### 5.6 Data files

#### 5.6.1 Snapshot files

Snapshot files contain the values of all evolving variables and are sufficient to restart a run. In the case of an MHD simulation with entropy equation, for example, the snapshot files will contain the values of velocity, logarithmic density, entropy and the magnetic vector potential.

There are two kinds of snapshot files: the current snapshot and permanent snapshots, both of which reside in the directory `data/procN/`. The parameter \texttt{isav} determines the frequency at which the current snapshot `data/procN/var.dat` is written. If you keep this frequency too high, the code will spend a lot of time on I/O, in particular for large jobs; too low a frequency makes it difficult to follow the evolution interactively during test runs. There is also the \texttt{ialive} parameter. Setting this to 1 or 10 gives an updated

---

4 The format specifiers are like in Fortran, apart from the fact that the E format will use standard scientific format, corresponding to the Fortran 1pE syntax. Seasoned Fortran IV programmers may use formats like (0pE13.4) to enjoy nostalgic feelings, or (1pF10.5) if they depend on getting wrong numbers.
timestep in the files ‘data/proc*/alive.info’. You can put ialive=0 to turn this off to limit the I/O on your machine.

The permanent snapshots ‘data/proc*/VARN’ are written every dsnap time units. These files are numbered consecutively from \( N = 1 \) upward and for long runs they can occupy quite some disk space. On the other hand, if after a run you realize that some additional quantity \( q \) would have been important to print out, these files are the only way to reconstruct the time evolution of \( q \) without re-running the code.

**File structure**  
Snapshot files consist of the following *Fortran records*[^1]:

1. variable vector \( f \ [mx \times my \times mz \times nvar] \)
2. time \( t \ [1] \), coordinate vectors \( x \ [mx] \), \( y \ [my] \), \( z \ [mz] \), grid spacings \( \delta x \ [1] \), \( \delta y \ [1] \), \( \delta z \ [1] \), shearing-box shift \( \Delta y \ [1] \)

All numbers (apart from the record markers) are single precision (4-byte) floating point numbers, unless you use double precision (see §5.21, in which case all numbers are 8-byte floating point numbers, while the record markers remain 4-byte integers.

The script `pc_tsnap` allows you to determine the time \( t \) of a snapshot file:

```plaintext
unix> pc_tsnap data/proc0/var.dat  
data/proc0/var.dat: t = 8.32456
unix> pc_tsnap data/proc0/VAR2  
data/proc0/VAR2: t = 2.00603
```

### 5.7 Video files and slices

We use the terms *video files* and *slice files* interchangeably. These files contain a time series of values of one variable in a given plane. The output frequency of these video snapshots is set by the parameter `dvid` (in code time units).

When output to video files is activated by some settings in ‘run.in’ (see example below) and the existence of ‘video.in’, slices are written for four planes:

1. \( x-z \) plane (\( y \) index \( iy \); file suffix .xz)
2. \( y-z \) plane (\( y \) index \( ix \); suffix .yz)
3. \( x-y \) plane (\( y \) index \( iz \); suffix .xy)
4. another slice parallel to the \( x-y \) plane (\( y \) index \( iz2 \); suffix .xy2)

You can specify the position of the slice planes by setting the parameters \( ix \), \( iy \), \( iz \) and \( iz2 \) in the namelist `run_pars` in ‘run.in’. Alternatively, you can set the input parameter `slice_position` to one of ‘p’ (periphery of box) or ‘m’ (middle of box). Or you can also specify the \( z \)–position in terms of \( z \) using the tags `zbot_slice` and `ztop_slice`. In this case, the `zbot_slice` slice will have the suffix .xy and the `ztop_slice` the suffix .xy2

In the file ‘video.in’ of your run directory, you can choose for which variables you want to get video files; valid choices are listed in §4.4.

[^1]: A *Fortran record* is marked by the 4-byte integer byte count of the data in the record at the beginning and the end, i.e. has the form \((N_{\text{bytes}}, \text{raw data}, N_{\text{bytes}})\)
The *slice files* are written in each processor directory `data/proc/*` and have a file name indicating the individual variable (e.g. `slice_uu1.yz` for a slice of $u_x$ in the $y$-$z$ plane). Before visualizing slices one normally wants to combine the sub-slices written by each processor into one global slice (for each plane and variable). This is done by running `src/read_videofiles.x`, which will prompt for the variable name, read the individual sub-slices and write global slices to `data/`. Once all global slices have been assembled you may want to remove the local slices `data/proc*/slice*`.

To read all sub-slices demanded in `video.in` at once use `src/read_all_videofiles.x`. This program doesn’t expect any user input and can thus be submitted in computing queues.

For visualization of slices, you can use the *IDL* routines `rvid_box.pro`, `rvid_plane.pro`, or `rvid_line.pro` which allows the flag `/png` for writing PNG images that can then be combined into an MPEG movie using `mpeg_encode`. Based on `rvid_box`, you can write your own video routines in *IDL*.

**An example** Suppose you have set up a run using `entropy.f90` and `magnetic.f90` (most probably together with `hydro.f90` and other modules). In order to animate slices of entropy $s$ and the $z$-component $B_z$ of the magnetic field, in planes passing through the center of the box, do the following:

1. Write the following lines to `video.in` in your run directory:
   ```
   ss
   bb
   divu
   ```

2. Edit the namelist `run_pars` in the file `run.in`. Request slices by setting `write_slices` and set `dvid` and `slice_position` to reasonable values, say
   ```
   !write_slices=T !(no longer works; write requested slices into video.in)
   dvid=0.05
   slice_position='m'
   ```

3. Run the-pencil code:
   ```
   unix> start.csh
   unix> run.csh
   ```

4. Run `src/read_videofiles.x` to assemble global slice files from those scattered across `data/proc/*`:
   ```
   unix> src/read_videofiles.x
   enter name of variable (lnrho, uu1, ..., bb3): ss
   unix> src/read_videofiles.x
   enter name of variable (lnrho, uu1, ..., bb3): bb3
   ```

5. Start *IDL* and run `rvid_box`:
   ```
   unix> idl
   IDL> rvid_box,’bb3’
   IDL> rvid_box,’ss’,min=-0.3,max=2.
   ```
   etc.
Another example  Suppose you have set up a run using `magnetic.f90` and some other modules. This run studies some process in a “surface” layer inside the box. This “surface” can represent a sharp change in density or turbulence. So you defined your box setting the \( z = 0 \) point at the surface. Therefore, your ‘start.in’ file will look something similar to:

```plaintext
&init_pars
  lperi=T,T,F
  bcz = 's','s','a','hs','s','s','a'
  xyz0 = -3.14159, -3.14159, -3.14159
  Lxyz = 6.28319, 6.28319, 9.42478
```

Now you can analyze quickly the surface of interest and some other \( xy \)–slice setting `zbot_slice` and `ztop_slice` in the ‘run.in’ file:

```plaintext
&run_pars
  slice_position='c'
  zbot_slice=0.
  ztop_slice=0.2
```

In this case, the slices with the suffix \( .xy \) will be at the “surface” and the ones with the suffix \( .xy2 \) will be at the position \( z = 0.2 \) above the surface. And you can visualize this slices by:

1. Write the following lines to ‘video.in’ in your run directory:

   ```plaintext
   bb
   ```

2. Edit the namelist `run_pars` in the file ‘run.in’ to include `zbot_slice` and `ztop_slice`.

3. Run the PENCIL CODE:

   ```plaintext
   unix> start.csh
   unix> run.csh
   ```

4. Run ‘src/read_videofiles.x’ to assemble global slice files from those scattered across ‘data/proc*/’:

   ```plaintext
   unix> src/read_videofiles.x
   enter name of variable (lnrho, uu1, ..., bb3):  bb3
   ```

5. Start IDL, load the slices with ‘pc_read_video’ and plot them at some time:

   ```plaintext
   unix> idl
   IDL> pc_read_video, field='bb3',cb=bb3,nt=ntv
   IDL> tvscl,bb3.xy(*,*,100)
   IDL> tvscl,bb3.xy2(*,*,100)
   ```

   etc.

File structure  Slice files consist of one Fortran record (see footnote on page 26) for each slice, which contains the data of the variable (without ghost zones), the time \( t \) of the snapshot and the position of the sliced variable (e.g. the \( x \) position for a \( y-z \) slice):

1. \( data_i [nx \times ny \times nz] \), time \( t_i [1] \), position\( _i [1] \)
2. data\textsubscript{2} \([nx \times ny \times nz]\), time \(t_2\) [1], position\textsubscript{2} [1]

3. data\textsubscript{3} \([nx \times ny \times nz]\), time \(t_3\) [1], position\textsubscript{3} [1]

etc.

5.8 Averages

5.8.1 One-dimensional output averaged in two dimensions

In the file ‘xyaver.in’, \(z\)-dependent (horizontal) averages are listed. They are written to the file ‘data/xyaverages.dat’. A new line of averages is written every \(it1\)th time steps.

There is the possibility to output two-dimensional averages. The result then depends on the remaining dimension. The averages are listed in the files ‘xyaver.in’, ‘xzaver.in’, and ‘yzaver.in’ where the first letters indicate the averaging directions. The output is then stored to the files ‘data/xyaverages.dat’, ‘data/xzaverages.dat’, and ‘data/yzaverages.dat’. The output is written every \(it1d\)th time steps.

The rms values of the so defined mean magnetic fields are referred to as \(bmz\), \(bmy\) and \(bmx\), respectively, and the rms values of the so defined mean velocity fields are referred to as \(umz\), \(umy\), and \(umx\). (The last letter indicates the direction on which the averaged quantity still depends.)

See Sect. 9.2 on how to add new averages.

In IDL such \(xy\)-averages can be read using the procedure ‘pc_read_xyaver’.

5.8.2 Two-dimensional output averaged in one dimension

There is the possibility to output one-dimensional averages. The result then depends on the remaining two dimensions. The averages are listed in the files ‘yaaver.in’, ‘zaver.in’, and ‘phiaver.in’ where the first letter indicates the averaging direction. The output is then stored to the files ‘data/yaaverages.dat’, ‘data/zaverages.dat’, and ‘data/phiaverages.dat’.

See Sect. 9.2 on how to add new averages.

Disadvantage: The output files, e.g. ‘data/zaverages.dat’, can be rather big because each average is just appended to the file.

5.8.3 Azimuthal averages

Azimuthal averages are controlled by the file ‘phiaver.in’, which currently supports the quantities listed in Sect. J.5. In addition, one needs to set \(lwrite\_phiaverages\), \(lwrite\_yaaverages\), or \(lwrite\_zaverages\) to .true.. For example, if ‘phiaver.in’ contains the single line

\[
b2mphi
\]

then you will get azimuthal averages of the squared magnetic field \(B^2\).
Azimuthal averages are written every $d2davg$ time units to the files 'data/averages/PHIAVG'. The file format of azimuthal-average files consists of the following Fortran records:

1. number of radial points $N_{r,\phi-avg}$ [1], number of vertical points $N_{z,\phi-avg}$ [1], number of variables $N_{var,\phi-avg}$ [1], number of processors in $z$ direction [1]

2. time $t$ [1], positions of cylindrical radius $r_{cyl}$ [$N_{r,\phi-avg}$] and $z$ [$N_{z,\phi-avg}$] for the grid, radial spacing $\delta r_{cyl}$ [1], vertical spacing $\delta z$ [1]

3. averaged data [$N_{r,\phi-avg} \times N_{z,\phi-avg}$]

4. label length [1], labels of averaged variables [$N_{var,\phi-avg}$]

All numbers are 4-byte numbers (floating-point numbers or integers), unless you use double precision (see §5.21).

To read and visualize azimuthal averages in IDL, use ‘$PENCIL_HOME/idl/files/pencil_read_phiavg.pro’

IDL> avg = pc_read_phiavg('data/averages/PHIAVG1')
IDL> contour, avg.b2mphi, avg.rcyl, avg.z, TITLE='!17B!U2!N!X'

or have a look at ‘$PENCIL_HOME/idl/phiavg.pro’ for a more sophisticated example.

5.8.4 Time averages

Time averages need to be prepared in the file ‘src/ctimeavg.local’, since they use extra memory. They are controlled by the averaging time $\tau_{avg}$ (set by the parameter $tavg$ in ‘run.in’), and by the indices $idx_{tavg}$ of variables to average.

Currently, averaging is implemented as exponential (memory-less) average

$\langle f \rangle_{t+\delta t} = \langle f \rangle_t + \frac{\delta t}{\tau_{avg}} [f(t+\delta t) - \langle f \rangle_t], \quad (16)$

which is equivalent to

$\langle f \rangle_t = \int_{t_0}^{t} e^{-(t-t')/\tau_{avg}} f(t') \, dt'. \quad (17)$

Here $t_0$ is the time of the snapshot the calculation started with, i.e. the snapshot read by the last run.x command. Note that the implementation (16) will approximate Eq. (17) only to first-order accuracy in $\delta t$. In practice, however, $\delta t$ is small enough to make this accuracy suffice.

At some point we may also implement the more straight-forward average

$\langle f \rangle_{t+\delta t} = \langle f \rangle_t + \frac{\delta t}{t-t_0+\delta t} [f(t+\delta t) - \langle f \rangle_t], \quad (14)$

which is equivalent to

$\langle f \rangle_t = \frac{1}{t-t_0} \int_{t_0}^{t} f(t') \, dt'. \quad (15)$

but we do not expect large differences.
In `src/ctimeavg.local`, you need to set the number of slots used for time averages. Each of these slots uses $mx \times my \times mz$ floating-point numbers, i.e. half as much memory as each fundamental variable.

For example, if you want to get time averages of all variables, set

```
integer, parameter :: mtavg=mvar
```

in `src/ctimeavg.local`, and don’t set `idx_tavg` in `run.in`.

If you are only interested in averages of variables 1–3 and 6–8 (say, the velocity vector and the magnetic vector potential in a run with `hydro.f90`, `density.f90`, `entropy.f90` and `magnetic.f90`), then set

```
integer, parameter :: mtavg=6
```

in `src/ctimeavg.local`, and set

```
idx_tavg = 1,2,3,6,7,8  ! time-average velocity and vector potential
```

in `run.in`.

Permanent snapshots of time averages are written every $tavg$ time units to the files `data/proc*/TAV*`. The current time averages are saved periodically in `data/proc*/timeavg.dat` whenever `data/proc*/var.dat` is written. The file format for time averages is equivalent to that of the snapshots; see §5.6.1 above.

### 5.9 Helper scripts

The `bin` directory contains a collection of utility scripts, some of which are discussed elsewhere. Here is a list of the more important ones.

**adapt-mkfile** Activate the settings in a `Makefile` that apply to the given computer, see §5.2.

**auto-test** Verify that the code compiles and runs in a set of run directories and compare the results to the reference output. These tests are carried out routinely to ensure that the svn version of the code is in a usable state.

**cleanf95** Can be use to clean up the output from the Intel x86 Fortran 95 compiler (ifc).

**copy-proc-to-proc** Used for restarting in a different directory. Example `copy-proc-to-proc seed.dat ../hydro256e`.

**copy-snapshots** Copy snapshots from a processor-local directory to the global directory. To be started in the background before `run.x` is invoked. Used by `start.csh` and `run.csh` on network connected processors.

**pc_copyvar var1 var2 source dest** Copies snapshot files from one directory (source) to another (dest). See documentation in file.

**pc_copyvar v v dir** Copies all `var.dat` files from current directory to `var.dat` in `dir` run directory. Used for restarting in a different directory.
pc_copyvar N v  Used to restart a run from a particular snapshot ‘VAR.N’. Copies a
specified snapshot ‘VAR.N’ to ‘var.dat’ where N and (optionally) the target run di-
rectory are given on the command line.

cvs-add-rundir  Add the current run directory to the svn repository.

cvsci_run  Similar to cvs-add-rundir, but it also checks in the ‘*.in’ and ‘src/*.local’
files. It also checks in the files ‘data/time_series.dat’, ‘data/dim.dat’ and
‘data/index.pro’ for subsequent processing in IDL on another machine. This is
particularly useful if collaborators want to check each other’s runs.

dx*  These script perform several data collection or reformatting exercises required to
read particular files into DX. They are called internally by some of the DX macros
in the ‘dx/macros/’ directory.

getconf.csh  See §4.3.4

gpgrowth  Plot simple time evolution with Gnuplot’s ASCII graphics for fast orienta-
tion via a slow modem line.

local  Materialize a symbolic link

mkcpparam  Based on ‘Makefile’ and ‘Makefile.local’, generate ‘src/cparam.inc’,
which specifies the number mvar of fundamental variables, and maux of auxiliary
variables. Called by the ‘Makefile’.

pc_mkdatadir  Creates a link to a data directory in a suitable workspace. By default
this is on ‘/var/tmp/’, but different locations are specified for different machines.

mkdotin  Generate minimal ‘start.in’, ‘run.in’ files based on ‘Makefile’ and
‘Makefile.local’.

mkinpars  Wrapper around ‘mkdotin’ — needs proper documentation.

mkproc-tree  Generates a multi-processor(‘procN’), directory structure. Useful when
copying data files in a processor tree, such as slice files.

mkwww  Generates a template HTML file for describing a run of the code, showing
input parameters and results.

move-slice  Moves all the slice files from a processor tree structure, ‘procN’, to a new
target tree creating directories where necessary.

nl2idl  Transform a Fortran namelist (normally the files ‘param.nml’, ‘param2.nml’ writ-
ten by the code) into an IDL structure. Generates an IDL file that can be sourced
from ‘start.pro’ or ‘run.pro’.

pacx-adapt-makefile  Version of adapt-makefile for highly distributed runs using
PACX MPI.

pc_newrun  Generates a new run directory from an old one. The new one contains a
copy of the old ‘*.local’ files, runs pc_setupsrc, and makes also a copy of the old
‘*.in’ and ‘k.dat’ files.

pc_newscan  Generates a new scan directory from an old one. The new one contains
a copy of the old, e.g. the one given under ‘samples/parameter_scan’. Look in the
‘README’ file for details.
5.10 RELOAD and STOP files

The code periodically (every \( \delta t \) time steps) checks for the existence of two files, ‘RELOAD’ and ‘STOP’, which can be used to trigger certain behavior.

Reloading run parameters In the directory where you started the code, create the file ‘RELOAD’ with

\[
\text{unix} > \text{touch RELOAD}
\]
to force the code to re-read the runtime parameters from ‘run.in’. This will happen the next time the code is writing monitoring output (the frequency of this happening is controlled by the input parameter \(it\), see Sect. 5.12).

Each time the parameters are reloaded, the new set of parameters is appended (in the form of namelists) to the file ‘data/params.log’ together with the time \(t\), so you have a full record of your changes. If ‘RELOAD’ contains any text, its first line will be written to ‘data/params.log’ as well, which allows you to annotate changes:

```
unix> echo "Reduced eta to get fields growing" > RELOAD
```

Use the command `summarize-history` to print a history of changes.

**Stopping the code**  In the directory where you started the code, create the file ‘STOP’ with

```
unix> touch STOP
```

to stop the code in a controlled manner (it will write the latest snapshot). Again, the action will happen the next time the code is writing monitoring output.

### 5.11 RERUN and NEWDIR files

After the code finishes (e.g., when the final timestep number is reached or when a ‘STOP’ file is found), the ‘run.csh’ script checks whether there is a ‘RERUN’ file. If so, the code will simply run again, perhaps even after you have recompiled the code. This is useful in the development phase when you changed something in the code, so you don’t need to wait for a new slot in the queue!

Even more naughty, as Tony says, is the ‘NEWDIR’ file, where you can enter a new directory path (relative path is ok, e.g. ‘../conv-slab’). If nothing is written in this file (e.g. via `touch NEWDIR`) it stays in the same directory. On distributed machines, the ‘NEWDIR’ method will copy all the ‘VAR\#' and ‘var.dat’ files back to and from the server. This can be useful if you want to run with new data files, but you better do it in a separate directory, because with ‘NEWDIR’ the latest data from the code are written back to the server before running again.

Oh, by the way, if you want to be sure that you haven’t messed up the content of the pair of ‘NEWDIR’ files, you may want to try out the `pc_jobtransfer` command. It writes the decisive ‘STOP’ file only after the script has checked that the content of the two ‘NEWDIR’ files points to existing run directory paths, so if the new run crashes, the code returns safely to the old run directory. I’m not sure what Tony would say now, but this is now obviously extremely naughty.

### 5.12 Start and run parameters

All input parameters in ‘start.in’ and ‘run.in’ are grouped in Fortran namelists. This allows arbitrary order of the parameters (within the given namelist; the namelists need no longer be in the correct order), as well as enhanced readability through inserted Fortran comments and whitespace. One namelist (init_pars / run_pars) contains general parameters for initialization/running and is always read in. All other namelists are specific to individual modules and will only be read if the corresponding module is used.
The syntax of a namelist (in an input file like ‘start.in’) is

```
&init_pars
 ip=5, Lxyz=2,4,2
 /
```

— in this example, the name of the namelist is `init_pars`, and we read just two variables (all other variables in the namelist retain their previous value): `ip`, which is set to 5, and `Lxyz`, which is a vector of length three and is set to (2, 4, 2).

While all parameters from the namelists can be set, in most cases reasonable default values are preset. Thus, the typical file ‘start.in’ will only contain a minimum set of variables or (if you are very minimalistic) none at all. If you want to run a particular problem, it is best to start by modifying an existing example that is close to your application.

Before starting a simulation run, you may want to execute the command `pc_configtest` in order to test the correctness of your changes to these configuration files.

As an example, we give here the start parameters for ‘samples/helical-MHDturb’

```
&init_pars
 cvsdi='$Id:$', ! identify version of start.in
 xyz0 = -3.1416, -3.1416, -3.1416, ! first corner of box
 Lxyz = 6.2832, 6.2832, 6.2832, ! box size
 lperi = T , T , T , ! periodic in x, y, z
 random_gen='nr_f90'
 /
 &hydro_init_pars
 /
 &density_init_pars
 gamma=1.
 /
 &magnetic_init_pars
 initaa='gaussian-noise', amplaa=1e-4
 /
```

The three entries specifying the location, size and periodicity of the box are just given for demonstration purposes here — in fact a periodic box from $-\pi$ to $-\pi$ in all three directions is the default. In this run, for reproducibility, we use a random number generator from the Numerical Recipes [27], rather than the compiler's built-in generator. The adiabatic index $\gamma$ is set explicitly to 1 (the default would have been 5/3) to achieve an isothermal equation of state. The magnetic vector potential is initialized with uncorrelated, normally distributed random noise of amplitude $10^{-4}$.

The run parameters for ‘samples/helical-MHDturb’ are

```
&run_pars
 cvsdi='$Id:$', ! identify version of start.in
 nt=10, it1=2, cdt=0.4, cdtv=0.80, isave=10, itorder=3
 dsnap=50, dvid=0.5
 random_gen='nr_f90'
 /
 &hydro_run_pars
 /
```
Here we run for \( nt = 10 \) timesteps, every second step, we write a line of diagnostic output; we require the time step to keep the advective Courant number \( \leq 0.4 \) and the diffusive Courant number \( \leq 0.8 \), save 'var.dat' every 20 time steps, and use the 3-step time-stepping scheme described in Appendix [H.4](#) (the Euler scheme \( itorder = 1 \) is only useful for tests). We write permanent snapshot file 'VAR.N' every \( dsnap = 50 \) time units and 2d slices for animation every \( dvid = 0.5 \) time units. Again, we use a deterministic random number generator. Viscosity \( \nu \) and magnetic diffusivity \( \eta \) are set to \( 5 \times 10^{-3} \) (so the mesh Reynolds number is about \( u_{\text{rms}}\delta x/\nu = 0.3 \times (2\pi/32)/5 \times 10^{-3} \approx 12 \), which is in fact rather a bit to high). The parameters in \textit{forcing_run_pars} specify fully helical forcing of a certain amplitude.

A full list of input parameters is given in Appendix [J](#).

### 5.13 Physical units

Many calculations are unit-agnostic, in the sense that all results remain the same independent of the unit system in which you interpret the numbers. E. g. if you simulate a simple hydrodynamical flow in a box of length \( L = 1 \) and get a maximum velocity of \( u_{\text{max}} = 0.5 \) after \( t = 3 \) time units, then you may interpret this as \( L = 1 \text{ m} \), \( u_{\text{max}} = 0.5 \text{ m/s} \), \( t = 3 \text{ s} \), or as \( L = 1 \text{ pc} \), \( u_{\text{max}} = 0.5 \text{ pc/Myr} \), \( t = 3 \text{ Myr} \), depending on the physical system you have in mind. The units you are using must of course be consistent, thus in the second example above, the units for diffusivities would be \( \text{pc}^2/\text{Myr} \), etc.

The units of magnetic field and temperature are determined by the values \( \mu_0 = 1 \) and \( c_p = 1 \) used internally by the code\(^7\). This means that if your units for density and velocity are \([\rho]\) and \([v]\), then magnetic fields will be in

\[
[B] = \sqrt{\mu_0 [\rho] [v]^2},
\]

and temperatures are in

\[
[T] = \frac{[v]^2}{c_p} = \frac{\gamma-1}{\gamma} \frac{[v]^2}{R/\mu}.
\]

For some choices of density and velocity units, Table 2 shows the resulting units of magnetic field and temperature.

\(^7\) Note that \( c_p = 1 \) is only assumed if you use the module \texttt{noionization.f90}. If you work with \texttt{ionization.f90}, temperature units are specified by \texttt{unit_temperature} as described below.
Table 2: Units of magnetic field and temperature for some choices of \([\rho]\) and \([v]\) according to Eqs. (18) and (19). Values are for a monatomic gas \((\gamma = 5/3)\) of mean atomic weight \(\bar{\mu}_g = \bar{\mu}/1\,\text{g}\) in grams.

<table>
<thead>
<tr>
<th>([\rho])</th>
<th>([v])</th>
<th>([B])</th>
<th>([T])</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 kg/m(^3)</td>
<td>1 m/s</td>
<td>1.12 mT = 11.2 G</td>
<td>((\bar{\mu}_g/0.6) \times 2.89 \times 10^{-5}) K</td>
</tr>
<tr>
<td>1 g/cm(^3)</td>
<td>1 cm/s</td>
<td>3.54 \times 10^{-4} T = 3.54 G</td>
<td>((\bar{\mu}_g/0.6) \times 2.89) nK</td>
</tr>
<tr>
<td>1 g/cm(^3)</td>
<td>1 km/s</td>
<td>35.4 T = 354 kG</td>
<td>((\bar{\mu}_g/0.6) \times 28.9) K</td>
</tr>
<tr>
<td>1 g/cm(^3)</td>
<td>10 km/s</td>
<td>354 T = 3.54 MG</td>
<td>((\bar{\mu}_g/0.6) \times 2890) K</td>
</tr>
</tbody>
</table>

On the other hand, as soon as material equations are used (e.g. one of the popular parameterizations for radiative losses, Kramers opacity, Spitzer conductivities or ionization, which implies well-defined ionization energies), the corresponding routines in the code need to know the units you are using. This information is specified in `start.in` or `run.in` through the parameters `unit_system`, `unit_length`, `unit_velocity`, `unit_density` and `unit_temperature` like e.g.

```
unit_system='SI',
unit_length=3.09e16, unit_velocity=978. ! [l]=1pc, [v]=1pc/Myr
```

Note: The default unit system is `unit_system='cgs'` which is a synonym for `unit_system='Babylonian cubits'`.

5.14 Minimum amount of viscosity

We emphasize that, by default, the code works with constant diffusion coefficients (viscosity \(\nu\), thermal diffusivity \(\chi\), magnetic diffusivity \(\eta\), or passive scalar diffusivity \(D\)). If any of these numbers is too small, you would need to have more meshpoints to get consistent numerical solutions; otherwise the code develops wiggles (‘ringing’) and will eventually crash. A useful criterion is given by the mesh Reynolds number based on the maximum velocity,

\[
Re_{\text{mesh}} = \max(|u|) \max(\delta x, \delta y, \delta z)/\nu,
\]

which should not exceed a certain value which can be problem-dependent. Often the largest possible value of \(Re_{\text{mesh}}\) is around 5. Similarly there exist mesh Péclet and mesh magnetic Reynolds numbers that should not be too large.

Note that in some cases, ‘wiggles’ in \(\ln \rho\) will develop despite sufficiently large diffusion coefficients, essentially because the continuity equation contains no dissipative term. For convection runs (but not only for these), we have found that this can often be prevented by upwinding, see Sect. [H.2]

If the Mach number of the code approaches unity, i.e. if the rms velocity becomes comparable with the speed of sound, shocks may form. In such a case the mesh Reynolds number should be smaller. In order to avoid excessive viscosity in the unshocked regions,

---

\(^8\) Note: the parameter `unit_temperature` is currently only used in connection with `ionization.f90`. If you are working with `noionization.f90`, the temperature unit is completely determined by Eq. (19) above.
one can use the so-called shock viscosity (Sect. 6.6.1) to concentrate the effects of a low mesh Reynolds number to only those areas where it is necessary.

5.15 The time step

5.15.1 The usual RK-2N time step

RK-2N refers to the third order Runge-Kutta scheme by Williamson (1980) with a memory consumption of two chunks. Therefore the 2N in the name.

The time step is normally specified as Courant time step through the coefficients \( cd_t \) (\( c_{dt} \)), \( cd_t v \) (\( c_{dtv} \)) and \( cd_t s \) (\( c_{dts} \)). The resulting Courant step is given by

\[
\delta t = \min \left( \frac{\delta x_{\text{min}}}{U_{\text{max}}}, \frac{\delta x_{\text{min}}^2}{D_{\text{max}}}, \frac{1}{H_{\text{max}}} \right),
\]

where

\[
\delta x_{\text{min}} \equiv \min(\delta x, \delta y, \delta z); \tag{22}
\]

\[
U_{\text{max}} \equiv \max \left( |u| + \sqrt{c_s^2 + v_A^2} \right), \tag{23}
\]

c\(_s\) and \( v_A \) denoting sound speed and Alfvén speed, respectively;

\[
D_{\text{max}} = \max(\nu, \gamma\chi, \eta, D), \tag{24}
\]

where \( \nu \) denotes kinematic viscosity, \( \chi = K/(c_p\rho) \) thermal diffusivity and \( \eta \) the magnetic diffusivity; and

\[
H_{\text{max}} = \max \left( \frac{2\nu S^2 + \zeta_{\text{shock}}(\nabla \cdot u)^2 + \ldots}{c_v T} \right), \tag{25}
\]

where dots indicate the presence of other terms on the rhs of the entropy equation.

To fix the time step \( \delta t \) to a value independent of velocities and diffusivities, explicitly set the run parameter \( dt \), rather than \( cd_t \) or \( cd_t v \) (see p.172).

If the time step exceeds the viscous time step the simulation may actually run ok for quite some time. Inspection of images usually helps to recognize the problem. An example is shown in Fig. 4.

Timestepping is accomplished using the Runge-Kutta 2N scheme. Regarding details of this scheme see Sect. [H.4]

5.15.2 The Runge-Kutta-Fehlberg time step

A fifth order Runge-Kutta-Fehlberg time stepping procedure is available. It is used mostly for chemistry application, often together with the double precision option. In order to make this work, you need to compile with

\[
\text{TIMESTEP} = \text{timestep_rkf}
\]

in ‘src/Makefile.local’. In addition, you must put itorder=5 in ‘run.in’. An example application is ‘samples/1d-tests/H2_flamespeed’. This procedure is still experimental.
5.16 Boundary conditions

5.16.1 Where to specify boundary conditions

In most tests that come with the PENCIL CODE, boundary conditions are set in ‘run.in’, which is a natural choice. However, this may lead to unexpected initial data written by ‘start.x’, since when you start the code (via ‘start.csh’), the boundary conditions are unknown and ‘start.x’ will then fill the ghost zones assuming periodicity (the default boundary condition) in all three directions. These ghost data will never be used in a calculation, as ‘run.x’ will apply the boundary conditions before using any ghost-zone values.

To avoid these periodic conditions in the initial snapshot, you can set the boundary conditions in ‘start.in’ already. In this case, they will be inherited by ‘run.x’, unless you also explicitly set boundary conditions in ‘run.in’.

5.16.2 How to specify boundary conditions

Boundary conditions are implemented through three layers of ghost points on either boundary, which is quite a natural choice for an MPI code that uses ghost zones for representing values located on the neighboring processors anyway. The desired type of boundary condition is set through the parameters $bc\{x,y,z\}$ in ‘run.in’; the nomenclature used is as follows. Set $bc\{x,y,z\}$ to a sequence of letters like

- $bcx = \{'p','p','p','p','p'\}$
- $bcx = \{'s','s','a','a2','c1:c2'\}$

for periodic boundaries, or

for non-periodic ones. Each element corresponds to one of the variables, which are those of the variables $u_x, u_y, u_z, \ln \rho, s/c_p, A_x, A_y, A_z, \ln c$ that are actually used in this order. The following conditions are available:

- ‘p’ periodic boundary condition
- ‘a’ antisymmetric condition w.r.t. the boundary, i.e. vanishing value

Figure 4: Example of a velocity slice from a run where the time step is too long. Note the spurious checkerboard modulation in places, for example near $x = -0.5$ and $-2.5 < y < -1.5$. This is an example of a hyperviscous turbulence simulations with $512^3$ meshpoints and a third order hyperviscosity of $\nu_3 = 5 \times 10^{-12}$. Hyperviscosity is explained in the Appendix E.
‘s’ symmetric condition w. r. t. the boundary, i. e. vanishing first derivative

‘a2’ antisymmetry w. r. t. the arbitrary value on the boundary, i. e. vanishing second derivative

‘c1’ special boundary condition for \( \ln \rho \) and \( s \): constant heat flux through the boundary

‘c2’ special boundary condition for \( s \): constant temperature at the boundary — requires boundary condition \( a2 \) for \( \ln \rho \)

‘cT’ special boundary condition for \( s \) or \( \ln T \): constant temperature at the boundary (for arbitrarily set \( \ln \rho \))

‘ce’ special boundary condition for \( s \): set temperature in ghost points to value on boundary (for arbitrarily set \( \ln \rho \))

‘db’ low-order one-sided derivatives (“no boundary condition”) for density

‘she’ shearing-sheet boundary condition (default when the module Shear is used)

‘g’ force the value of the corresponding field on vertical boundaries (should be used in combination with the force_lower_bound and force_upper_bound flags set in the namelist init.pars)

‘hs’ special boundary condition for \( \ln \rho \) and \( s \) which enforces hydrostatic equilibrium on vertical boundaries

The extended syntax \( a:b \) (e. g. ‘c1:c2’) means: use boundary condition \( a \) at the left/lower boundary, but \( b \) at the right/upper one.

If you build a new ‘run.in’ file from another one with a different number of variables (noentropy vs. entropy, for example), you need to remember to adjust the length of the arrays bcx to bcz. The advantage of the present approach is that it is very easy to exchange all boundary conditions by a new set of conditions in a particular direction (for example, make everything periodic, or switch off shearing sheet boundary conditions and have stress-free instead).

### 5.17 Restarting a simulation

When a run stops at the end of a simulation, you can just resubmit the job again, and it will start from the latest snapshot saved in ‘data/proc*/var.dat’. The value of the latest time is saved in a separate file, ‘data/proc*/time.dat’. On parallel machines it is possible that some (or just one) of the ‘var.dat’ are corrupt; for example after a system crash. Check for file size and date, and restart from a good ‘VAR’ file instead.

If you want to run on a different machine, you just need to copy the ‘data/proc*/var.dat’ (and, just to be sure) ‘data/proc*/time.dat’) files into a new directory tree. You may also need the ‘data/proc*/seed.dat’ files for the random number generator. The easiest way to get all these other files is to run start.csh again on the new machine (or in a new directory) and then to overwrite the ‘data/proc*/var.dat’ files with the correct ones.

For restarting from runs that didn’t have magnetic fields, passive scalar fields, or test fields, see Sect. F.3
5.18 One- and two-dimensional runs

If you want to run two-dimensional problems, set the number of mesh points in one direction to unity, e.g. \( \text{nygrid}=1 \) or \( \text{nzgrid}=1 \) in ‘cparam.local’. Remember that the number of mesh points is still divisible by the number of processors. For 2D-runs, it is also possible to write only 2D-snapshots (i.e. VAR files written only in the considered \((x,y)\) or \((x,z)\) plane, with a size seven times smaller as we do not write the third unused direction). To do that, please add the logical flag ‘lwrite_2d=T’ in the namelist init.pars in ‘start.in’.

Similarly, for one-dimensional problems, set, for example, \( \text{nygrid}=1 \) and \( \text{nzgrid}=1 \) in ‘cparam.local’. You can even do a zero-dimensional run, but then you better set \( dt \) (rather than \( cdt \)), because there is no Courant condition for the time step.

See 0d, 1d, 2d, and 3d tests with examples.

5.19 Visualization

5.19.1 Gnuplot

Simple visualization can easily be done using "Gnuplot" ([http://www.gnuplot.info](http://www.gnuplot.info)), an open-source plotting program suitable for two-dimensional plots.

For example, suppose you have the variables

```
---it-----t-------dt-------urms-----umax-----rhom-----ssm------dtc---
```

in ‘time_series.dat’ and want to plot \( u_{rms}(t) \). Just start gnuplot and type

```
gnuplot> plot "data/time_series.dat" using 2:4 with lines
```

If you work over a slow line and want to see both \( u_{rms}(t) \) and \( u_{max}(t) \), use ASCII graphics:

```
gnuplot> set term dump
gnuplot> set logscale y
gnuplot> plot "data/time_series.dat" using 2:4 title "urms",
     "data/time_series.dat" using 2:5 title "umax"
```

5.19.2 Data explorer


The PENCIL CODE provides a few networks for "DX". It is quite easy to read in a snapshot file from "DX" (the only tricky thing is the four extra bytes at the beginning of the file, representing a Fortran record marker), and whenever you run ‘start.x’, the code writes a file ‘var.general’ that tells "DX" all it needs to know about the data structure.

As a starting point for developing your own "DX" programs or networks, you can use a few generic "DX" scripts provided in the directory ‘dx/basic/’. From the run directory, start "DX" with

```
unix> dx -edit $PENCIL_HOME/dx/basic/lnrho
```
to load the file ‘dx/basic/lnrho.net’, and execute it with (Ctrl-o) or Execute \rightarrow Execute
Once. You will see a set of iso-surfaces of logarithmic density. If the viewport does not
fit to your data, you can reset it with (Ctrl-f). To rotate the object, drag the mouse over
the Image window with the left or right mouse button pressed. Similar networks are
provided for entropy (‘ss.net’), velocity (‘uu.net’) and magnetic field (‘bb.net’).

When you expand these simple networks to much more elaborate ones, it is probably
a good idea to separate the different tasks (like Importing and Selecting, visualizing
velocity, visualizing entropy, and Rendering) onto separate pages through Edit \rightarrow Page.

Note Currently, DX can only read in data files written by one single processor, so from
a multi-processor run, you can only visualize one subregion at a time.

5.19.3 GDL

GDL, also known as Gnu Data Language is a free visualization package that can be
found at {http://gnudatalanguage.sourceforge.net/}. It aims at replacing the very ex-
pensive IDL package (see S. 5.19.4). For the way we use IDL for the Pencil Code, com-
patibility is currently not completely sufficient, but you can use GDL for many of the
visualization tasks. If you get spurious “Error opening file” messages, you can normally
simply ignore them.

This section tells you how to get started with using GDL for visualization.

Setup As of GDL 0.9 – at least the version packed with Ubuntu Jaunty (9.10) – you
will need to add GDL’s ‘examples/pro’ directory to your PATH variable. So the first call
after starting GDL should be

```
GDL> .run setup_gdl
```

Starting visualization There are mainly two possibilities for visualization: using a sim-
ple GUI or loading the data with pc_read and work with it interactively. Please note that
the GUI was written and tested only with IDL, see §5.19.4.

Here, the pc_read family of IDL routines to read the data is described. Try

```
GDL> pc_read
```
to get an overview.

To plot a time series, use

```
GDL> pc_read_ts, OBJECT=ts
GDL> help, ts, /STRUCT ;; (to see which slots are available)
GDL> plot, ts.t, ts.umax
GDL> oplot, ts.t, ts.urms
```

Alternatively, you could simply use the ‘ts.pro’ script:

```
GDL> .run ts
```

To work with data from ‘var.dat’ and similar snapshot files, you can e.g. use the following routines:
Having thus read the data structures, we can have a look at them to see what information is available:

GDL> help, dim, /STRUCT
GDL> help, par, /STRUCT
GDL> help, grid, /STRUCT
GDL> help, var, /STRUCT

To visualize data, we can e.g. do

GDL> plot, grid.x, var.ss[*], dim.ny/2, dim.nz/2
GDL> contourfill, var.ss[*], *, dim.nz/2], grid.x, grid.y

GDL> ux_slice = var.uu[*], *, dim.nz/2, 0]
GDL> uy_slice = var.uu[*], *, dim.nz/2, 1]
GDL> wdvelovect, ux_slice, uy_slice, grid.x, grid.y

GDL> surface, var.lnrho[*], *, dim.nz/2, 0]

See also Sect. 5.19.4

5.19.4 IDL

IDL is a commercial visualization program for two-dimensional and simple threedimensional graphics. It allows to access and manipulate numerical data in a fashion quite similar to how Fortran handles them.

In ‘$PENCIL_HOME/idl’, we provide a number of general-purpose IDL scripts that we are using all the time in connection with the PENCIL CODE. While IDL is quite an expensive software package, it is quite useful for visualizing results from numerical simulations. In fact, for many applications, the 7-minute demo version of IDL is sufficient.

Visualization in IDL  The Pencil Code GUI is a data post-processing tool for the usage on a day-to-day basis. It allows fast inspection of many physical quantities, as well as advanced features like horizontal averages, streamline tracing, freely orientable 2D-slices, and extraction of cut images and movies. To use the Pencil Code GUI, it is sufficient to run:

unix> idl
IDL> .r pc_gui

If you like to load only some subvolume of the data, like any 2D-slices from the given data snapshots, or 3D-subvolumes, it is possible to choose the corresponding options in the file selector dialog. The Pencil Code GUI offers also some options to be set on the command-line, please refer to their description in the source code.

9 If contourfill produces just contour lines instead of a color-coded plot, your version of GDL is too old. E.g. the version shipped with Ubuntu 9.10 is based on GDL 0.9rc1 and has this problem.
There are also other small GUIs available, e.g. the file ‘time-series.dat’ can easily be analyzed with the command:

```
unix> idl
IDL> pc_show_ts
```

The easiest way to derive physical quantities at the command-line is to use one of the many `pc_read_var`-variants (`pc_read_var_raw` is recommended for large datasets because of its high efficiency regarding computation and memory usage) for reading the data. With that, one can make use of `pc_get_quantity` to derive any implemented physical quantity. Packed in a script, this is the recommended way to get reproducible results, without any chance for accidental errors on the interactive IDL command-line.

Alternatively, by using the command-line to see the time evolution of e.g. velocity and magnetic field (if they are present in you run), start `IDL` and run ‘ts.pro’:

```
unix> idl
IDL> .run ts
```

The IDL script ‘ts.pro’ reads the time series data from ‘data/time_series.dat’ and sorts the column into the structure `ts`, with the slot names corresponding to the name of the variables (taken from the header line of ‘data/time_series.dat’). Thus, you can refer to time as `ts.t`, to the rms velocity as `ts.urms`, and in order to plot the mean density as a function of time, you would simply type

```
IDL> plot, ts.t, ts.rhom
```

The basic command sequence for working with a snapshot is:

```
unix> idl
IDL> .run start
IDL> .run r
IDL> [specific commands]
```

You run ‘start.pro’ once to initialize (or reinitialize, if the mesh size has changed, for example) the fields and read in the startup parameters from the code. To read in a new snapshot, run ‘r.pro’ (or ‘rall.pro’, see below).

If you are running in parallel on several processors, the data are scattered over different directories. To reassemble everything in IDL, use

```
unifx> idl
IDL> .run start
IDL> .run r
IDL> [specific commands]
```

You run ‘start.pro’ once to initialize (or reinitialize, if the mesh size has changed, for example) the fields and read in the startup parameters from the code. To read in a new snapshot, run ‘r.pro’ (or ‘rall.pro’, see below).

If you are running in parallel on several processors, the data are scattered over different directories. To reassemble everything in IDL, use

```
unifx> idl
IDL> .run start
IDL> .run r
IDL> [specific commands]
```

You run ‘start.pro’ once to initialize (or reinitialize, if the mesh size has changed, for example) the fields and read in the startup parameters from the code. To read in a new snapshot, run ‘r.pro’ (or ‘rall.pro’, see below).

If you run IDL from the command line, you will highly appreciate the following tip: IDL’s command line editing is broken beyond hope. But you can fix it, by running IDL under `rlwrap`, a wrapper for the excellent GNU `readline` library.

Download and install `rlwrap` from [http://utopia.knoware.nl/~hlub/uck/rlwrap/](http://utopia.knoware.nl/~hlub/uck/rlwrap/) (on some systems you just need to run ‘emerge rlwrap’, or ‘apt-get install rlwrap’), and alias your `idl` command:

```
csh> alias idl 'rlwrap -a -c idl'
bash> alias idl='rlwrap -a -c idl'
```

From now on, you can

- use long command lines that correctly wrap around;
- type the first letters of a command and then `PageUp` to recall commands starting with these letters;
- capitalize, uppercase or lowercase a word with `Esc`-C, `Esc`-U, `Esc`-L;
- use command line history across IDL sessions (you might need to create ‘~/idl_history’ for this);
- complete file names with `Tab` (works to some extent);
- ... use all the other `readline` features that you are using in bash, octave, bc, gnuplot, ftp, etc.
instead of \texttt{.r} \texttt{r} (here, \texttt{.r} is a shorthand for \texttt{.run}). The procedure `\texttt{rall.pro}` reads (and assembles) the data from all processors and correspondingly requires large amounts of memory for very large runs. If you want to look at just the data from one processor, use `\texttt{.r.pro}` instead.

If you need the magnetic field or the current density, you can calculate them in IDL by

\begin{verbatim}
IDL> bb=curl(aa)
IDL> jj=curl2(aa)
\end{verbatim}

By default one is reading always the current snapshot `\texttt{data/procN/var.dat}`; if you want to read one of the permanent snapshots, use (for example)

\begin{verbatim}
IDL> varfile='VAR2'
IDL> .r r (or .r rall)
\end{verbatim}

See Sect. 5.6.1 for details on permanent snapshots.

With `\texttt{.r.pro}`, you can switch the part of the domain by changing the variable `\texttt{datadir}`:

\begin{verbatim}
IDL> datadir='data/proc3'
IDL> .r r
\end{verbatim}

will read the data written by processor 3.

**Reading data into IDL arrays or structures** As an alternative to the method described above, there is also the possibility to read the data into structures. This makes some more operations possible, e.g. reading data from an IDL program where the command `\texttt{.r}` is not allowed.

An efficient and still scriptable way would look like the following:

\begin{verbatim}
IDL> pc_read_var_raw, obj=var, tags=tags
IDL> bb = pc_get_quantity ('B', var, tags)
IDL> jj = pc_get_quantity ('j', var, tags)
\end{verbatim}

This reads the data into an array `\texttt{var}`, as well as the array indices of the contained physical variables into a separate structure `\texttt{tags}`. To use a caching mechanism within `\texttt{pc_get_quantity}`, please refer to the documentation and the examples contained in `\texttt{pencil-code/idl/pc_get_quantity.pro}`, where you can also start adding not yet implemented physical quantities.

To read a snapshot `\texttt{VAR10}` into the IDL structure \texttt{ff}, type the following command

\begin{verbatim}
IDL> pc_read_var, obj=ff, varfile='VAR10', /trimall
\end{verbatim}

The option `/trimall` removes ghost zones from the data. A number of other options are documented in the source code of \texttt{pc_read_var}. You can see what data the structure contains by using the command `\texttt{tag_names}`

\begin{verbatim}
IDL> print, tag_names(ff)
  T X Y Z DX DY DZ UU LNRHO AA
\end{verbatim}

\footnote{Keep in mind that \texttt{jj=curl1(bb)} would use iterated first derivatives instead of the second derivatives and thus be numerically less accurate than \texttt{jj=curl12(aa)}, particularly at small scales.}
One can access the individual variables by typing `ff.varname`, e.g.

```idl
IDL> help, ff.aa
<Expression>       FLOAT = Array[32, 32, 32, 3]
```

There are a number of files that read different data into structures. They are placed in the directory `~PENCIL_HOME/idl/files`. Here is a list of files (including suggested options to call them with):

- `pc_read_var_raw`, `obj=var`, `tags=tags`
  Efficiently read a snapshot into an array.
- `pc_read_var`, `obj=ff`, `/trimall`
  Read a snapshot into a structure.
- `pc_read_ts`, `obj=ts`
  Read the time series into a structure.
- `pc_read_xyaver`, `obj=xya`
  `pc_read_xzaver`, `obj=xza`
  `pc_read_yzaver`, `obj=yza`
  Read 1-D time series into a structure.
- `pc_read_const`, `obj=cst`
  Read code constants into a structure.
- `pc_read_pvar`, `obj=fp`
  Read particle data into a structure.
- `pc_read_param`, `obj=par`
  Read startup parameters into a structure.
- `pc_read_param`, `obj=par2`, `/param2`
  Read runtime parameters into a structure.

Other options are documented in the source code of the files.

For some examples on how to use these routines, see Sect. 5.19.3.

### 5.19.5 Python

Pencil supports reading, processing and the visualization of data using `python`. A number of scripts are placed in the subfolder `~PENCIL_HOME/python`. A README file placed under that subfolder contains the information needed to read Pencil output data into `python`.

#### Installation

For modern operating systems, Python is generally installed together with the system. If not, it can be installed via your preferred package manager or downloaded from the website [https://www.python.org/](https://www.python.org/). For convenience, it is strongly recommend to also install IPython, which is a more convenient console for Python. You will also need the NumPy, matplotlib and Tk library.

Perhaps the easiest way to obtain all the required software mentioned above is to install either Continuum’s Anaconda or Enthought’s Canopy. These Python distributions also provide (or indeed are) integrated graphical development environments.
In order for Python to find the Pencil Code commands you will have to add to your 
.bashrc:

```
export PYTHONPATH=$PENCIL_HOME/python
```

**ipythonrc** If you use IPython, for convenience, you should modify your 
.ipython/ipythonrc (create it if it doesn't exist) and add:

```python
import_all pencil
```

Additional, add to your .ipython/profile_default/startup/init.py the following lines:

```python
import numpy as np
import pylab as plt
import pencil as pc
import matplotlib
from matplotlib import rc

plt.ion()
matplotlib.rcParams['savefig.directory'] = ''
```

**.pythonrc** In case you are on a cluster and don’t have access to IPython you can edit 
your .pythonrc:

```python
#!/usr/bin/python
import numpy as np
import pylab as plt
import pencil as pc
import atexit
#import readline
# enables search with CTR+r in the history
try:
  import readline
except ImportError:
  print "Module readline not available."
else:
  import rlcompleter
  readline.parse_and_bind("tab: complete")
  # enables command history
  historyPath = os.path.expanduser("~/pyhistory")
  def save_history(historyPath=historyPath):
    import readline
    readline.write_history_file(historyPath)
  if os.path.exists(historyPath):
    readline.read_history_file(historyPath)
  atexit.register(save_history)
  del os, atexit, readline, rlcompleter, save_history, historyPath

plt.ion()
```
create the file .pythonhistory and add to your .bashrc:

```bash
export PYTHONSTARTUP=~/.pythonrc
```

**Pencil Code Commands in General**  For a list of all Pencil Code commands start IPython and type `pc. <TAB>` (as with auto completion). To access the help of any command just type the command followed by a '?' (no spaces), e.g.:

```python
In [1]: pc.dot?
```

Type: function

String Form:<function dot at 0x7f9d96cb0cf8>

File: ~/pencil-code/python/pencil/math/vector_multiplication.py

Definition: pc.dot(a, b)

Docstring:

take dot product of two pencil-code vectors a & b with shape

a.shape = (3,mz,my,mx)

You can also use `help(pc.dot)` for a more complete documentation of the command.

There are various reading routines for the Pencil Code data. All of them return an object with the data. To store the data into a user defined variable type e.g.

```python
In [1]: ts = pc.read_ts()
```

Most commands take some arguments. For most of them there is a default value, e.g.

```python
In [1]: pc.read_ts(filename='time_series.dat', datadir='data', plot_data=True)
```

You can change the values by simply typing e.g.

```python
In [1]: pc.read_ts(plot_data = False)
```

**Reading and Plotting Time Series**  Reading the time series file is very easy. Simply type

```python
In [1]: ts = pc.read_ts()
```

and Python stores the data in the variable `ts`. The physical quantities are members of the object `ts` and can be accessed accordingly, e.g. `ts.t`, `ts.emag`. To check which other variables are stored simply do the tab auto completion `ts. <TAB>`.

Plot the data with the matplotlib commands:

```python
In [1]: plt.plot(ts.t, ts.emag)
```

The standard plots are not perfect and need a little polishing. See the online wiki for a few examples on how to make pretty plots [https://github.com/pencil-code/pencil-code/wiki/PythonForPencil](https://github.com/pencil-code/pencil-code/wiki/PythonForPencil). You can save the plot into a file using the GUI or with

```python
In [1]: plt.savefig('plot.eps')
```

**Reading and Plotting VAR files and slice files**  Read var files:

```python
In [1]: var = pc.read_var()
```
5.20 Running on multi-processor computers

Read slice files:

In [1]: slices, t = pc.read_slices(field='bb1', extension='xy')

This returns the array slices with indices slices[nTimes, my, mx] and the time array t.

If you want to plot e.g. the x-component of the magnetic field at the central plane simply type:

In [1]: plt.imshow(zip(*var.bb[0, 128, :, :]), origin='lower', extent=[-4, 4, -4, 4])

For a complete list of arguments of plt.imshow refer to its documentation.

For a more interactive plot use:

In [1]: pc.animate_interactive(slices, t)

**Be aware:** arrays from the reading routines are ordered f[nvar, mz, my, mx], i.e. reversed to IDL. This affects reading var files and slice files.

5.20 Running on multi-processor computers

The code is parallelized using MPI (*message passing interface*) for a simple domain decomposition (data-parallelism), which is a straight-forward and very efficient way of parallelizing finite-difference codes. The current version has a few restrictions, which need to be kept in mind when using the MPI features.

The global number of grid points (but excluding the ghost zones) in a given direction must be an exact multiple of the number of processors you use in that direction. For example if you have nprocy=8 processors for the y direction, you can run a job with nygrid=64 points in that direction, but if you try to run a problem with nygrid=65 or nygrid=94, the code will complain about an inconsistency and stop. (So far, this has not been a serious restriction for us.)

5.20.1 How to run a sample problem in parallel

To run the sample problem in the directory ‘samples/conv-slab’ on 16 CPUs, you need to do the following (in that directory):

1. Edit ‘src/Makefile.local’ and replace
   
   MPICOMM = nompicomm

   by

   MPICOMM = mpicomm

2. Edit ‘src/cparam.local’ and replace

   integer, parameter :: ncpus=1, nprocy=1, nprocz=ncpus/nprocy, nprocx=1
   integer, parameter :: nxgrid=32, nygrid=nxgrid, nzgrid=nxgrid

   by

   integer, parameter :: ncpus=16, nprocy=4, nprocz=ncpus/nprocy, nprocx=1
   integer, parameter :: nxgrid=128, nygrid=nxgrid, nzgrid=nxgrid
The first line specifies a $4 \times 4$ layout of the data in the $y$ and $z$ direction. The second line increases the resolution of the run because running a problem as small as $32^3$ on 16 CPUs would be wasteful. Even $128^3$ may still be quite small in that respect. For performance timings, one should try and keep the size of the problem per CPU the same, so for example $256^3$ on 16 CPUs should be compared with $128^3$ on 2 CPUs.

3. Recompile the code

```
unix> (cd src; make)
```

4. Run it

```
unix> start.csh
unix> run.csh
```

Make sure that all CPUs see the same ‘data/’ directory; otherwise things will go wrong. Remember that in order to visualize the full domain with IDL (rather than just the domain processed and written by one processor), you need to use ‘rall.pro’ instead of ‘r.pro’.

5.20.2 Hierarchical networks (e.g. on Beowulf clusters)

On big Beowulf clusters, a group of nodes is often connected with a switch of modest speed, and all these groups are connected via a $n$ times faster uplink switch. When bandwidth-limited, it is important to make sure that consecutive processors are mapped onto the mesh such that the load on the uplink is $\lesssim n$ times larger than the load on the slower switch within each group. On a 512 node cluster, where groups of 24 processors are linked via fast ethernet switches, which in turn are connected via a Gigabit uplink ($\sim 10$ times faster), we found that $nprocy=4$ is optimal. For 128 processors, for example we find that $nprocy \times nprocz = 4 \times 32$ is the optimal layout, while. For comparison, $8 \times 16$ is 3 times slower, and $16 \times 8$ is 17 (!) times slower. These results can be understood from the structure of the network, but the basic message is to watch out for such effects and to try varying $nprocy$ and $nprocz$.

In cases where $nygrid > nzgrid$ it may be advantageous to swap the ordering of processor numbers. This can be done by setting $lprocz_{\text{slowest}}=F$.

5.20.3 Extra workload caused by the ghost zones

Normally, the workload caused by the ghost zones is negligible. However, if one increases the number of processors, a significant fraction of work is done in the ghost zones. In other words, the effective mesh size becomes much larger than the actual mesh size.

Consider a mesh of size $N_w = N_x \times N_y \times N_z$, and distribute the task over $P_w = P_x \times P_y \times P_z$ processors. If no communication were required, the number of points per processor would be

$$\frac{N_w}{P_w} = \frac{N_x \times N_y \times N_z}{P_x \times P_y \times P_z}.$$  \hspace{1cm} (26)

However, for finite difference codes some communication is required, and the amount of communication depends on spatial order of the scheme, $Q$. The PENCIL CODE works by
default with sixth order finite differences, so \( Q = 6 \), i.e. one needs 6 ghost zones, 3 on each end of the mesh. With \( Q \neq 0 \) the number of points per processor is

\[
\frac{N_{w}^{(\text{eff})}}{P_{w}} = \left( \frac{N_{x}}{P_{x}} + Q \right) \times \left( \frac{N_{y}}{P_{y}} + Q \right) \times \left( \frac{N_{z}}{P_{z}} + Q \right).
\]

(27)

There is efficient scaling only when

\[
\min \left( \frac{N_{x}}{P_{x}}, \frac{N_{y}}{P_{y}}, \frac{N_{z}}{P_{z}} \right) \gg Q.
\]

(28)

In the special case were \( N_{x} = N_{y} = N_{z} \equiv N = N_{w}^{1/3} \), with \( P_{x} = 1 \) and \( P_{y} = P_{z} \equiv P = P_{w}^{1/2} \), we have

\[
\frac{N_{w}^{(\text{eff})}}{P_{w}} = (N + Q) \times \left( \frac{N}{P} + Q \right)^{2}.
\]

(29)

For \( N = 128 \) and \( Q = 6 \) the effective mesh size exceeds the actual mesh size by a factor

\[
\frac{N_{w}^{(\text{eff})}}{N_{w}} = (N + Q) \times \left( \frac{N}{P} + Q \right)^{2} \times \frac{P_{w}}{N_{w}}.
\]

(30)

These factors are listed in Table 3.

<table>
<thead>
<tr>
<th>( P ) ( \backslash ) ( N )</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.15</td>
<td>1.07</td>
<td>1.04</td>
<td>1.02</td>
<td>1.01</td>
</tr>
<tr>
<td>2</td>
<td>1.19</td>
<td>1.09</td>
<td>1.05</td>
<td>1.02</td>
<td>1.01</td>
</tr>
<tr>
<td>4</td>
<td>1.25</td>
<td>1.12</td>
<td>1.06</td>
<td>1.03</td>
<td>1.01</td>
</tr>
<tr>
<td>8</td>
<td>1.34</td>
<td>1.16</td>
<td>1.08</td>
<td>1.04</td>
<td>1.02</td>
</tr>
<tr>
<td>16</td>
<td>1.48</td>
<td>1.22</td>
<td>1.11</td>
<td>1.05</td>
<td>1.03</td>
</tr>
<tr>
<td>32</td>
<td>1.68</td>
<td>1.31</td>
<td>1.15</td>
<td>1.07</td>
<td>1.04</td>
</tr>
<tr>
<td>64</td>
<td>1.98</td>
<td>1.44</td>
<td>1.21</td>
<td>1.10</td>
<td>1.05</td>
</tr>
<tr>
<td>128</td>
<td>2.45</td>
<td>1.64</td>
<td>1.30</td>
<td>1.14</td>
<td>1.07</td>
</tr>
<tr>
<td>256</td>
<td>3.21</td>
<td>1.93</td>
<td>1.43</td>
<td>1.20</td>
<td>1.10</td>
</tr>
<tr>
<td>512</td>
<td>4.45</td>
<td>2.40</td>
<td>1.62</td>
<td>1.29</td>
<td>1.14</td>
</tr>
</tbody>
</table>

Ideally, one wants to keep the work in the ghost zones at a minimum. If one accepts that 20–25% of work are done in the ghost zones, one should use 4 processors for \( 128^{3} \) mesh points, 16 processors for \( 256^{3} \) mesh points, 64 processors for \( 512^{3} \) mesh points, 256 processors for \( 1024^{3} \) mesh points, and 512 processors for \( 1536^{3} \) mesh points.

### 5.21 Running in double-precision

With many compilers, you can easily switch to double precision (8-byte floating point numbers) as follows.

Add the lines

```plaintext
# Use double precision
REAL_PRECISION = double
```
to ‘src/Makefile.local’ and (re-)run pc_setupsrc.

If REAL PRECISION is set to ‘double’, the flag FFLAGS DOUBLE is appended to the Fortran compile flags. The default for FFLAGS DOUBLE is -r8, which works for g95 or ifort; for gfortran, you need to make sure that FFLAGS DOUBLE is set to -fdefault-real-8.

You can see the flags in ‘src/Makefile.inc’, which is the first place to check if you have problems compiling for double precision.

Using double precision might be important in turbulence runs where the resolution is 256^3 meshpoints and above (although such runs often seem to work fine at single precision). To continue working in double precision, you just say lread_from_other_prec=T in run_pars; see Section [5.1]

5.22 Power spectrum

Given a real variable $u$, its Fourier transform $\tilde{u}$ is given by

$$\tilde{u}(k_x, k_y, k_z) = \mathcal{F}(u(x, y, z)) = \frac{1}{N_xN_yN_z} \sum_{p=0}^{N_x-1} \sum_{q=0}^{N_y-1} \sum_{r=0}^{N_z-1} u(x_p, y_q, z_r) \times \exp(-ik_xx_p) \exp(-ik_yy_q) \exp(-ik_zz_r),$$

(31)

where

$$|k_x| < \frac{\pi N_x}{L_x}, \quad |k_y| < \frac{\pi N_y}{L_y}, \quad |k_z| < \frac{\pi N_z}{L_z},$$

when $L$ is the size of the simulation box. The three-dimensional power spectrum $P(k)$ is defined as

$$P(k) = \frac{1}{2} \tilde{u}\tilde{u}^*,$$

(32)

where

$$k = \sqrt{k_x^2 + k_y^2 + k_z^2}.$$  

(33)

Note that we can only reasonably calculate $P(k)$ for $k < \pi N_x/L_x$.

To get power spectra from the code, edit ‘run.in’ and add for example the following lines

```
    dspec=5., ou_spec=T, ab_spec=T ! (for energy spectra)
    oned=T
```

under run_pars. The kinetic (vel_spec) and magnetic (mag_spec) power spectra will now be calculated every 5.0 (dspec) time units. The Fourier spectra is calculated using fftpack. In addition to calculating the three-dimensional power spectra also the one-dimensional power spectra will be calculated (oned).

In addition one must edit ‘src/Makefile.local’ and add the lines

```
    FOURIER = fourier_fftpack
    POWER = power_spectrum
```

Running the code will now create the files ‘powerhel_mag.dat’ and ‘power_kin.dat’ containing the three-dimensional magnetic and kinetic power spectra respectively. In addition to these three-dimensional files we will also find the one-dimensional files ‘powerbx_x.dat’, ‘powerby_x.dat’, ‘powerbz_x.dat’, ‘powerux_x.dat’, ‘poweruy_x.dat’ and
'poweruz_x.dat'. In these files the data are stored such that the first line contains the time of the snapshot, the following \( nxgrid/2 \) numbers represent the power at each wavenumber, from the smallest to the largest. If several snapshots have been saved, they are being stored immediately following the preceding snapshot.

You can read the results with the idl procure 'power', like this:

```idl
power,'_kin','_mag',k=k,spec1=spec1,spec2=spec2,i=n,tt=t,/noplot
power,'hel_kin','hel_mag',k=k,spec1=spec1h,spec2=spec2h,i=n,tt=t,/noplot
```

If powerhel is invoked, krms is written during the first computation. The relevant output file is 'power_krms.dat'. This is needed for a correct calculation of \( k \) used in the realizability conditions.

A caveat of the implementation of Fourier transforms in the PENCIL CODE is that, due to the parallelization, the permitted resolution is limited to the case when one direction is an integer multiple of the other. So, it can be done for

\[ Nx = n*Ny \]

Unfortunately, for some applications one wants \( Nx < Ny \). Wlad experimented with arbitrary resolution by interpolating \( x \) to the same resolution of \( y \) prior to transposing, then transform the interpolated array and then interpolating it back (check 'fourier_transform_y' in 'fourier_fftpack.f90').

A feature of our current implementation with \( x \) parallelization is that fft_xyz_parallel_3D requires nygrid to be an integer multiple of nprocy*nprocz. Examples of good mesh layouts are listed in Table 4.

**Table 4:** Examples of mesh layouts for which Fourier transforms with \( x \) parallelization is possible.

<table>
<thead>
<tr>
<th>ny</th>
<th>nprocy</th>
<th>nprocy</th>
<th>nprocz</th>
<th>ncpus</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>1</td>
<td>16</td>
<td>16</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
<td>16</td>
<td>16</td>
<td>512</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>16</td>
<td>16</td>
<td>1024</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>2048</td>
</tr>
<tr>
<td>288</td>
<td>2</td>
<td>16</td>
<td>18</td>
<td>576</td>
</tr>
<tr>
<td>512</td>
<td>2</td>
<td>16</td>
<td>32</td>
<td>1024</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>16</td>
<td>16</td>
<td>1024</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>16</td>
<td>32</td>
<td>2048</td>
</tr>
<tr>
<td>576</td>
<td>4</td>
<td>18</td>
<td>32</td>
<td>2304</td>
</tr>
<tr>
<td>576</td>
<td>8</td>
<td>18</td>
<td>32</td>
<td>4608</td>
</tr>
<tr>
<td>576</td>
<td>16</td>
<td>18</td>
<td>32</td>
<td>9216</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>32</td>
<td>32</td>
<td>4096</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>16</td>
<td>64</td>
<td>4096</td>
</tr>
<tr>
<td>1024</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>4096</td>
</tr>
<tr>
<td>1152</td>
<td>4</td>
<td>36</td>
<td>32</td>
<td>4608</td>
</tr>
<tr>
<td>1152</td>
<td>4</td>
<td>32</td>
<td>36</td>
<td>4608</td>
</tr>
<tr>
<td>2304</td>
<td>2</td>
<td>32</td>
<td>72</td>
<td>4608</td>
</tr>
<tr>
<td>2304</td>
<td>4</td>
<td>36</td>
<td>64</td>
<td>9216</td>
</tr>
<tr>
<td>2304</td>
<td>4</td>
<td>32</td>
<td>72</td>
<td>9216</td>
</tr>
</tbody>
</table>

To visualize with IDL just type power and you get the last snapshot of the three-
dimensional power spectrum. See head of ‘$PENCIL_HOME/idl/power.pro’ for options to power.

5.23 Structure functions

We define the p-th order longitudinal structure function of $u$ as

$$S_{\text{long}}^p(l) = \langle |u_x(x+l, y, z) - u_x(x, y, z)|^p \rangle,$$  \hspace{1cm} (34)

while the transverse is

$$S_{\text{trans}}^p(l) = \langle |u_y(x+l, y, z) - u_y(x, y, z)|^p \rangle + \langle |u_z(x+l, y, z) - u_z(x, y, z)|^p \rangle.$$  \hspace{1cm} (35)

Edit ‘run.in’ and add for example the following lines

dspec=2.3,
lsfu=T, lsfb=T, lsfz1=T, lsfz2=T

under run_pars. The velocity (lsfu), magnetic (lsfb) and Elsasser (lsfz1 and lsfz2) structure functions will now be calculated every 2.3 (dspec) time unit.

In addition one must edit ‘src/Makefile.local’ and add the line

```
STRUCT_FUNC = struct_func
```

In ‘src/cparam.local’, define $lb_{\text{nxgrid}}$ and make sure that

```
xgrid = nygrid = nzgrid = 2**lb_{\text{nxgrid}}
```

E.g.

```'
integer, parameter :: lb_{\text{nxgrid}}=5
integer, parameter :: nxgrid=2**lb_{\text{nxgrid}},nygrid=nxgrid,nzgrid=nxgrid
```

Running the code will now create the files:

‘sfu-1.dat’, ‘sfu-2.dat’, ‘sfu-3.dat’ (velocity),
‘sfb-1.dat’, ‘sfb-2.dat’, ‘sfb-3.dat’ (magnetic field),
‘sfz1-1.dat’, ‘sfz1-2.dat’, ‘sfz1-3.dat’ (first Elsasser variable),
‘sfz2-1.dat’, ‘sfz2-2.dat’, ‘sfz2-3.dat’ (second Elsasser variable),

which contains the data of interest. The first line in each file contains the time $t$ and the number $q_{\text{max}}$, such that the largest moment calculated is $q_{\text{max}} - 1$. The next $i_{\text{max}}$ numbers represent the first moment structure function for the first snapshot, here

$$i_{\text{max}} = 2 \frac{\ln(n_{\text{grid}})}{\ln 2} - 2.$$  \hspace{1cm} (36)

The next $i_{\text{max}}$ numbers contain the second moment structure function, and so on until $q_{\text{max}} - 1$. The following $i_{\text{max}}$ numbers then contain the data of the signed third order structure function i.e. $S_{\text{long}}^3(l) = \langle |u_x(x+l, y, z) - u_x(x, y, z)|^3 \rangle$.

The following $i_{\text{max}} \times q_{\text{max}} \times 2$ numbers are zero if $n_{\text{r-directions}} = 1$ (default), otherwise they are the same data as above but for the structure functions calculated in the y and z directions.
If the code has been run long enough as to calculate several snapshots, these snapshots will now follow, being stored in the same way as the first snapshot.

To visualize with IDL just type \texttt{structure} and you get the time-average of the first order longitudinal structure function (be sure that \texttt{pencil-runs/forced/idl/} is in IDL\_PATH). See head of \texttt{pencil-runs/forced/idl/structure.pro} for options to structure.

### 5.24 Particles

The PENCIL CODE has modules for tracer particles and for dust particles (see Sect.\[6.14]\). The particle modules are chosen by setting the value of the variable \texttt{PARTICLES} in \texttt{Makefile.local} to either \texttt{particles_dust} or \texttt{particles_tracers}. For the former case each particle has six degrees of freedom, three positions and three velocities. For the latter it suffices to have only three position variables as the velocity of the particles are equal to the instantaneous fluid velocity at that point. In addition one can choose to have several additional internal degrees of freedoms for the particles. For example one can temporally evolve the particles radius by setting \texttt{PARTICLES\_RADIUS} to \texttt{particles\_radius} in \texttt{Makefile.local}.

All particle infrastructure is controlled and organized by the \texttt{Particles\_main} module. This module is automatically selected by \texttt{Makefile.src} if \texttt{PARTICLES} is different from \texttt{noparticles}. Particle modules are compiled as a separate library. This way the main part of the Pencil Code only needs to know about the \texttt{particles\_main.a} library, but not of the individual particle modules.

For a simulation with particles one must in addition define a few parameters in \texttt{cparam.local}. Here is a sample of \texttt{cpparam.local} for a parallel run with 2,000,000 particles:

\begin{verbatim}
integer, parameter :: ncpus=16, nprocy=4, nprocz=4, nprocx=1
integer, parameter :: nxgrid=128, nygrid=256, nzgrid=128
integer, parameter :: npar=2000000, mpar\_loc=400000, npar\_mig=1000
integer, parameter :: npar\_species=2
\end{verbatim}

The parameter \texttt{npar} is the number of particles in the simulation, \texttt{mpar\_loc} is the number of particles that is allowed on each processor and \texttt{npar\_mig} is the number of particles that are allowed to migrate from one processor to another in any time-step. For a non-parallel run it is enough to specify \texttt{npar}. The number of particle species is set through \texttt{npar\_species} (assumed to be one if not set). The particle input parameters are given in \texttt{start.in} and \texttt{run.in}. Here is a sample of the particle part of \texttt{start.in} for dust particles:

\begin{verbatim}
/ &particles\_init\_pars
   initxp='gaussian-z', initvvp='random'
   zp0=0.02, delta_vp0=0.01, eps\_dtog=0.01, tausp=0.1
   lparticle\_mesh\_tsc=T
/
\end{verbatim}

The initial positions and velocities of the dust particles are set in \texttt{initxxp} and \texttt{initvvp}. The next four input parameters are further specifications of the initial condition. Interaction between the particles and the mesh, e.g. through drag force or self-gravity, require a mapping of the particles on the mesh. The PENCIL CODE currently supports NGP (Nearest Grid Point, default), CIC (Cloud in Cell, set \texttt{lparticle\_mesh\_cic=T}) and TSC

Here is a sample of the particle part of run.in (also for dust particles):

```
/ &particles_run_pars
  ldragforce_gas_par=T
  cdtp=0.2
/
```

The logical ldragforce_gas_par determines whether the dust particles influence the gas with a drag force. cdtp tells the code how many friction times should be resolved in a minimum time-step.

The sample run ‘samples/sedimentation/’ contains the latest setup for dust particles.

5.24.1 Particles in parallel

The particle variables (e.g. $x_i$ and $v_i$) are kept in the arrays fp and dfp. For parallel runs, particles must be able to move from processor to processor as they pass out of the $(x,y,z)$-interval of the local processor. Since not all particles are present at the same processor at the same time (hopefully), there is some memory optimization in making fp not big enough to contain all the particles at once. This is achieved by setting the code variable mpar_loc less than npar in cparam.local for parallel runs. When running with millions of particles, this trick is necessary to keep the memory need of the code down.

The communication of migrating particles between the processors happens as follows (see the subroutine redist_particles_procs in particles_sub.f90):

1. In the beginning of each time-step all processors check if any of their particles have crossed the local $(x,y,z)$-interval. These particles are called migrating particles. A run can have a maximum of npar_mig migrating particles in each time-step. The value of npar_mig must be set in cparam.local. The number should (of course) be slightly larger than the maximum number of migrating particles at any time-step during the run. The diagnostic variable nmigmax can be used to output the maximum number of migrating particles at a given time-step. One can set lmigration_redo=T in &particles_run_pars to force the code to redo the migration step if more than npar_mig particles want to migrate. This does slow the code down somewhat, but has the benefit that the code does not stop when more than npar_mig particles want to migrate.

2. The index number of the receiving processor is then calculated. This requires some assumption about the grid on other processors and will currently not work for nonequidistant grids. Particles do not always pass to neighboring processors as the global boundary conditions may send them to the other side of the global domains (periodic or shear periodic boundary conditions).

3. The migrating particle information is copied to the end of fp, and the empty spot left behind is filled up with the particle of the highest index number currently present at the processor.

4. Once the number of migrating particles is known, this information is shared with neighboring processors (including neighbors over periodic boundaries) so that they
all know how many particles they have to receive and from which processors.

5. The communication happens as directed MPI communication. That means that processors 0 and 1 can share migrating particles at the same time as processors 2 and 3 do it. The communication happens from a chunk at the end of fp (migrating particles) to a chunk that is present just after the particle of the highest index number that is currently at the receiving processor. Thus the particles are put directly at their final destination, and the migrating particle information at the source processor is simply overwritten by other migrating particles at the next time-step.

6. Each processor keeps track of the number of particles that it is responsible for. This number is stored in the variable `npar_loc`. It must never be larger than `npar` (see above). When a particle leaves a processor, `npar_loc` is reduced by one, and then increased by one at the processor that receives that particle. The maximum number of particles at any processor is stored in the diagnostic variable `nparmax`. If this value is not close to `npar/ncpus`, the particles have piled up in such a way that computations are not evenly shared between the processors. One can then try to change the parallelization architecture (`nprocy` and `nprocz`) to avoid this problem.

In simulations with many particles (comparable to or more than the number of grid cells), it is crucial that particles are shared relatively evenly among the processors. One can as a first approach attempt to not parallelize directions with strong particle density variations. However, this is often not enough, especially if particles clump locally.

Alternatively one can use Particle Block Domain Decomposition (PBDD, see Johansen et al. 2011). The steps in Particle Block Domain Decomposition scheme are as follows:

1. The fixed mesh points are domain-decomposed in the usual way (with `ncpus=nprocx×nprocy×nprocz`).

2. Particles on each processor are counted in bricks of size `nbx×nby×nbz` (typically `nbx=nby=nbz=4`).

3. Bricks are distributed among the processors so that each processor has approximately the same number of particles

4. Adopted bricks are referred to as blocks.

5. The Pencil Code uses a third order Runge-Kutta time-stepping scheme. In the beginning of each sub-time-step particles are counted in blocks and the block counts communicated to the bricks on the parent processors. The particle density assigned to ghost cells is folded across the grid, and the final particle density (defined on the bricks) is communicated back to the adopted blocks. This step is necessary because the drag force time-step depends on the particle density, and each particle assigns density not just to the nearest grid point, but also to the neighboring grid points.

6. In the beginning of each sub-time-step the gas density and gas velocity field is communicated from the main grid to the adopted particle blocks.

7. Drag forces are added to particles and back to the gas grid points in the adopted blocks. This partition aims at load balancing the calculation of drag forces.

8. At the end of each sub-time-step the drag force contribution to the gas velocity field is communicated from the adopted blocks back to the main grid.
Particle Block Domain Decomposition is activated by setting PARTICLES = particles-dust_blocks and PARTICLES_MAP = particles_map_blocks in Makefile.local. A sample of cparam.local for Particle Block Domain Decomposition can be found in samples/sedimentation/blocks:

```fortran
integer, parameter :: ncpus=4, nprocx=2, nprocy=2, nprocz=1
integer, parameter :: nxgrid=32, nygrid=32, nzgrid=32
integer, parameter :: npar=10000, mpar_loc=5000, npar_mig=100
integer, parameter :: npar_species=4
integer, parameter :: nbrickx=4, nbricky=4, nbrickz=4, nblockmax=32
```

The last line defines the number of bricks in the total domain – here we divide the grid into $4 \times 4 \times 4$ bricks each of size $8 \times 8 \times 8$ grid points. The parameter nblockmax tells the code the maximum number of blocks any processor may adopt. This should not be so low that there is not room for all the bricks with particles, nor so high that the code runs out of memory.

### 5.24.2 Large number of particles

When dealing with large number of particles, one needs to make sure that the number of particles npar is less than the maximum integer that the compiler can handle with. The maximum integer can be checked by the Fortran intrinsic function huge,

```fortran
program huge_integers
  print *, huge(0_4) ! for default Fortran integer (32 Bit)
  print *, huge(0_8) ! for 64 Bit integer in Fortran
end program huge_integers
```

If the number of particles npar is larger than default maximum integer, one can promote the maximum integer to 64 Bit by setting

```fortran
integer(kind=8), parameter :: npar=4294967296
```

in the cparam.local file. This works because the data type of npar is only set here. It is worth noting that one should not use the flag

```fortran
FFLAGS += -integer-size 64
```

to promote all the integers to 64 Bit. This will break the Fortran-C interface. One should also make sure that npar_mig<=$npar/ncpus. It is also beneficial to set mpar_loc=2*npar/ncpus.

### 5.24.3 Random number generator

There are several methods to generate random number in the code. It is worth noting that when simulating coagulation with the super-particle approach, one should use the intrinsic random number generator of FORTRAN instead of the one implemented in the code. When invoking random_number_wrapper, there will be back-reaction to the gas flow. This unexpected back-reaction can be tracked by inspecting the power spectra, which exhibits the oscillation at the tail. To avoid this, one should set luser_random_number_wrapper=F under the module particles_coag_run_pars in run.in.
5.25 Non-cartesian coordinate systems

Since the spring of 2007 spherical and cylindrical polar coordinates have been implemented, although this development is not yet completed. Spherical coordinates are invoked by adding the following line in the file ‘start.in’

```
&init_pars
  coord_system='spherical_coords'
```

Another possibility is to put `cylindrical_coords` instead. In practice, the names \((x, y, z)\) are still used, but they refer then to \((r, \theta, \phi)\) or \((r, \phi, z)\) instead.

Bug reports, corrections, and improvements on these are appreciated.
6 The equations

The equations solved by the PENCIL CODE are basically the standard compressible MHD equations. However, the modular structure allows some variations of the MHD equations, as well as switching off some of the equations or individual terms of the equation (nomagnetic, noentropy, etc.).

In this section the equations are presented in their most complete form. It may be expected that the code can evolve most subsets or simplifications of these equations.

6.1 Continuity equation

In the code the continuity equation, \( \partial \rho / \partial t + \nabla \cdot \rho \mathbf{u} = 0 \), is written in terms of \( \ln \rho \),

\[
\frac{D \ln \rho}{Dt} = -\nabla \cdot \mathbf{u} .
\]  

(37)

Here \( \rho \) denotes density, \( \mathbf{u} \) the fluid velocity, \( t \) is time and \( D/Dt \equiv \partial/\partial t + \mathbf{u} \cdot \nabla \) is the convective derivative.

6.2 Equation of motion

In the equation of motion, using a perfect gas, the pressure term, can be expressed as

\[-\rho^{-1} \nabla p = -c_s^2 (\nabla s/c_p + \nabla \ln \rho) ,\]

where the squared sound speed is given by

\[c_s^2 = \frac{\gamma}{\rho} = c_{s0}^2 \exp \left[ \frac{\gamma s}{c_p} + (\gamma - 1) \frac{\ln \rho}{\rho_0} \right] ,\]  

(38)

and \( \gamma = c_p/c_v \) is the ratio of specific heats, or adiabatic index. Note that \( c_s^2 \) is proportional to the temperature with \( c_s^2 = (\gamma - 1)c_pT \).

The equation of motion is accordingly

\[
\frac{D \mathbf{u}}{Dt} = -c_s^2 \nabla \left( \frac{s}{c_p} + \ln \rho \right) - \nabla \Phi_{\text{grav}} + \frac{j \times \mathbf{B}}{\rho} \\
+ \nu \left( \nabla^2 \mathbf{u} + \frac{1}{3} \nabla \nabla \cdot \mathbf{u} + 2 \mathbf{S} \cdot \nabla \ln \rho \right) + \zeta (\mathbf{\nabla} \mathbf{\nabla} \cdot \mathbf{u}) ;
\]  

(39)

Here \( \Phi_{\text{grav}} \) is the gravity potential, \( j \) the electric current density, \( \mathbf{B} \) the magnetic flux density, \( \nu \) is kinematic viscosity, \( \zeta \) describes a bulk viscosity, and, in Cartesian coordinates

\[ S_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \nabla \cdot \mathbf{u} \right) \]  

(40)

is the rate-of-shear tensor that is traceless, because it can be written as the generic rate-of-strain tensor minus its trace. In curvilinear coordinates, we have to replace partial differentiation by covariant differentiation (indicated by semicolons), so we write \( S_{ij} = \frac{1}{2} (u_{ij} + u_{ji}) - \frac{1}{3} \delta_{ij} \nabla \cdot \mathbf{u} \).

The interpretation of the two viscosity terms varies greatly depending upon the Viscosity module used, and indeed on the parameters given to the module. See [\[6.6\].

For isothermal hydrodynamics, see [\[6.4\] below.
6.3 Induction equation

\[ \frac{\partial A}{\partial t} = u \times B - \eta \mu_0 j \, . \] (41)

Here \( A \) is the magnetic vector potential, \( B = \nabla \times A \) the magnetic flux density, \( \eta = 1/(\mu_0 \sigma) \) is the magnetic diffusivity (\( \sigma \) denoting the electrical conductivity), and \( \mu_0 \) the magnetic vacuum permeability. This form of the induction equation corresponds to the Weyl gauge \( \Phi = 0 \), where \( \Phi \) denotes the scalar potential.

6.4 Entropy equation

The current thermodynamics module entropyformulates the thermal part of the physics in terms of entropy \( s \), rather than thermal energy \( e \), which you may be more familiar with. Thus the two fundamental thermodynamical variables are \( \ln \rho \) and \( s \). The reason for this choice of variables is that entropy is the natural physical variable for (at least) convection processes: the sign of the entropy gradient determines convective (in)stability, the Rayleigh number is proportional to the entropy gradient of the associated hydrostatic reference solution, etc. The equation solved is

\[ \frac{\rho T}{D_t} = \mathcal{H} - C + \nabla \cdot (K \nabla T) + \eta \mu_0 j^2 + 2 \rho \nu S \otimes S + \zeta \rho (\nabla \cdot u)^2 \, . \] (42)

Here, \( T \) is temperature, \( c_p \) the specific heat at constant pressure, \( \mathcal{H} \) and \( C \) are explicit heating and cooling terms, \( K \) is the radiative (thermal) conductivity, \( \zeta \) describes a bulk viscosity, and \( S \) is the rate-of-shear tensor that is traceless.

In the entropy module we solve for the specific entropy, \( s \). The heat conduction term on the right hand side can be written in the form

\[ \frac{\nabla \cdot (K \nabla T)}{\rho T} \]

\[ = c_p \chi \left[ \nabla^2 \ln T + \nabla \ln T \cdot \nabla (\ln T + \ln \chi + \ln \rho) \right] \] (44)

\[ = c_p \chi \left[ \nabla^2 s/c_p + (\gamma - 1) \nabla^2 \ln \rho \right] + c_p \chi \left[ \nabla (\nabla s/c_p + \nabla \ln \rho) \cdot [\nabla (\nabla s/c_p + \nabla \ln \rho) + \nabla \ln \chi] \right] , \] (45)

where \( \chi = K/(\rho c_p) \) is the thermal diffusivity. The latter equation shows that the diffusivity for \( s \) is \( \gamma \chi \), which is what we have used in Eq. (24).

In an alternative formulation for a constant \( K \), the heat conduction term on the right hand side can also be written in the form

\[ \frac{\nabla \cdot (K \nabla T)}{\rho T} = \frac{K}{\rho} \left[ \nabla^2 \ln T + (\nabla \ln T)^2 \right] \] (46)

which is the form used when constant \( K \) is chosen.

Note that by setting \( \gamma = 1 \) and initially \( s = 0 \), one obtains an isothermal equation of state (albeit at some unnecessary expense of memory). Similarly, by switching off the evolution terms of entropy, one immediately gets polytropic behavior (if \( s \) was initially constant) or generalized polytropic behavior (where \( s \) is not uniform, but \( \partial s/\partial t = 0 \)).

A better way to achieve isothermality is to use the noentropy module.
6.4.1 Viscous heating

We can write the viscosity as the divergence of a tensor $\tau_{ij,j}$,

$$\rho \frac{\partial u_i}{\partial t} = ... + \tau_{ij,j}, \quad (47)$$

where $\tau_{ij} = 2\nu \rho S_{ij}$ is the stress tensor. The viscous power density $P$ is

$$P = u_i \tau_{ij} \quad (48)$$

$$= \frac{\partial}{\partial x_j} (u_i \tau_{j}) - u_{i,j} \tau_{ij} \quad (49)$$

The term under the divergence is the viscous energy flux and the other term is the kinetic energy loss due to heating. The heating term $+u_{i,j} \tau_{ij}$ is positive definite, because $\tau_{ij}$ is a symmetric tensor and the term only gives a contribution from the symmetric part of $u_{i,j}$, which is $\frac{1}{2}(u_{i,j} + u_{j,i})$, so

$$u_{i,j} \tau_{ij} = \frac{1}{2} \nu \rho (u_{i,j} + u_{j,i}) (2S_{ij}). \quad (50)$$

But, because $S_{ij}$ is traceless, we can add anything proportional to $\delta_{ij}$ and, in particular, we can write

$$u_{i,j} \tau_{ij} = \frac{1}{2} (u_{i,j} + u_{j,i}) (2\nu \rho S_{ij}) \quad (51)$$

$$= \frac{1}{2} (u_{i,j} + u_{j,i} - \frac{1}{3} \delta_{ij} \nabla \cdot u) (2\nu \rho S_{ij}) \quad (52)$$

$$= 2\nu \rho S^2, \quad (53)$$

which is positive definite.

6.4.2 Alternative description

By setting pretend\_lnTT=T in init\_pars or run\_pars (i.e. the general part of the name list) the logarithmic temperature is used instead of the entropy. This has computational advantages when heat conduction (proportional to $K \nabla T$) is important. Another alternative is to use another module, i.e. set ENTROPY=temperature\_ideal\_gas in 'Makefile.local'.

When pretend\_lnTT=T is set, the entropy equation

$$\frac{\partial s}{\partial t} = -u \cdot \nabla s + \frac{1}{\rho T} \text{RHS} \quad (54)$$

is replaced by

$$\frac{\partial \ln T}{\partial t} = -u \cdot \nabla \ln T + \frac{1}{\rho c_v T} \text{RHS} - (\gamma - 1) \nabla \cdot u, \quad (55)$$

where RHS is the right hand side of equation (42).
6.5 Transport equation for a passive scalar

In conservative form, the equation for a passive scalar is

\[
\frac{\partial}{\partial t}(\rho c) + \nabla \cdot [\rho c \mathbf{u} - \rho D \nabla c] = 0.
\] (56)

Here \( c \) denotes the concentration (per unit mass) of the passive scalar and \( D \) its diffusion constant (assumed constant). In the code this equation is solved in terms of \( \ln c \),

\[
\frac{D \ln c}{Dt} = D \left[ \nabla^2 \ln c + (\nabla \ln \rho + \nabla \ln c) \cdot \nabla \ln c \right].
\] (57)

Using \( \ln c \) instead of \( c \) has the advantage that it enforces \( c > 0 \) for all times. However, the disadvantage is that one cannot have \( c = 0 \). For this reason we ended up using the non-logarithmic version by invoking PSCALAR=pscalar_nolog.

6.6 Bulk viscosity

For a monatomic gas it can be shown that the bulk viscosity vanishes. We therefore don’t use it in most of our runs. However, for supersonic flows, or even otherwise, one might want to add a shock viscosity which, in its simplest formulation, take the form of a bulk viscosity.

6.6.1 Shock viscosity

Shock viscosity, as it is used here and also in the Stagger Code of Åke Nordlund, is proportional to positive flow convergence, maximum over five zones, and smoothed to second order,

\[
\zeta_{\text{shock}} = c_{\text{shock}} \left\langle \max_{5} \left( -\nabla \cdot \mathbf{u} \right) \right\rangle \left( \min(\delta x, \delta y, \delta z) \right)^2,
\] (58)

where \( c_{\text{shock}} \) is a constant defining the strength of the shock viscosity. In the code this dimensionless coefficient is called nu_shock, and it is usually chosen to be around unity.

Assume that the shock viscosity only enters as a bulk viscosity, so the whole stress tensor is then

\[
\tau_{ij} = 2 \rho \nu S_{ij} + \rho \zeta_{\text{shock}} \delta_{ij} \nabla \cdot \mathbf{u}.
\] (59)

Assume \( \nu = \text{const} \), but \( \zeta \neq \text{const} \), so

\[
\rho^{-1} F_{\text{visc}} = \nu \left( \nabla^2 \mathbf{u} + \frac{1}{3} \nabla \nabla \cdot \mathbf{u} + 2 \mathbf{S} \cdot \nabla \ln \rho \right) + \zeta_{\text{shock}} \left[ \nabla \nabla \cdot \mathbf{u} + (\nabla \ln \rho + \nabla \ln \zeta_{\text{shock}}) \nabla \cdot \mathbf{u} \right].
\] (60)

and

\[
\rho^{-1} \Gamma_{\text{visc}} = 2 \nu S^2 + \zeta_{\text{shock}} (\nabla \cdot \mathbf{u})^2.
\] (61)

In the special case with periodic boundary conditions, we have

\[
2 \langle S^2 \rangle = \langle \omega^2 \rangle + \frac{8}{3} \langle (\nabla \cdot \mathbf{u})^2 \rangle.
\]

6.7 Equation of state

In its present configuration only hydrogen ionization is explicitly included. Other constituents (currently He and H\(_2\)) can have fixed values. The pressure is proportional to the total number of particles, i.e.

\[
p = (n_{\text{HII}} + n_{\text{HII}} + n_{\text{H}_2} + n_e + n_{\text{He}} + \ldots) k_B T.
\] (62)
It is convenient to normalize to the total number of \( \text{H} \) both in atomic and in molecular hydrogen, \( n_{\text{Htot}} \equiv n_{\text{H}} + 2n_{\text{H}_2} \), where \( n_{\text{H}} + n_{\text{HII}} = n_{\text{H}} \), and define \( x_e \equiv n_e/n_{\text{Htot}} \), \( x_{\text{He}} \equiv n_{\text{He}}/n_{\text{Htot}} \), and \( x_{\text{H}_2} \equiv n_{\text{H}_2}/n_{\text{Htot}} \). Substituting \( n_{\text{H}} = n_{\text{Htot}} - 2n_{\text{H}_2} \), we have
\[
p = (1 - x_{\text{H}_2} + x_e + x_{\text{He}} + \ldots)n_{\text{Htot}}k_BT. \tag{63}
\]
This can be written in the more familiar form
\[
p = \frac{\mathcal{R}}{\mu} \rho T, \tag{64}
\]
where \( \mathcal{R} = k_B/m_u \) and \( m_u \) is the atomic mass unit (which is for all practical purposes the same as \( m_{\text{Htot}} \)) and
\[
\mu = \frac{n_{\text{H}} + 2n_{\text{H}_2} + n_e + 4n_{\text{He}}}{n_{\text{H}} + n_{\text{H}_2} + n_e + n_{\text{He}}} = \frac{1 + 4x_{\text{He}}}{1 - x_{\text{H}_2} + x_e + x_{\text{He}}} \tag{65}
\]
is the mean molecular weight (which is here dimensionless; see Kippenhahn & Weigert 1990, p. 102). The factor 4 is really to be substituted for 3.9715. Some of the familiar relations take still the usual form, in particular \( e = c_vT \) and \( h = c_pT \) with \( c_v = \frac{3}{2}\mathcal{R}/\mu \) and \( c_p = \frac{5}{2}\mathcal{R}/\mu \).

The number ratio, \( x_{\text{He}} \), is more commonly expressed as the mass ratio, \( Y = m_{\text{He}}n_{\text{He}}/(n_{\text{HII}}n_{\text{Htot}} + m_{\text{He}}n_{\text{He}}) \), or \( Y = 4x_{\text{He}}/(1 + 4x_{\text{He}}) \), or \( 4x_{\text{He}} = (1/Y - 1)^{-1} \). For example, \( Y = 0.27 \) corresponds to \( x_{\text{He}} = 0.092 \) and \( Y = 0.25 \) to \( x_{\text{He}} = 0.083 \). Note also that for 100% \( \text{H}_2 \) abundance, \( x_{\text{H}_2} = 1/2 \).

In the following, the ionization fraction is given as \( y = n_e/n_{\text{H}} \), which can be different from \( x_e \) if there is \( \text{H}_2 \). Substituting for \( n_{\text{H}} \) in terms of \( n_{\text{Htot}} \) yields \( y = x_e/(1 - 2x_{\text{H}_2}) \).

### 6.8 Ionization

This part of the code can be invoked by setting \( \text{EOS=eos\_ionization} \) (or \( \text{EOS=eos\_temperature\_ionization} \)) in the ’Makefile.local’ file. The equation of state described below works for variable ionization, and the entropy offset is different from that used in Eq. (38), which is now no longer valid. As a replacement, one can use \( \text{EOS=eos\_fixed\_ionization} \), where the degree of ionization can be given by hand. Here the normalization of the entropy is the same as for \( \text{EOS=eos\_ionization} \). This case is described in more detail below\(^\text{12}\).

We treat the gas as being composed of partially ionized hydrogen and neutral helium. These are four different particle species, each of which regarded as a perfect gas.

The ionization fraction \( y \), which gives the ratio of ionized hydrogen to the total amount of hydrogen \( n_{\text{H}} \), is obtained from the Saha equation which, in this case, may be written as
\[
\frac{y^2}{1 - y} = \frac{1}{n_{\text{H}}} \left( \frac{m_e k_B T}{2\pi h^2} \right)^{3/2} \exp \left( - \frac{\chi_{\text{H}}}{k_B T} \right). \tag{66}
\]

The temperature \( T \) cannot be obtained directly from the PENCIL CODE’s independent variables \( \ln \rho \) and \( s \), but is itself dependent on \( y \). Hence, the calculation of \( y \) essentially becomes a root finding problem.

\(^{12}\text{We omit here the contribution of } \text{H}_2.\)
The entropy of a perfect gas consisting of particles of type \( i \) is known from the Sackur-Tetrode equation

\[
S_i = k_B N_i \left( \ln \left( \frac{1}{n_{\text{tot}}} \left( \frac{m_i k_B T}{2 \pi \hbar^2} \right)^{3/2} \right) + \frac{5}{2} \right).
\] (67)

Here \( N_i \) is the number of particles of a single species and \( n_{\text{tot}} \) is the total number density of all particle species.

In addition to the individual entropies we also have to take the entropy of mixing,

\[
S_{\text{mix}} = -N_{\text{tot}} k_B \sum_i p_i \ln p_i,
\]

into account. Summing up everything, we can get a closed expression for the specific entropy \( s \) in terms of \( y, \ln \rho \) and \( T \), which may be solved for \( T \).

![Figure 5: Dependence of temperature on entropy for different values of the density.](image)

For given \( \ln \rho \) and \( s \) we are then able to calculate the ionization fraction \( y \) by finding the root of

\[
f(y) = \ln \left[ \frac{1 - y}{y^2} \frac{1}{n_H} \left( \frac{m_e k_B T(y)}{2 \pi \hbar^2} \right)^{3/2} \right] - \frac{\chi_H}{k_B T(y)}.
\] (68)

In the ionized case, several thermodynamic quantities of the gas become dependent on the ionization fraction \( y \) such as its pressure, \( P = (1 + y + x_{\text{He}}) n_H k_B T \), and its internal energy, \( E = \frac{3}{2} (1 + y + x_{\text{He}}) n_H k_B T + y \chi_H \), where \( x_{\text{He}} \) gives the ratio of neutral helium to the total amount of hydrogen. The dependence of temperature on entropy is shown in Fig. 5 for different values of the density.

For further details regarding the procedure of solving for the entropy see Sect. 4.6 in the appendix.

6.8.1 Ambipolar diffusion

Another way of dealing with ionization in the PENCIL CODE is through use of the neutrals module. That module solves the coupled equations of neutral and ionized gas, in a two-fluid model.
where the subscripts $n$ and $i$ are for neutral and ionized, respectively. The terms $G$ and $F$, through which the two fluids exchange mass and momentum, are given by

\[
G = \zeta \rho_n - \alpha \rho_i^2 \quad (74)
\]

\[
F = \zeta \rho_n u_n - \alpha \rho_i^2 u_i + \gamma \rho_i \rho_n (u_n - u_i) \quad (75)
\]

In the above equations, $\zeta$ is the ionization coefficient, $\alpha$ is the recombination coefficient, and $\gamma$ the collisional drag strength. By the time of writing (spring 2009), these three quantities are supposed constant. The electron pressure $p_e$ is also assumed equal to the ion pressure. Only isothermal neutrals are supported so far.

In the code, Eq. (69) and Eq. (71) are solved in ‘density.f90’ and ‘hydro.f90’ respectively. Equation (70) is solved in ‘neutraldensity.f90’ and Eq. (72) in ‘neutralvelocity.f90’. The sample ‘1d-test/ambipolar-diffusion’ has the current setup for a two-fluid simulation with ions and neutrals.

### 6.9 Radiative transfer

Here we only state the basic equations. A full description about the implementation is given in Sect. [H.7](#) and in the original paper by Heinemann et al. (2006).

The basic equation for radiative transfer is

\[
\frac{dI}{d\tau} = -I + S \quad (76)
\]

where

\[
\tau = \int_0^s \chi(s') ds' \quad (77)
\]

is the optical depth ($s$ is the geometrical coordinate along the ray).

Note that radiative transfer is called also in ‘start.csh’, and again each time a snapshot is being written, provided the output of auxiliary variable is being requested lwrite_aux=T. (Also, of course, the pencil check runs radiative transfer 7 times, unless you put pencil_check_small=F.)
6.10 Self-gravity

The PENCIL CODE can consider the self-gravity of the fluid in the simulation box by adding the term
\[
\frac{\partial \mathbf{u}}{\partial t} = \ldots - \nabla \phi_{\text{self}}
\] (78)
to the equation of motion. The self-potential \( \phi_{\text{self}} \) (or just \( \phi \) for simplicity) satisfies Poisson’s equation
\[
\nabla^2 \phi = 4\pi G \rho.
\] (79)
The solution for a single Fourier component at scale \( k \) is
\[
\phi_k = \frac{-4\pi G \rho_k}{k^2}.
\] (80)

Here we have assumed periodic boundary conditions. The potential is obtained by Fourier-transforming the density, then finding the corresponding potential at that scale, and finally Fourier-transforming back to real space.

The \( x \)-direction in the shearing sheet is not strictly periodic, but is rather shear periodic with two connected points at the inner and outer boundary separated by the distance \( \Delta y(t) = \text{mod}[(3/2)\Omega_0 L_x t, L_y] \) in the \( y \)-direction. We follow here the method from [16] to allow for shear-periodic boundaries in the Fourier method for self-gravity. First we take the Fourier transform along the periodic \( y \)-direction. We then shift the entire \( y \)-direction by the amount \( \delta y(x) = \Delta y(t)x/L_x \) to make the \( x \)-direction periodic. Then we proceed with Fourier transforms along \( x \) and then \( z \). After solving the Poisson equation in Fourier space, we transform back to real space in the opposite order. We differ here from the method by [16] in that we shift in Fourier space rather than in real space.\(^{13}\)

The Fourier interpolation formula has the advantage over polynomial interpolation in that it is continuous and smooth in all its derivatives.

6.11 Incompressible and anelastic equations

This part has not yet been documented and is still under development.

6.12 Dust equations

The code treats gas and dust as two separate fluids.\(^{14}\) The dust and the gas interact through a drag force. This force can most generally be written as an additional term to the equation of motion as
\[
\frac{D \mathbf{u}_d}{Dt} = \ldots - \frac{1}{\tau_s} (\mathbf{u}_d - \mathbf{u}) .
\] (81)

Here \( \tau_s \) is the so-called stopping time of the considered dust species. This measures the coupling strength between dust and gas. In the Epstein drag regime
\[
\tau_s = \frac{a_d \rho_s}{c_s \rho} ,
\] (82)

\(^{13}\)We were kindly made aware of the possibility of interpolating in Fourier space by C. McNally on his website.

\(^{14}\)See master's thesis of A. Johansen (can be downloaded from http://www.mpia.de/homes/johansen/research_en.php)
where \( a_d \) is the radius of the dust grain and \( \rho_s \) is the solid density of the dust grain.

Two other important effects work on the dust. The first is coagulation controlled by the discrete coagulation equation

\[
\frac{dn_k}{dt} = \frac{1}{2} \sum_{i+j=k} A_{ij} n_i n_j - n_k \sum_{i=1}^{\infty} A_{ik} n_i.
\]  

(83)

In the code \( N \) discrete dust species are considered. Also the bins are logarithmically spaced in order to give better mass resolution. It is also possible to keep track of both number density and mass density of each bin, corresponding to having a variable grain mass in each bin.

Dust condensation is controlled by the equation

\[
\frac{dN}{dt} = \frac{1}{\tau_{\text{cond}}} N^{d-1}.
\]  

(84)

Here \( N \) is the number of monomers in the dust grain (such as water molecules) and \( d \) is the physical dimension of the dust grain. The condensation time \( \tau_{\text{cond}} \) is calculated from

\[
\frac{1}{\tau_{\text{cond}}} = A_1 v_{\text{th}} \alpha n_{\text{mon}} \left\{ 1 - \frac{1}{S_{\text{mon}}} \right\},
\]  

(85)

where \( A_1 \) is the surface area of a monomer, \( \alpha \) is the condensation efficiency, \( n_{\text{mon}} \) is the number density of monomers in the gas and \( S_{\text{mon}} \) is the saturation level of the monomer given by

\[
S_{\text{mon}} = \frac{P_{\text{mon}}}{P_{\text{sat}}}. \]

(86)

Here \( P_{\text{sat}} \) is the saturated vapor pressure of the monomer. Currently only water ice has been implemented in the code.

All dust species fulfill the continuity equation

\[
\frac{\partial \rho_d}{\partial t} + \nabla \cdot (\rho_d \mathbf{u}_d) = 0.
\]  

(87)

### 6.13 Cosmic ray pressure in diffusion approximation

Cosmic rays are treated in the diffusion approximation. The equation of state is \( p_c = (\gamma_c) e_c \) where the value of \( \gamma_c \) is usually somewhere between \( 14/9 \) and \( 4/3 \). In the momentum equation (89) the cosmic ray pressure force, \( -\rho^{-1} \nabla p_c \) is added on the right hand side, and \( e_c \) satisfies the evolution equation

\[
\frac{\partial e_c}{\partial t} + \nabla \cdot (e_c \mathbf{u}) + p_c \nabla \cdot \mathbf{u} = \partial_i (K_{ij} \partial_j e_c) + Q_c,
\]  

(88)

where \( Q_c \) is a source term and

\[
K_{ij} = K_\perp \delta_{ij} + (K_\parallel - K_\perp) \hat{B}_i \hat{B}_j
\]  

(89)

is an anisotropic diffusivity tensor.
In the non-conservative formulation of this code it is advantageous to expand the diffusion term using the product rule, i.e.

\[
\partial_i(K_{ij} \partial_j e_c) = -U_c \cdot \nabla e_c + K_{ij} \partial_i \partial_j e_c. 
\]

(90)

where \( U_{ci} = -\partial K_{ij} / \partial x_j \) acts like an extra velocity trying to straighten magnetic field lines. We can write this term also as \( U_c = -(K_\parallel - K_\perp) \nabla \cdot (\hat{B} \hat{B}) \), where the last term is a divergence of the dyadic product of unit vectors\(^\text{15}\). However, near magnetic nulls, this term can becomes infinite. In order to avoid this problem we are forced to limit \( \nabla \cdot (\hat{B} \hat{B}) \), and hence \(|U_c|\), to the maximum possible value that can be resolved at a given resolution.

A physically appealing way of limiting the maximum propagation speed is to restore an explicit time dependence in the equation for the cosmic ray flux, and to replace the diffusion term in Eq. (88) by a divergence of a flux that in turn obeys the equation

\[
\frac{\partial F_{ci}}{\partial t} = -\tilde{K}_{ij} \nabla_j e_c - \frac{F_{ci}}{\tau} \quad \text{(non-Fickian diffusion)},
\]

(91)

where \( K_{ij} = \tau \tilde{K}_{ij} \) would be the original diffusion tensor of Eq. (89), if the time derivative were negligible. Further details are described in Snodin et al. (2006).

6.14 Particles

Particles are entities that each have a space coordinate and a velocity vector, where a fluid only has a velocity vector field (the continuity equation of a fluid in some way corresponds to the space coordinate of particles). In the code particles are present either as tracer particles or as dust particles

6.14.1 Tracer particles

Tracer particles always have the local velocity of the gas. The dynamical equations are thus

\[
\frac{\partial x_i}{\partial t} = u,
\]

(92)

where the index \( i \) runs over all particles. Here \( u \) is the gas velocity at the position of the particle. One can choose between a first order (default) and a second order spline interpolation scheme (set \( \text{linear interpolation=T} \) in \&particles_init_pars) to calculate the gas velocity at the position of a tracer particle.

The sample run ‘samples/dust-vortex’ contains the latest setup for tracer particles.

6.14.2 Dust particles

Dust particles are allowed to have a velocity that is not similar to the gas,

\[
\frac{dx_i}{dt} = v_i.
\]

(93)

\(^{15}\)In practice, we calculate \( \partial_j(\hat{B}_i \hat{B}_j) = (\delta_{ij} - 2 \hat{B}_i \hat{B}_k) \hat{B}_j B_{k,j} / |B| \), where derivatives of \( B \) are calculated as \( B_{i,j} = \epsilon_{ilk} A_{l,j,k} \).
The particle velocity follows an equation of motion similar to a fluid, only there is no advection term. Dust particles also experience a drag force from the gas (proportional to the velocity difference between a particle and the gas).

\[ \frac{dv_i}{dt} = \ldots - \frac{1}{\tau_s} (v_i - u). \]  

(94)

Here \( \tau_s \) is the stopping time of the dust particle. The interpolation of the gas velocity to the position of a particle is done using one of three possible particle-mesh schemes,

- **NGP (Nearest Grid Point, default)**
  The gas velocity at the nearest grid point is used.

- **CIC (Cloud in Cell, set \texttt{lparticlemesh\_cic=T})**
  A first order interpolation is used to obtain the gas velocity field at the position of a particle. Affects 8 grid points.

- **TSC (Triangular Shaped Cloud, set \texttt{lparticlemesh\_tsc=T})**
  A second order spline interpolation is used to obtain the gas velocity field at the position of a particle. Affects 27 grid points.

The particle description is the proper description of dust grains, since they do not feel any pressure forces (too low number density). Thus there is no guarantee that the grains present within a given volume will be equilibrated with each other, although drag force may work for small grains to achieve that. Larger grains (meter-sized in protoplanetary discs) must be treated as individual particles.

To conserve momentum the dust particles must affect the gas with a friction force as well. The strength of this force depends on the dust-to-gas ratio \( \epsilon_d \), and it can be safely ignored when there is much more gas than there is dust, e.g. when \( \epsilon_d = 0.01 \). The friction force on the gas appears in the equation of motion as

\[ \frac{\partial u}{\partial t} = \ldots - \frac{\rho_p^{(i)}}{\rho} \left( \frac{\partial v^{(i)}}{\partial t} \right)_{\text{drag}} \]  

(95)

Here \( \rho_p^{(i)} \) is the dust density that particle \( i \) represents. This can be set through the parameter \texttt{eps\_todt} in \texttt{particle\_init\_pars}. The drag force is assigned from the particles onto the mesh using either NGP, CIC or TSC assignment. The same scheme is used both for interpolation and for assignment to avoid any risk of a particle accelerating itself (see Hockney & Eastwood 1981).

### 6.15 N-body solver

The \( N \)-body code takes advantage of the existing Particles module, which was coded with the initial intent of treating solid particles whose radius \( a_s \) is comparable to the mean free path \( \lambda \) of the gas, for which a fluid description is not valid. A \( N \)-body implementation based on that module only needed to include mass as extra state for the particles, solve for the \( N^2 \) gravitational pair interactions and distinguish between the \( N \)-body and the small bodies that are mapped into the grid as a \( \rho_p \) density field.

The particles of the \( N \)-body ensemble evolve due to their mutual gravity and by interacting with the gas and the swarm of small bodies. The equation of motion for particle \( i \) is
\[
\frac{dv_{pi}}{dt} = F_{gi} - \sum_{j \neq i}^{N} \frac{GM_j \hat{R}_{ij}}{R_{ij}^2} 
\]

where \( R_{ij} = |r_{pi} - r_{pj}| \) is the distance between particles \( i \) and \( j \), and \( \hat{R}_{ij} \) is the unit vector pointing from particle \( j \) to particle \( i \). The first term of the R.H.S. is the combined gravity of the gas and of the dust particles onto the particle \( i \), solved via

\[
F_{gi} = -G \int_V \frac{[\rho_g(r) + \rho_p(r)]R_i}{(R_i^2 + b_i^2)^{3/2}} dV, 
\]

where the integration is carried out over the whole disk. The smoothing distance \( b_i \) is taken to be as small as possible (a few grid cells). For few particles (<10), calculating the integral for every particle is practical. For larger ensembles one would prefer to solve the Poisson equation to calculate their combined gravitational potential.

The evolution of the particles is done with the same third-order Runge-Kutta time-stepping routine used for the gas. The particles define the timestep also by the Courant condition that they should not move more than one cell at a time. For pure particle runs, where the grid is absent, one can adopt a fixed time-step \( t_p \ll 2\pi \Omega_{fp}^{-1} \) where \( \Omega_{fp} \) is the angular frequency of the fastest particle.

By now (spring 2009), no inertial accelerations are included in the \( N \)-body module, so only the inertial frame - with origin at the barycenter of the \( N \)-body ensemble - is available. For a simulation of the circular restricted three-body problem with mass ratio \( q=10^{-3} \), the Jacobi constant of a test particle initially placed at position \( (x,y)=(2,0) \) was found to be conserved up to one part in \( 10^5 \) within the time span of 100 orbits.

We stress that the level of conservation is poor when compared to integrators designed to specifically deal with long-term \( N \)-body problems. These integrators are usually symplectic, unlike the Runge-Kutta scheme of the PENCIL CODE. As such, PENCIL should not be used to deal with evolution over millions of years. But for the time-span typical of astrophysical hydrodynamical simulations, this degree of conservation of the Jacobi constant can be deemed acceptable.

As an extension of the particle’s module, the \( N \)-body is fully compatible with the parallel optimization of PENCIL, which further speeds up the calculations. Parallelization, however, is not yet possible for pure particle runs, since it relies on splitting the grid between the processors. At the time of writing (spring 2009), the \( N \)-body code does not allow the particles to have a time-evolving mass.

### 6.16 Test-field equations

The test-field method is used to calculate turbulent transport coefficients for magnetohydrodynamics. This is a rapidly evolving field and we refer the interested reader to recent papers in this field, e.g. by Sur et al. (2008) or Brandenburg et al. (2008). For technical details, see also Sect. [E.3]

### 6.17 Gravitational wave equations

The expansion of the universe with time is described by the scale factor \( a(\tau) \), where \( \tau \) is the physical time. Using conformal time, \( t(\tau) = \int_0^\tau d\tau' / a(\tau') \), and dependent vari-
Table 5: Scale factor and conformal Hubble parameter for different values of $n$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$a$</th>
<th>$H$</th>
<th>$H_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>$\eta/2$</td>
<td>$1/\eta$</td>
<td>$1/\eta$</td>
</tr>
<tr>
<td>2/3</td>
<td>$\eta^2/3$</td>
<td>$2/\eta$</td>
<td>$2/\eta^2$</td>
</tr>
</tbody>
</table>

ables that are appropriately scaled with powers of $a$, the hydromagnetic equations can be expressed completely without scale factor [8, 15]. This is not true, however, for the gravitational wave (GW) equations, where a dependence on $a$ remains [15]. The time dependence of $a$ can be modeled as a power law, $a \propto \tau^n$, where $n = 1/2$ applies to the radiation-dominated era; see Table 5 the general relationship. To compare with cases where the expansion is ignored, we put $n = 0$.

In the transverse traceless (TT) gauge, the six components of the spatial part of the symmetric tensor characterizing the linearized evolution of the metric perturbations $h_{ij}$, reduce to two components which, in the linear polarization basis, are the + and $\times$ polarizations. However, the projection onto that basis is computationally intensive, because it requires nonlocal operations involving Fourier transformations. It is therefore advantageous to evolve instead the perturbation of the metric tensor, $h_{ij}$, in an arbitrary gauge, compute then $h_{ij}^{TT}$ in the TT gauge, and perform then the decomposition into the linear polarization basis whenever we compute diagnostic quantities such as averages or spectra. Thus, we solve the linearized GW equation in the form [15]

$$\frac{\partial^2 h_{ij}}{\partial t^2} = -2H \frac{\partial h_{ij}}{\partial t} + c^2 \nabla^2 h_{ij} + \frac{16\pi G}{a^2 c^2} T_{ij}$$  \hspace{1cm} (98)$$

for the six components $1 \leq i \leq j \leq 3$, where $t$ is comoving time, $a$ is the scale factor, $H = \dot{a}/a$ is the comoving Hubble parameter, $T_{ij}$ is the source term, $c$ is the speed of light, and $G$ is Newton’s constant. For $n = 0$, when the cosmic expansion is ignored, we have $a = 1$ and $H = 0$. We use the PENCIL CODE; for the numerical treatment of Eq. (98) and equations (100)–(102). For most of the simulations, we use 1152$^3$ meshpoints on 1152 cores of a Cray XC40 system with 2.3 GHz processors.

The source term is chosen to be the traceless part of the stress tensor,

$$T_{ij}(\mathbf{x}, t) = \rho u_i u_j - B_i B_j - \frac{1}{3} \delta_{ij} (\rho u^2 - B^2).$$  \hspace{1cm} (99)$$

The removal of the trace is in principle not necessary, but it helps preventing a continuous build-up of a large trace, which would be numerically disadvantageous. We have ignored here the viscous stress, which is usually small.

We compute $T_{ij}$ by solving the energy, momentum, and induction equations for an ultra-relativistic gas in the form [8, 10]

$$\frac{\partial \ln \rho}{\partial t} = -\frac{4}{3} (\nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla \ln \rho) + \frac{1}{\rho} \left[ \mathbf{u} \cdot (\mathbf{J} \times \mathbf{B}) + \eta \mathbf{J}^2 \right],$$  \hspace{1cm} (100)$$

$$\frac{\mathrm{D} \mathbf{u}}{\mathrm{D} t} = \frac{\mathbf{u}}{3} (\nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla \ln \rho) - \frac{\mathbf{u}}{\rho} \left[ \mathbf{u} \cdot (\mathbf{J} \times \mathbf{B}) + \eta \mathbf{J}^2 \right]$$
$$- \frac{1}{4} \nabla \ln \rho + \frac{3}{4\rho} \mathbf{J} \times \mathbf{B} + \frac{2}{\rho} \nabla \cdot (\rho \nu \mathbf{S}) + \mathbf{f},$$  \hspace{1cm} (101)$$
\[
\frac{\partial B}{\partial t} = \nabla \times (u \times B - \eta J),
\]
where \( B = \nabla \times A \) is the magnetic field expressed in terms of the magnetic vector potential to ensure that \( \nabla \cdot B = 0 \), \( J = \nabla \times B \) is the current density, \( D/\partial t + u \cdot \nabla \) is the advective derivative, \( S_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) - \frac{1}{2}\delta_{ij}u_{k,k} \) is the trace-free rate of strain tensor, and \( p = \rho c^2 \) is the pressure, where \( c_0 = c/\sqrt{3} \) is the sound speed for an ultra-relativistic gas. Lorentz-Heaviside units for the magnetic field are used.

We are interested in the rms value of the metric tensor perturbations and the GW energy density in the linear polarization basis. To compute \( h_{ij}^{TT} \) from \( h_{ij} \), we Fourier transform the six components of \( h_{ij} \) and \( h_{ij} \),

\[
\tilde{h}_{ij}(k, t) = \int h_{ij}(x, t) e^{-i k \cdot x} d^3 x \quad \text{for } 1 \leq i \leq j \leq 3
\]
and compute the components in the TT gauge as

\[
\tilde{h}_{ij}^{TT}(k, t) = (P_{il}P_{jm} - \frac{1}{2}P_{ij}P_{lm}) \tilde{h}_{lm}(k, t),
\]
where \( P_{ij} = \delta_{ij} - \hat{k}_i \hat{k}_j \) is the projection operator, and \( \hat{k} = k/k \) is the unit vector of \( k \), with \( k = |k| \) being the modulus. Next, we compute the linear polarization bases

\[
e^+_{ij} = e^1_i e^1_j - e^2_i e^2_j, \quad e^-_{ij} = e^1_i e^2_j + e^2_i e^1_j,
\]
where \( e^1 \) and \( e^2 \) are unit vectors perpendicular to \( k \). Thus

\[
\tilde{h}_+(k, t) = \frac{1}{2} e^+_{ij}(k) \tilde{h}_{ij}(k, t),
\]
\[
\tilde{h}_\times(k, t) = \frac{1}{2} e^-_{ij}(k) \tilde{h}_{ij}(k, t).
\]
We then return into real space and compute

\[
h_{+/\times}(x, t) = \int \tilde{h}_{+/\times}(k, t) e^{i k \cdot x} d^3 k/(2\pi)^3.
\]
Analogous calculations are performed for \( h_{+/\times}(x, t) \), which are used to compute the GW energy via

\[
\mathcal{E}_{GW}(t) = \frac{c^2}{32\pi G} \left( \langle \dot{h}_+^2 \rangle + \langle \dot{h}_\times^2 \rangle \right),
\]
where angle brackets denote volume averages.

Analogously to kinetic and magnetic energy and helicity spectra, it is convenient to compute the GW energy and polarization spectra integrated over concentric shells of surface \( \int_{4\pi} k^2 d\Omega_k \) in \( k \) space, defined by

\[
S_h(k) = \int_{4\pi} \left( |\dot{h}_+|^2 + |\dot{h}_\times|^2 \right) k^2 d\Omega_k,
\]
\[
A_h(k) = \int_{4\pi} 2 \text{Im} \left( \dot{h}_+ \dot{h}_\times^* \right) k^2 d\Omega_k,
\]
and normalized such that \( \int_0^\infty S_h(k) \, dk = \langle \dot{h}_+^2 \rangle + \langle \dot{h}_\times^2 \rangle \) is proportional to the energy density and \( \int_0^\infty A_h(k) \, dk \) is proportional to the polarized energy density. The \( A_h(k) \) spectra are not to be confused with the magnetic vector potential \( A(x, t) \). The corresponding GW energy spectra are noted by

\[
E_{GW}(k) = \left( c^2/32\pi G \right) S_h(k),
\]
\[
H_{GW}(k) = \left( c^2/32\pi G \right) A_h(k).
\]
We also define spectra for the metric tensor perturbation,

\[ S_h(k) = \int_{4\pi} \left( |\tilde{h}_+|^2 + |\tilde{h}_\times|^2 \right) k^2 d\Omega_k, \]
\[ A_h(k) = \int_{4\pi} 2 \text{Im} \left( \tilde{h}_+ \tilde{h}_\times^* \right) k^2 d\Omega_k, \]

which are normalized such that \( \int_0^\infty S_h(k) dk = h_{\text{rms}}^2 \) is the mean squared metric tensor perturbation.
7 Troubleshooting / Frequently Asked Questions

7.1 Download and setup

7.1.1 Download forbidden


7.1.2 When sourcing the ‘sourceme.sh’/‘sourceme.csh’ file or running pc_setupsrc, I get error messages from the shell, like ‘if: Expression Syntax.’ or ‘set: Variable name must begin with a letter.’

A: This sounds like a buggy shell setup, either by yourself or your system administrator — or a shell that is even more idiosyncratic than the ones we have been working with.

To better diagnose the problem, collect the following information before filing a bug report to us:

1. `uname -a`
2. `/bin/csh -v`
3. `echo $version`
4. `echo $SHELL`
5. `ps -p $$`
6. If you have problems while sourcing the ‘sourceme’ script,
   
   (a) unset the PENCIL_HOME variable:
   
   ```bash
   for csh and similar: unsetenv PENCIL_HOME
   for bash and similar: unexport PENCIL_HOME; unset PENCIL_HOME
   ```

   (b) switch your shell in verbose mode,
   
   ```bash
   for csh and similar: set verbose; set echo
   for bash and similar: set -v; set -x
   ```

   then source again.

7. If you have problems with pc_setupsrc, run it with csh in verbose mode:

```bash
/bin/csh -v -x $PENCIL_HOME/bin/pc_setupsrc
```
7.2 Compilation

7.2.1 Linker can’t find the syscalls functions:

ld: 0711-317 ERROR: Undefined symbol: .is_nan_c
ld: 0711-317 ERROR: Undefined symbol: .sizeof_real_c
ld: 0711-317 ERROR: Undefined symbol: .system_c
ld: 0711-317 ERROR: Undefined symbol: .get_env_var_c
ld: 0711-317 ERROR: Undefined symbol: .get_pid_c
ld: 0711-317 ERROR: Undefined symbol: .file_size_c

A: The Pencil Code needs a working combination of a Fortran- and a C-compiler. If this is not correctly set up, usually the linker won’t find the functions inside the syscalls module. If that happens, either the combination of C- and Fortran-compiler is inappropriate (e.g. ifort needs icc), or the compiler needs additional flags, like g95 might need the option ‘-fno-second-underscore’ and xlf might need the option ‘-qextname’. Please refer to Sect. 5.2, Table 1.

7.2.2 Make gives the following error now:

PGF90-S-0017-Unable to open include file: chemistry.h (nochemistry.f90: 43)
  0 inform, 0 warnings, 1 severes, 0 fatal for chemistry

Line 43 of the nochemistry routine, only has 'contains'.

A: This is because somebody added a new module (together with a corresponding nomodule.f90 and a module.h file (chemistry in this case). These files didn’t exist before, so you need to say:

    pc_setupsrcc

If this does not help, say first make clean and then pc_setupsrcc.

7.2.3 How do I compile the Pencil Code with the Intel (ifc) compiler under Linux?

A: The Pencil Code should compile successfully with ifc 6.x, ifc 7.0, sufficiently recent versions of ifc 7.1 (you should get the latest version; if yours is too old, you will typically get an ‘internal compiler error’ during compilation of ‘src/hydro.f90’), as well as with recent versions of ifort 8.1 (8.0 may also work).


On many current (as of November 2003) Linux systems, there is a mismatch between the glibc versions used by the compiler and the linker. To work around this, use the following flag for compiling

    FC=ifc -i_dynamic

and set the environment variable

    LD_ASSUME_KERNEL=2.4.1; export LD_ASSUME_KERNEL

or
setenv LD_ASSUME_KERNEL 2.4.1

This has solved the problems e.g. on a system with glibc-2.3.2 and kernel 2.4.22.
Thanks to Leonardo J. Milano (http://udel.edu/~lmilano/) for part of this info.

### 7.2.4 I keep getting segmentation faults with ‘start.x’ when compiling with ifort 8.0

A: There was/is a number of issues with ifort 8.0. Make sure you have the latest patches applied to the compiler. A number of things to consider or try are:

1. Compile with the the ‘-static -nothreads’ flags.
2. Set your stacksize to a large value (but a far too large value may be problematic, too), e.g.
   
   ```bash
   limit stacksize 256m
   ulimit -s 256000
   
   3. Set the environment variable KMP_STACKSIZE to a large value (like 100M)
   ```


### 7.2.5 When compiling with MPI on a Linux system, the linker complains:

mpicomm.o: In function ‘mpicomm_mpicomm_init_’:
mpicomm.o(.text+0x36): undefined reference to ‘mpi_init_’
mpicomm.o(.text+0x55): undefined reference to ‘mpi_comm_size_’
mpicomm.o(.text+0x6f): undefined reference to ‘mpi_comm_rank_’

A: This is the infamous underscore problem. Your MPI libraries have been compiled with G77 without the option ‘-fno-second-underscore’, which makes the MPI symbol names incompatible with other Fortran compilers.

As a workaround, use

```bash
MPICOMM = mpicomm_
```

in ‘Makefile.local’. Or, even better, you can set this globally (for the given computer) by inserting that line into the file ‘~/.adapt-mkfile.inc’ (see perldoc adapt-mkfile for more details).

### 7.2.6 Compilation stops with the cryptic error message:

```bash
f95 -O3 -u -c .f90.f90
Error : Could not open sourcefile .f90.f90
compilation aborted for .f90.f90 (code 1)
make[1]: *** [.f90.o] Error 1
```

What is the problem?

A: There are two possibilities:
1. One of the variables for `make` has not been set, so `make` expands it to the empty string. Most probably you forgot to specify a module in `src/Makefile.local`. One possibility is that you have upgraded from an older version of the code that did not have some of the modules the new version has.

Compare your `src/Makefile.local` to one of the examples that work.

2. One of the variables for `make` has a space appended to it, e.g. if you use the line

```
MPICOMM = mpicomm_ 
```

(see §[7.2.5](#)) with a trailing blank, you will encounter this error message. Remove the blank. This problem can also occur if you added a new module (and have an empty space after the module name in `src/Makefile.src`, i.e. CHIRAL=nochiral_), in which case the compiler will talk about “circular dependence” for the file ‘nochiral’.

### 7.2.7 The code doesn’t compile,

...there is a problem with `mvar`:

```
make start.x run.x
f95 -O3 -u -c cdata.f90
```

Error: cdata.f90, line 71: Implicit type for MVAR detected at MVAR@

[f95 terminated - errors found by pass 1]

```
make[1]: *** [cdata.o] Error 2
```

**A:** Check and make sure that ‘mkcparam’ (directory ‘$PENCIL_HOME/bin’) is in your path. If this doesn’t help, there may be an empty ‘cparam.inc’ file in your ‘src’ directory. Remove ‘cparam.inc’ and try again (Note that ‘cparam.inc’ is automatically generated from the ‘Makefile’).

### 7.2.8 Some samples don’t even compile,

as you can see on the web,[http://www.nordita.org/software/pencil-code/tests.html](http://www.nordita.org/software/pencil-code/tests.html)

samples/helical-MHDturb:

```
Compiling.. not ok:
make start.x run.x read_videofiles.x
make[1]: Entering directory '/home/dobler/f90/pencil-code/samples/helical-MHDturb/src'
/usr/lib/lam/bin/mpif95 -O3 -c initcond.f90
/usr/lib/lam/bin/mpif95 -O3 -c density.f90
use Gravity, only: gravz, nu_epicycle
```

Error 208 at (467:density.f90) : No such entity in the module

Error 355 : In procedure INIT_LNRHO variable NU_EPICYCLE has not been given a type

Error 355 : In procedure POLYTROPIC_LNRHO_DISC variable NU_EPICYCLE has not been given a

3 Errors

compilation aborted for density.f90 (code 1)

```
make[1]: *** [density.o] Error 1
```
make[1]: Leaving directory `/home/dobler/f90/pencil-code/samples/helical-MHDturb/src'
make: *** [code] Error 2

A: Somebody may have checked in something without having run auto-test beforehand. The problem here is that something has been added in one module, but not in the corresponding no-module. You can of course check with svn who it was...

7.2.9 Internal compiler error with Compaq/Dec F90

The Dec Fortran optimizer has occasional problems with ‘nompicomm.f90’:

```
maker.x run.x read_videofiles.x
f90  -fast -03 -tune ev6 -arch ev6  -c cparam.f90
[...]
f90  -fast -03 -tune ev6 -arch ev6  -c nompicomm.f90
otal vm 2755568  otal vm 2765296  otal vm 2775024
otal vm 2784752  otal...
Assertion failure: Compiler internal error - please submit problem r...
    GEM ASSERTION, Compiler internal error - please submit problem report
Fatal error in: /usr/lib/cmplrs/fort90_540/decfort90 Terminated
*** Exit 3
Stop.
*** Exit 1
Stop.
```

A: The occurrence of this problem depends upon the grid size; and the problem never seems to occur with ‘mpicomm.f90’, except when ncpus=1. The problem can be avoided by switching off the loop transformation optimization (part of the ‘-O3’ optimization), via:

```
#OPTFLAGS=-fast -03 -notransform_loops
```

This is currently the default compiler setting in ‘Makefile’, although it has a measurable performance impact (some 8% slowdown).

7.2.10 Assertion failure under SunOS

Under SunOS, I get an error message like

```
user@sun> f90 -c param_io.f90
Assertion failed: at_handle_table[at_idx].tag == VAR_TAG,
    file ../srcfw/FWcvrt.c, line 4018
f90: Fatal error in f90comp: Abort
```

A: This is a compiler bug that we find at least with Sun’s WorkShop Compiler version ‘5.00/05/17 FORTRAN 90 2.0 Patch 107356-05’. Upgrade the compiler version (and possibly also the operating system): we find that the code compiles and works with version ‘Sun WorkShop 6 update 2 Fortran 95 6.2 Patch 111690-05 2002/01/17’ under SunOS version ‘5.8 Generic_108528-11’.
7.2.11 *After some dirty tricks I got pencil code to compile with MPI, ...*

> Before that i installed lam-7.1.4 from source.

Goodness gracious me, you shouldn’t have to compile your own MPI library.

**A:** Then don’t use the old LAM-MPI. It is long superseded by open-mpi now. Open-mpi doesn’t need a daemon to be running. I am using the version that ships with Ubuntu (e.g. 9.04):

```
frenesi:~> aptitude -w 210 search openmpi | grep '^i'
```

```
i libopenmpi-dev - high performance message passing library -- header files
i A libopenmpi1 - high performance message passing library -- shared library
i openmpi-bin - high performance message passing library -- binaries
i A openmpi-common - high performance message passing library -- common files
i openmpi-doc - high performance message passing library -- man pages
```

Install that and keep your configuration (Makefile.src and getconf.csh) close to that for ‘frenesi’ or ‘norlx50’. That should work.

7.2.12 *Error: Symbol ‘mpi_comm_world’ at (1) has no IMPLICIT type*

I installed the pencil code on Ubuntu system and tested "run.csh" in `samples\conv-slab`. Here the code worked pretty well.

Nevertheless, running (auto-test), I found there are some errors.

The messages are,

```
Error: Symbol ‘mpi_comm_world’ at (1) has no IMPLICIT type
Fatal Error: Error count reached limit of 25.
make[2]: *** [mpicomm_double.o] Error 1
make[2]: Leaving directory
‘/home/pkiwan/Desktop/pencil-code/samples/2d-tests/selfgravitating-shearwave/src’
make[1]: *** [code] Error 2
make[1]: Leaving directory
‘/home/pkiwan/Desktop/pencil-code/samples/2d-tests/selfgravitating-shearwave/src’
make: *** [default] Error 2
```

Finally, ### auto-test failed ###

Will it be OK? Or, how can I fix this?

**A:** Thanks for letting me know about the status, and congratulations on your progress! Those tests that fail are those that use MPI. If your machine is a dual or multi core machine, you could run faster by running under MPI. But this is probably not crucial for you at this point. (I just noticed that there is a ToDo listed in the auto-test command to implement the option not to run the MPI tests, but this hasn’t been done yet. So I guess you can start with the science next.
7.2.13 Error: Can’t open included file ‘mpif.h’

It always worked, but now, after some systems upgrade, I get

```bash
 gfortran -O3 -o mpicomm.o -c mpicomm.f90
 Error: Can’t open included file ’mpif.h’
```

When I say `locate mpif.h` I only get things like

```
/scratch/ntest/1.2.7p1-intel/include/mpif.h
```

But since I use `FC=mpif90` I thought I don’t need to worry.

**A:** Since you use `FC=mpif90` there must definitely be something wrong with their setup. Try `mpif90 -showme` or `mpif90 -show`; the ‘-I’ option should say where it looks for ’mpif.h’. If those directories don’t exist, it’s no wonder that it doesn’t work, and it is time to complain.

### 7.3 Pencil check

#### 7.3.1 The pencil check complains for no reason.

**A:** The pencil check only complains for a reason.

#### 7.3.2 The pencil check reports MISSING PENCILS and quits

**A:** This could point to a serious problem in the code. Check where the missing pencil is used in the code. Request the right pencils, likely based on input parameters, by adapting one or more of the `pencil_criteria_MODULE` subroutines.

#### 7.3.3 The pencil check reports unnecessary pencils

The pencil check reports possible overcalculation... pencil rho (43) is requested, but does not appear to be required!

**A:** Such warnings show that your simulation is possibly running too slowly because it is calculating pencils that are not actually needed. Check in the code where the unnecessary pencils are used and adapt one or more of the `pencil_criteria_MODULE` subroutines to request pencils only when they are actually needed.

#### 7.3.4 The pencil check reports that most or all pencils are missing

**A:** This is typically a thing that can happen when testing new code development for the first time. It is usually an indication that the reference \( df \) changes every time you call `pde`. Check whether any newly implemented subroutines or functionality has a “memory”, i.e. if calling the subroutine twice with the same \( f \) gives different output \( df \).
7.3.5 Running the pencil check triggers mathematical errors in the code

A: The pencil check puts random numbers in $f$ before checking the dependence of $df$ on the chosen set of pencils. Sometimes these random numbers are inconsistent with the physics and cause errors. In that case you can set lrandom_f_pencil_check=F in $\&$run-_pars in ‘run.in’. The initial condition may contain many idealized states (zeros or ones) which then do not trigger pencil check errors when lrandom_f_pencil_check=F, even if pencils are missing. But it does prevent mathematical inconsistencies.

7.3.6 The pencil check still complains

A: Then you need to look into the how the code and the pencil check operate. Reduce the problem in size and dimensions to find the smallest problem that makes the pencil check fail (e.g. 1x1x8 grid points). At the line of ‘pencil_check.f90’ when a difference is found between $df_{\text{ref}}$ and $df$, add some debug lines telling you which variable is inconsistent and in what place. Often you will be surprised that the pencil check has correctly found a problem in the simulation.

7.3.7 The pencil check is annoying so I turned it off

A: Then you are taking a major risk. If one or more pencils are not calculated properly, then the results will be wrong.

7.4 Running

7.4.1 Why does ‘start.x’ / ‘start.csh’ write data with periodic boundary conditions?

A: Because you are setting the boundary conditions in ‘run.in’, not in ‘start.in’; see Sect. 5.16.1. There is nothing wrong with the initial data — the ghost-zone values will be re-calculated during the very first time step.

7.4.2 csh problem?

Q: On some rare occasions we have problems with csh not being supported on other machines. (We hope to fix this by contacting the responsible person, but may not be that trivial today!) Oliver says this is a well known bug of some years ago, etc. But maybe in the long run it would be good to avoid csh.

A: These occasions will become increasingly frequent, and eventually for some architectures, there may not even be a csh variant that can be installed.

We never pushed people to use pc_run and friends (and to report corresponding bugs and get them fixed), but if we don’t spend a bit of effort (or annoy users) now, we create a future emergency, where someone needs to run on some machine, but there is no csh and he or she just gets stuck.

We don’t have that many csh files, and for years now it should be possible to compile run without csh (using bin/pc_run) — except that people still fall back on the old way of
doing things. This is both cause and consequence of the ‘new’ way not being tested that much, at least for the corner cases like ‘RERUN’, ‘NEWDIR’, ‘SCRATCH_DIR’.

### 7.4.3 ‘run.csh’ doesn’t work:

Invalid character ‘‘’ in NAMELIST input
Program terminated by fatal I/O error
Abort

A: The string array for the boundary condition, e.g. $bcx$ or $bcz$ is too long. Make sure it has exactly as many elements as $nvar$ is big.

### 7.4.4 Code crashes after restarting

> > removing $mu_r$ from the namelist just ‘like that’ makes the code
> > backwards incompatible.
> >
> > That means that we can never get rid of a parameter in start.in once we have introduced it, right?

A: In the current implementation, without a corresponding cleaning procedure, unfortunately yes.

Of course, this does not affect users’ private changes outside the central svn tree.

### 7.4.5 auto-test gone mad...?

Q: Have you ever seen this before:

```
giga01:/home/pg/n7026413/cvs-src/pencil-code/samples/conv-slab> auto-test

/home/pg/n7026413/cvs-src/pencil-code/samples/conv-slab:
  Compiling... ok
  No data directory; generating data -> /var/tmp/pencil-tmp-25318
  Starting... ok
  Running... ok
  Validating results... Malformed UTF-8 character (unexpected continuation byte 0x80, with no preceding start byte) in split at /home/pg/n7026413/cvs-src/pencil-code/bin/auto-test line 263.
```

A: You are running on a RedHat 8 or 9 system, right?

Set $\text{LANG}=$\text{POSIX}$ in your shell’s startup script and life will be much better.
7.4.6 Can I restart with a different number of cpus?

Q: I am running a simulation of nonhelical turbulence on the cluster using MPI. Suppose if I am running a $128^3$ simulation on 32 cpus/cores i.e.

```fortran
integer, parameter :: ncpus=32, nprocy=2, nprocz=ncpus/nprocy, nprocx=1
integer, parameter :: nxgrid=128, nygrid=nxgrid, nzgrid=nxgrid
```

And I stop the run after a bit. Is there a way to resume this run with different number of cpus like this:

```fortran
integer, parameter :: ncpus=16, nprocy=2, nprocz=ncpus/nprocy, nprocx=1
integer, parameter :: nxgrid=128, nygrid=nxgrid, nzgrid=nxgrid
```

I understand it has to be so in a new directory but making sure that the run starts from where I left it off in the previous directory.

A: The answer is no, if you use the standard distributed io. There is also parallel io, but I never used it. That would write the data in a single file, and then you could use the data for restart in another processor layout.

7.4.7 Can I restart with a different number of cpus?

Q: Is it right that once the simulation is resumed, pencil-code takes the last data from var.dat (which is the current snapshot of the fields)? If that is true, then, is it not possible to give that as the initial condition for the run in the second directory (with changed "ncpus")? Is there a mechanism already in place for that?

A: Yes, the code restarts from the last var.dat. It is written after a successful completion of the run, but it crashes or you hit a time-out, there will be a var.dat that is overwritten every isave timesteps. If the system stops during writing, some var.dat files may be corrupt or have the wrong time. In that case you could restart from a good VAR file, if you have one, using, e.g.,

```
restart-new-dir-VAR . 46
```

where 46 is the number of your VAR file, i.e., VAR46 in this case. To restart in another directory, you say, from the old run directory,

```
restart-new-dir ../another_directory
```

Hope this helps. Look into pencil-code/bin/restart-new-dir to see what it is doing.

7.4.8 fft_xyzParallel_3D: nygrid needs to be an integer multiple...

Q: I just got an:

```
fft_xyzParallel_3D: nygrid needs to be an integer multiple of nprocy*nprocz
```

In my case, nygrid=2048, nprocy=32, and nprocz=128, so nprocy*nprocz=4096. In other words, 2048 needs to be a multiple of 4096. But isn't this the case then?

A: No, because 2048 = 0.5 * 4096 and 0.5 is not an integer. Maybe try either setting nprocz=64 or nprocy=64. You could compensate the change of ncpus with the $x$-direction.
7.4 Running

For $2048^3$ simulations, $nprocy=32$ and $nprocz=64$ would be good. A list of good meshes is given in Table 4.

7.4.9 Unit-agnostic calculations?

Q: The manual speaks about unit-agnostic calculations, stating that one may choose to interpret the results in any (consistent) units, depending on the problem that is solved at hand. So, for example, if I chose to run the ‘2d-tests/battery_term’ simulation for an arbitrary number of time-steps and then choose to examine the diagnostics, am I correct in assuming the following:

1) $[\text{Brms}] = \text{Gauss}$ (as output by unit_magnetic, before the run begins)
2) $[t] = \text{s}$ (since the default unit system is left as CGS)
3) $[\text{urms}] = \text{cm/s}$ (again, as output by unit_velocity, before the run begins)
4) and etc. for the units of the other diagnostics

A: Detailed correspondence on this item can be found on:
https://groups.google.com/forum/?fromgroups#!topic/pencil-code-discuss/zek-uYNbgXI
There is also working material on unit systems under
http://www.nordita.org/~brandenb/teach/PencilCode/MixedTopics.html with a link to
http://www.nordita.org/~brandenb/teach/PencilCode/material/AlfvenWave_SIunits/
Below is a pedagogical response from Wlad Lyra:

In the sample battery-term, the sound speed $c_s=1$ sets the unit of velocity. Together with the unit of length, that sets your unit of time. The unit of magnetic field follows from the unit of velocity, density, and your choice of magnetic permittivity, according to the definition of the Alfven velocity.

If you are assuming cgs, you are saying that your sound speed $c_s=1$ actually means $[U]=1 \text{ cm/s}$. Your unit of length is equivalently 1 cm, and therefore the unit of time is $[t] = [L]/[U] = 1 \text{ s}$. The unit of density is $[\rho] = 1 \text{ g/cm}^3$. Since in cgs $v_A=B/\sqrt{4\pi\rho}$, your unit of magnetic field is $[B] = [U] * \sqrt{[\rho] * 4\pi}$ $\approx 3.5 \text{ sqrt(g/cm) / s} = 3.5 \text{ Gauss}$.

If instead you are assuming SI, you have $c_s=1$ assuming that means $[U]=1 \text{ m/s}$ and $\rho_0=1$ assuming that to mean $[\rho]=1 \text{ kg/m}^3$. Using $[L]=1 \text{ m}$, you have still $[t]=1 \text{ s}$, but now what appears as $B=1$ in your output is actually $[B] = [U] * \sqrt{(\mu_0 * [\rho])} = 1 \text{ m/s} * \sqrt{4\pi * 1e-7 \text{ N*A}^{-2} / 1 \text{ kg/m}^3} \approx 0.0011210 \text{ kg/(s}^2\text{A)} \approx 11 \text{ Gauss}$.

You can make it more interesting and use units relevant to the problem. Say you are at the photosphere of the Sun. You may want to use dimensionless $c_s=1$ meaning a sound speed of 10 km/s. Your appropriate length can be a megameter. Now your time unit is $[t]=[L]/[U] = 1e3 \text{ km/} 10 \text{ km/s} = 10^2 \text{ s}$, i.e., roughly 1.5 minute. For density, assume $\rho_0=2x10^{-4} \text{ kg/m}^3$, typical of the solar photosphere. Your unit of magnetic field is therefore $[B] = [U] * \sqrt{[\rho]} = 1e6 \text{ cm/s} * \sqrt{4\pi * 2e-7 \text{ g/cm}^3} \approx 1585.33 \text{ Gauss}$.

Notice that for $\mu_0=1$ and $\rho_0=1$ you simply have $v_A=B$. Then you can conveniently set the field strength by your choice of plasma beta (=
2*cs^-2/vA^-2). There’s a reason why we like dimensionless quantities!

7.5 Visualization

7.5.1 ‘start.pro’ doesn’t work:

Reading grid.dat..
Reading param.nml..
/% Expression must be a structure in this context: PAR.
/% Execution halted at: \$MAIN\$   104
/home/brandenb/pencil-code/runs/forced/hel1/.../.../.../idl/start.pro

A: You don’t have the subdirectory ‘data’ in your IDL variable '/path'. Make sure you source ‘sourceme.csh’/‘sourceme.sh’ or set a sufficient IDL path otherwise.

7.5.2 ‘start.pro’ doesn’t work:

Isn’t there some clever (or even trivial) way that one can avoid the annoying error messages that one gets, when running e.g. ”.r rall” after a new variable has been introduced in ”idl/varcontent.pro”? Ever so often there’s a new variable that can’t be found in my param2.nml – this time it was IECR, IGG, and ILNTT that I had to circumvent...

A: The simplest solution is to invoke ‘NOERASE’, i.e. say

touch NOERASE
start.csh

or, alternatively, start_run.csh. What it does is that it reruns src/start.x with a new version of the code; this then produces all the necessary auxiliary files, but it doesn’t overwrite or erase the ‘var.dat’ and other ‘VAR’ and ‘slice’ files.

7.5.3 Something about tag name undefined:

Q: In one of my older run directories I can’t read the data with idl anymore. What should I do? Is says something like

Reading param.nml..
% Tag name LEQUIDIST is undefined for structure <Anonymous>.
% Execution halted at: $MAIN$   182
/people/disk2/brandenb/pencil-code/idl/start.pro

A: Go into ‘data/param.nml’ and add , LEQUIDIST=T anywhere in the file (but before the last slash).

7.5.4 Something INC in start.pro

Q: start doesn’t even work:
% Compiled module: $MAIN$.
nname= 11

Reading grid.dat..
Reading param.nml..

Can’t locate Namelist.pm in INC (INC contains: /etc/perl
/usr/local/lib/perl/5.8.4 /usr/local/share/perl/5.8.4
BEGIN failed--compilation aborted at /home/brandenb/pencil-code/bin/nl2idl line 49.

A: Go into ‘$PENCIL_HOME’ and say svn up sourceme.csh and/or svn up sourceme.sh.
(They were just out of date.)

7.5.5 nl2idl problem when reading param2.nml

Q: Does anybody encounter a backward problem with nl2idl? The file param*.nml files
are checked in under ‘pencil-code/axel/couette/SStrat128a_mu0.20_g2’ and the problem
is below.

at /people/disk2/brandenb/pencil-code/bin/nl2idl line 120
HCOND0= 0.0,HCOND1= 1.000000,HCOND2= 1.000000,WIDTHSS= 1.192093E-06,MPOLY0=
^------ HERE
at /people/disk2/brandenb/pencil-code/bin/nl2idl line 120

A: The problem is the stupid ifc compiler writing the following into the namelist file:

    COOLING_PROFILE='gaussian', COOLTYPE='Temp
    'COOL= 0.0, CS2COOL= 0.0, RCOOL= 1.000000, WCODL= 0.1000000, FBOT= 0.0, CHI_T= 0.0

If you add a comma after the closing quote:

    COOLING_PROFILE='gaussian', COOLTYPE='Temp
    ', COOL= 0.0, CS2COOL= 0.0, RCOOL= 1.000000, WCODL= 0.1000000, FBOT= 0.0, CHI_T= 0.0

things will work.

Note that ifc cannot even itself read what it is writing here, so if this happened to occur
in param.nml, the code would require manual intervention after each start.csh.

7.5.6 Spurious dots in the time series file

Q: Wolfgang, you explained it to me once, but I forget. How can one remove spurious
dots after the timestep number if the time format overflows?

A: I don’t know whether it exists anywhere, but it’s easy. In Perl you’d say

    perl -pe 's/\(^s*\([-0-9]+)\)\.(\([-0-9eEdD])$/\1 $2/g'

and in sed (but that’s harder to read)

    sed 's/\(^ [ -0-9]+ \)\.(\([-0-9eEdD])\)/\1 \2/g'

7.5.7 Problems with pc_varcontent.pro

Q:
% Subscript range values of the form low:high must be >= 0, < size, with low <= high: VARCONTENT.
% Error occurred at: PC_VARCONTENT 391
/home/brandenb/pencil-code/idl/read/pc_varcontent.pro
% PC_READ_VAR 318
/home/brandenb/pencil-code/idl/read/pc_read_var.pro
% $MAIN$

A: Make sure you don’t have any unused items in your src/cparam.local such as
! MAUX CONTRIBUTION 3
! COMMUNICATED AUXILIARIES 3
They would leave gaps in the counting of entries in your data/index.pro file.

7.6 General questions

7.6.1 “Installation” procedure

Why don’t you use GNU autoconf/automake for installation of the PENCIL CODE?

A: What do you mean by “installation”? Unlike the applications that normally use autoconf, the Pencil Code is neither a binary executable, nor a library that you compile once and then dump somewhere in the system tree. Autoconf is the right tool for these applications, but not for numerical codes, where the typical compilation and usage pattern is very different:

You have different directories with different ‘Makefile.local’ settings, recompile after introducing that shiny new term in your equations, etc. Moreover, you want to sometimes switch to a different compiler (but just for that run directory) or another MPI implementation. Our adapt-mkfile approach gives you this flexibility in a reasonably convenient way, while doing the same thing with autoconf would be using that system against most of its design principles.

Besides, it would really get on my (WD’s) nerves if I had to wait two minutes for autoconf to finish before I can start compiling (or maybe 5–10 minutes if I worked on a NEC machine...).

Finally, if you have ever tried to figure out what a ‘configure’ script does, you will appreciate a comprehensible configuration system.

7.6.2 Small numbers in the code

What is actually the difference between epsi, tini and tiny?

A:

F90 has two functions epsilon() and tiny(), with

\[
\text{epsilon}(x) = 1.1920929 \times 10^{-7} \\
\text{tiny}(x) = 1.1754944 \times 10^{-38} \\
(\text{and then there is huge}(x) = 3.4028235 \times 10^{38})
\]
for a single-precision number $x$.

$\epsilon(x)$ is the smallest number that satisfies

$$1 + \epsilon(1.) \neq 1,$$

while $\text{tiny}(x)$ is the smallest number that can be represented without precision loss.

In the code we have variants hereof,

\begin{verbatim}
epsi=5*epsilon(1.0)
tini=5*tiny(1.0)
huge1=0.2*huge(1.0)
\end{verbatim}

that have added safety margins, so we don't have to think about doing things like $1/\text{tini}$.

So in sub.f90,

\begin{verbatim}
-   evr = evr / spread(r_mn+epsi,2,3)
+   evr = evr / spread(r_mn+tini,2,3)
\end{verbatim}

did (minimally) affect the result for $r_{mn}=O(1)$, while the correct version only avoids overflow.

### 7.6.3 Why do we need a /lphysics/namelist in the first place?

Wolfgang answered on 29 July 2010: “'cdata.f90' has the explanation”

\begin{verbatim}
! Constant 'parameters' cannot occur in namelists, so in order to get the ! now constant module logicals into the lphysics name list... ! We have some proxies that are used to initialize private local variables ! called lhydro etc, in the lphysics namelist!
\end{verbatim}

So the situation is this: we want to write parameters like ldensity to param.nml so IDL (and potentially octave, python, etc.) can know whether density was on or not. To avoid confusion, we want them to have exactly their original names. But we cannot assemble the original ldensity etc. constants in a namelist, so we have to define a local ldensity variable. And to provide it with the value of the original cdata.ldensity, we need to transfer the value via ldensity\_var. That’s pretty scary, although it seems to work fine. I can track the code back to the big eos\_merger commit, so it may originate from that branch. One obvious problem is that you have to add code in a number of places (the ldensity → ldensity\_var assignment and the local definition of ldensity) to really get what you need. And when adding a new boolean of that sort to 'cdata.f90', you may not even have a clue that you need all the other voodoo.

There may be a cleaner solution involving generated code. Maybe something like

\begin{verbatim}
logical :: ldensity ! INCLUDE_IN_LPHYSICS
\end{verbatim}

could later generate code (in some param\_io\_extra.inc file) that looks like this:

\begin{verbatim}
write(unit, *) 'ldensity = ', ldensity
\end{verbatim}

i.e. we can manually write in namelist format. But maybe there are even simpler solutions?
7.6.4 Can I run the code on a Mac?

**A**: Macs work well for Linux stuff, except that the file structure is slightly different. Problems when following Linux installs can usually be traced to the PATH. For general reference, if you need to set an environment variable for an entire OS-X login session, google environment.plist. That won’t be needed here.

For a Mac install, the following should work:

a) Install Dev Tools (an optional install on the MacOS install disks). Unfortunately, last time I checked the svn version that comes with DevTools is obsolete. So:

b) Install MacPorts (download from web). Note that MacPorts installs to a non-standard location, and will need to be sourced. The installation normally drops an appropriate line in .profile. If it does so, make sure that that line gets sourced. Otherwise

   ```
   export PATH=/opt/local/bin:/opt/local/sbin:$PATH
   export MANPATH=/opt/local/share/man:$MANPATH
   ```

c) Install g95 (download from web). Make sure it is linked in /bin.

d) execute macports svn install

e) download the pencil-code and enjoy.

Note: the above way to get svn works. It takes a while however, so there are certainly faster ways out there. If you already have a non-obsolete svn version, use that instead.

7.6.5 Pencil Code discussion forum

Do I just need to send an email somewhere to subscribe or what?

**A**: The answer is yes; just go to:

   http://groups.google.com/group/pencil-code-discuss

7.6.6 The manual

It would be a good idea to add this useful information in the manual, no?

**A**: When you have added new stuff to the code, don’t forget to mention this in the ‘pencil-code/doc/manual.tex’ file.

Again, the answer is yes; just go to:

   cd pencil-code/doc/
   vi manual.tex
   svn ci -m "explanations about a new module in the code"
Part II

Programming the PENCIL Code

All developers are supposed to have an up-to-date entry in the file ‘pencil-code/license/developers.txt’ so that they can be contacted in case a code change breaks an auto-test or other code functionality.

Several PENCIL Code committers have done several hundred check-ins, but many of the currently 68 registered people on the repository have hardly done anything. To put a number to this, one can define an $h$ index, which gives the number of users, who have done at least as many as that number of check-ins. This $h$ index is currently 32, i.e., 32 users have done at least 32 check-ins; see Figure 6.

![Figure 6](image)

Figure 6: The $h$ index of PENCIL Code check-ins.

The PENCIL Code has expanded approximately linearly in the number of lines of code and the number of subroutines (Fig. 7). The increase in the functionality of the code is documented by the rise in the number of sample problems (Fig. 8). It is important to monitor the performance of the code as well. Figure 9 shows that for most of the runs the run time has not changed much.

Before making changes to the code, it is important that you verify that you can run the `pc_auto-test` successfully. Don’t do this when you have already modified the code, because then you cannot be sure that any problems are caused by your changes, or because it wouldn’t have worked anyway. Also, keep in mind that the code is public, so your changes should make sense from a broader perspective and should not only be intended for yourself. Regarding more general aspects about coding standards see Sect. B.2.

In order to keep the development of the code going, it is important that the users are able to understand and modify (program!) the code. In this section we explain first how
Figure 7: Number of lines of code and the number of subroutines since the end of 2001. The jump in the Summer of 2005 was the moment when the developments on the side branch (eos branch) were merged with the main trunk of the code. Note the approximately linear scaling with time.

to orient yourself in the code and to understand what is in it, and then to modify it according to your needs.

The Pencil Code check-ins occur regularly all the time. By the Pencil Code User Meeting 2010 we have arrived at a revision number of 15,000. In February 2017, the number of check-ins has risen to 26,804; see [https://github.com/pencil-code/pencil-code](https://github.com/pencil-code/pencil-code). Major code changes are nowadays being discussed by the Pencil Code Steering Committee ([https://www.nordita.org/~brandenb/pencil-code/PCSC/](https://www.nordita.org/~brandenb/pencil-code/PCSC/)). The increase of the revision number with time is depicted in Figure 10. The number of Pencil Code developers increases too (Figure 11), but the really active ones are getting rare. This may indicate that new users can produce new science with the code as it is, but it may also indicate that it is getting harder to understand the code. How to understand the code will be discussed in the next section.

Figure 8: Number of tests in the sample directory that are used in the nightly auto tests. Note again the approximately linear scaling with time.
Figure 9: Run time of the daily auto-tests since August 17, 2008. For most of the runs the run time has not changed much. The occasional spikes are the results of additional load on the machine.
Figure 10: Number of check-ins since 2002. Note again the linear increase with time, although in the last part of the time series there is a notable speed-up.

Figure 11: Check-ins since 2002 per user. Users with more than 100 check-ins are color coded.
8 Understanding the code

Understanding the code means looking through the code. This is not normally done by just printing out the entire code, but by searching your way through the code in order to address your questions. The general concept will be illustrated here with an example.

8.1 Example: how is the continuity equation being solved?

All the physics modules are solved in the routine pde, which is located in the file and module ‘Equ’. Somewhere in the pde subroutine you find the line

```fortran
    call dlhrho_dt(f,df,p)
```

This means that here the part belonging to $\partial \ln \rho / \partial t$ is being assembled. Using the grep command you will find that this routine is located in the module density, so look in there and try to understand the pieces in this routine. We quickly arrive at the following crucial part of code,

```fortran
    if (lcontinuity_gas) then
        if (ldensity_nolog) then
            df(l1:l2,m,n,irho) = df(l1:l2,m,n,irho) - p%ugrh - p%rho*p%divu
        else
            df(l1:l2,m,n,ilnrho) = df(l1:l2,m,n,ilnrho) - p%uglnrho - p%divu
        endif
    endif
```

where, depending on some logicals that tell you whether the continuity equation should indeed be solved and whether we do want to solve for the logarithmic density and not the actual density, the correct right hand side is being assembled. Note that all these routines always only add to the existing $df(l1:l2,m,n,ilnrho)$ array and never reset it. Resetting $df$ is only done by the timestepping routine. Next, the pieces $p%uglnrho$ and $p%divu$ are being subtracted. These are pencils that are organized in the structure with the name p. The meaning of their names is obvious: $uglnrho$ refers to $u \cdot \nabla \ln \rho$ and $divu$ refers to $\nabla \cdot u$. In the subroutine pencil_criteria_density you find under which conditions these pencils are requested. Using grep, you also find where they are calculated. For example $p%uglnrho$ is calculated in ‘density.f90’; see

```fortran
    call u_dot_grad(f,ilnrho,p%glnrho,p%uu,p%uglnrho,UPWIND=lupw_lnrho)
```

So this is a call to a subroutine that calculates the $u \cdot \nabla$ operator, where there is the possibility of upwinding, but this is not the default. The piece $divu$ is calculated in ‘hydro.f90’ in the line

```fortran
    if (pencil(i_uuj)) call gij(f,iiu,p%uu,p%uglnrho)
    if (pencil(i_divu)) call div_mn(p%uu,p%divu,p%uu)
```
Note that the divergence calculation uses the velocity gradient matrix as input, so no new derivatives are recalculated. Again, using grep, you will find that this calculation and many other ones happen in the module and file 'sub.f90'. The various derivatives that enter here have been calculated using the $g_{ij}$ routine, which calls the $der$ routine, e.g., like so

\begin{verbatim}
  k1=k-1
  do i=1,3
    do j=1,3
      if (nder==1) then
        call der(f,k1+i,tmp,j)
    endif
  enddo
\end{verbatim}

For all further details you just have to follow the trail. So if you want to know how the derivatives are calculated, you have to look in deriv.f90, and only here is it where the indices of the $f$ array are being addressed.

If you are interested in magnetic fields, you have to look in the file 'magnetic.f90'. The right hand side of the equation is assembled in the routine

\begin{verbatim}
!**************************************************************
  subroutine daa_dt(f,df,p)
  !
  ! Magnetic field evolution.
  !
  ! Calculate $dA/dt=uxB+3/2 \Omega_0 A_y \mathbf{x}_\text{dir} \,-\, \eta \mu_0 J$.
  ! For mean field calculations one can also add $dA/dt=\ldots+\alpha*bb+\delta*\mathbf{WXJ}$.
  ! Add $jxb/rho$ to momentum equation.
  ! Add $\eta \mu_0 j2/rho$ to entropy equation.
  !
  where the header tells you already a little bit of what comes below. It is also here where ohmic heating effects and other possible effects on other equations are included, e.g.
  !
  ! Add Ohmic heat to entropy or temperature equation.
  !
  if (lentropy .and. lohmic_heat) then
    df(l1:l2,m,n,iss) = df(l1:l2,m,n,iss) &
    + etatotal*mu0*p%j2*p%rho1*p%TT1
  endif
\end{verbatim}

We leave it at this and encourage the user to do similar inspection work on a number of other examples. If you think you find an error, file a ticket at http://code.google.com/p/pencil-code/issues/list. You can of course also repair it!
9 Adapting the code

9.1 The PENCIL Code coding standard

As with any code longer than a few lines the appearance and layout of the source code is of the utmost importance. Well laid out code is more easy to read and understand and as such is less prone to errors.

A consistent coding style has evolved in the PENCIL Code and we ask that those contributing try to be consistent for everybody's benefit. In particular, it would be appreciated if those committing changes of existing code via svn follow the given coding style.

There are not terribly many rules and using existing code as a template is usually the easiest way to proceed. In short the most important rules are:

- tab characters do not occur anywhere in the code (in fact the use of tab character is an extension to the Fortran standard).

- Code in any delimited block, e.g. if statements, do loops, subroutines etc., is indented be precisely 2 spaces. E.g.

  ```fortran
  if (l_cylindrical) then
      call fatal_error('del2fjv','del2fjv not implemented')
  endif
  ```

- continuation lines (i.e. the continuation part of a logical line that is split using the & sign) are indented by 4 spaces. E.g. (note the difference from the previous example)

  ```fortran
  if (l_cylindrical) &
      call fatal_error('del2fjv','del2fjv not implemented')
  ```

- There is always one space separation between 'if' and the criterion following in parenthesis:

  ```fortran
  if (ldensity_nolog) then
      rho=f(l1:l2,m,n,irho)
  endif
  ```

  This is wrong:

  ```fortran
  if(ldensity_nolog) then ! WRONG
      rho=f(l1:l2,m,n,irho)
  endif
  ```

- In general, try to follow common practice used elsewhere in the code. For example, in the code fragment above there are no empty spaces within the mathematical expressions programmed in the code. A unique convention helps in finding certain expressions and patterns in the code. However, empty spaces are often used after commas and semicolons, for examples in name lists.

- Relational operators are written with symbols (==, /=, <, <=, >, >=), not with characters (.eq., .ne., .lt., .le., .gt., .ge.).

- In general all comments are placed on their own lines with the "!" appearing in the first column.
9.2 Adding new output diagnostics

With the implementation of new physics and the development of new procedures it will become necessary to monitor new diagnostic quantities that have not yet been implemented in the code. In the following, we describe the steps necessary to set up a new diagnostic variable.

This is nontrivial as, in order to keep latency effects low on multi-processor machines, the code minimizes the number of global reduction operations by assembling all quantities that need the maximum taken in $f_{\text{max}}$, and those that need to be summed up over all processors (mostly for calculating mean quantities) in $f_{\text{sum}}$ (see subroutine diagnostic in file ‘src/equ.f90’).

As a sample variable, let us consider $jbm$ (the volume average $\langle j \cdot B \rangle$). Only the module $\text{magnetic}$ will be affected, as you can see (the diagnostic quantity $jbm$ is already implemented) with

```
unix> grep -i jbm src/*.f90
```

If we pretend for the sake of the exercise that no trace of $jbm$ was in the code, and we were only now adding it, we would need to do the following

1. add the variable $\text{id}iag_{\text{jbm}}$ to the module variables of $\text{Magnetic}$ in both ‘magnetic.f90’ and ‘nomagnetic.f90’:

   ```
   integer :: idiag_jbm=0
   ```

   The variable $\text{id}iag_{\text{jbm}}$ is needed for matching the position of $jbm$ with the list of diagnostic variables specified in ‘print.in’.

2. in the subroutine $\text{daa}_\text{dt}$ in ‘magnetic.f90’, declare and calculate the quantity $jb$ (the average of which will be $jbm$), and call $\text{sum}_\text{mn}_\text{name}$

   ```
   real, dimension (nx) :: jb  \ assoc { jj \cdot BB
   [...] if (ldiagnos) then ! only calculate if diagnostics is required
   if (idiag_jbm/=0) then ! anybody asked for $jbm$?
   call dot_mn(jj,bb,jb) ! assuming $jj$ and $bb$ are known
   call sum_mn_name(jb,i_jbm)
   endif
   endif
   ```
3. in the subroutine \texttt{rprint\_magnetic} in both \texttt{magnetic.f90}, add the following:

\begin{verbatim}
!  reset everything in case of RELOAD
!  (this needs to be consistent with what is defined above!)
!
if (lreset) then  ! need to reset list of diagnostic variables?
  [...]
  idiaj_jbm=0
  [...]
endif
!
! check for those quantities that we want to evaluate online
!
do iname=1,nname
  [...]
  call parse_name(iname,cname(iname),cform(iname),'jbm',idiag_jbm)
  [...]
enddo
![...]
!
! write column, \texttt{i\_XYZ}, where our variable \texttt{XYZ} is stored
!
... write(3,*) 'i\_jbm=',idiag_jbm
![...]
\end{verbatim}

4. in the subroutine \texttt{rprint\_magnetic} in \texttt{nomagnetic.f90}, add the following (newer versions of the code may not require this any more):

\begin{verbatim}
!  write column, \texttt{i\_jbm}, where our variable \texttt{jbm} is stored
!  idl needs this even if everything is zero
!
... write(3,*) 'i\_jbm=',idiag_jbm
![...]
\end{verbatim}

5. and don't forget to add your new variable to \texttt{print.in}:

\begin{verbatim}
  jbm(f10.5)
\end{verbatim}

If, instead of a mean value, you want a new maximum quantity, you need to replace \texttt{sum\_mn\_name()} by \texttt{max\_mn\_name()}.

Sect.\ref{sec:output-avg} describes how to output horizontal averages of the magnetic and velocity fields. New such averages can be added to the code by using the existing averaging procedures \texttt{calc\_bmz()} or \texttt{calc\_jmz()} as examples.
9.3 The f-array

The ‘f’ array is the largest array in the PENCIL CODE and its primary role is to store the current state of the timestepped PDE variables. The f-array and its slightly smaller counter part (the df-array; see below) are the only full size 3D arrays in the code. The f-array is of type real but PDEs for a complex variable may be solved by using two slots in the f-array. The actual size of the f-array is \( mx \times my \times mz \times mfarray \). Here, \( mfarray = mvar + maux + mglobal + mscratch \) where \( mvar \) refers to the number of real PDE variables.

As an example, we describe here how to put the time-integrated velocity, \( uut \), into the f-array (see ‘hydro.f90’). If this is to be invoked, there must be the following call somewhere in the code:

```fortran
  call farray_register_auxiliary('uut',iuut,vector=3)
```

Here, \( iuut \) is the index of the variable \( uut \) in the f-array. Of course, this requires that \( maux \) is increased by 3, but in order to do this for a particular run only one must write a corresponding entry in the ‘cparam.local’ file,

```fortran
  ! cparam.local
  ! -*-f90-*- (for Emacs)
  !
  !** AUTOMATIC CPARAM.INC GENERATION ****************************************
  ! Declare (for generation of cpnam.inc) the number of f array
  ! variables and auxiliary variables added by this module
  !
  ! MAUX CONTRIBUTION 3
  !
  !***************************************************************************
  ! Local settings concerning grid size and number of CPUs.
  ! This file is included by cpnam.f90
  !
  integer, parameter :: ncpus=1,nprocy=1,nprocz=ncpus/nprocy,nprocx=1
  integer, parameter :: nxgrid=16,nygrid=nxgrid,nzgrid=nxgrid
```

This way such a change does not affect the memory usage for other applications where this addition to ‘cparam.local’ is not made. In order to output this part of the f-array, one must write \( lwrite_aux=T \) in the init_pars of ‘start.in’. (Technically, \( lwrite_aux=T \) can also be invoked in run_pars of ‘run.in’, but this does not work at the moment.)

9.4 The df-array

The ‘df’ array is the second largest chunk of data in the PENCIL CODE. By using a 2N storage scheme (see H.4) after Williamson [29] the code only needs one more storage area for each timestepped variable on top of the current state stored in the f-array. As such, and in contrast to the f-array, the df-array is of size \( mx \times my \times mz \times mvar \). Like the df-array it is of type real. In fact the ghost zones of df are not required or calculated but having f- and df-arrays of the same size make the coding more transparent. For \( mx \), \( my \) and \( mz \) large the wasted storage becomes negligible.
9.5 The fp-array

Similar to the ‘f’ array the code also has a ‘fp’ array which contains current states of all the particles. Like the f-array the fp-array also has a time derivative part, the dfp-array. The dimension of the fp-array is \( mpar_{local} \times mpar \) where \( mpar_{local} \) is the number of particles in the local processor (for serial runs this is the total number of particles) and \( mpar \) depends on the problem at hand. For example if we are solving for only tracer particles then \( mpar = 3 \), for dust particles \( mpar = 6 \). The sequence in which the slots in the fp-array are filled up depends on the sequence in which different particle modules are called from the particles_main.f90. The following are the relevant lines from particles_main.f90.

```fortran
!**********************************************************************
subroutine particles_register_modules()
!
! Register particle modules.
!
! 07-jan-05/anders: coded
!
    call register_particles     ()
    call register_particles_radius ()
    call register_particles_spin   ()
    call register_particles_number ()
    call register_particles_mass    ()
    call register_particles_selfgrav ()
    call register_particles_nbody   ()
    call register_particles_viscosity ()
    call register_pars_diagnos_state ()
!
endsubroutine particles_register_modules
!**********************************************************************
```

The subroutine register_particles can mean either the tracer particles or dust particles. For the former the first three slots of the fp-array are the three spatial coordinates. For the latter the first six slots of the fp-array are the three spatial coordinates followed by the three velocity components. The seventh slot (or the fourth if we are use tracer particles) is the radius of the particle which can also change as a function of time as particles collide and fuse together to form bigger particles.

9.6 The pencil case

Variables that are derived from the basic physical variables of the code are stored in one-dimensional pencils of length \( nx \). All the pencils that are defined for a given set of physics modules are in turn bundled up in a Fortran structure called \( p \) (or, more illustrative, the pencil case). Access to individual pencils happens through the variable \( p\%name \), where \( name \) is the name of a pencil, e.g. \( rho \) that is a derived variable of the logarithmic density \( \ln rho \).

The pencils provided by a given physics module are declared in the header of the file, e.g. in the Density module:
PENCILS PROVIDED lnrho; rho; rho1; glnrho(3); grho(3); uglnrho; ugrho

Notice that the pencil names are separated with a semi-colon and that vector pencils are declared with “(3)” after the name. Before compiling the code, the script ‘mkcparam’ collects the names of all pencils that are provided by the chosen physics modules. It then defines the structure p with slots for every single of these pencils. The definition of the pencil case p is written in the include file ‘cparam_pencils.inc’. When the code is run, the actual pencils that are needed for the run are chosen based on the input parameters. This is done in the subroutines pencil_criteria_modulename that are present in each physics module. They are all called once before entering the time loop. In the pencil_criteria subroutines the logical arrays lpenc_requested, lpenc_diagnos, lpenc_diagnos2d, and lpenc_video are set according to the pencils that are needed for the given run. Some pencils depend on each other, e.g. uglnrho depends on uu and glnrho. Such interdependencies are sorted out in the subroutines pencil_interdep_modulename that are called after pencil_criteria_modulename.

In each time-step the values of the pencil logicals lpenc_requested, lpenc_diagnos, lpenc_diagnos2d, and lpenc_video are combined to one single pencil array lpencil which is different from time-step to time-step depending on e.g. whether diagnostics or video output are done in that time-step. The pencils are then calculated in the subroutines calc_pencils_modulename. This is done before calculating the time evolution of the physical variables, as this depends very often on derived variables in pencils.

The centralized pencil calculation scheme is a guarantee that

- All pencils are only calculated once
- Pencils are always calculated by the proper physics module

Since the PENCIL CODE is a multipurpose code that has many different physics modules, it can lead to big problems if a module tries to calculate a derived variable that actually belongs to another module, because different input parameters can influence how the derived variables are calculated. One example is that the Density module can consider both logarithmic and non-logarithmic density, so if the Magnetic module calculates

$$\rho = \exp(f(l1:l2,m,n,lnrho))$$

it is wrong if the Density module works with non-logarithmic density! The proper way for the Magnetic module to get to know the density is to request the pencil rho in pencil_criteria_magnetic.

9.6.1 Pencil check

To check that the correct pencils have been requested for a given run, one can run a pencil consistency check in the beginning of a run by setting the logical lpencil_check in &run_pars. The check is meant to see if

- All needed pencils have been requested
- All requested pencils are needed

The consistency check first calculates the value of df with all the requested pencils. Then the pencil requests are flipped one at a time – requested to not requested, not requested to requested. The following combination of events can occur:
• not requested $\rightarrow$ requested, $\partial f$ not changed  
The pencil is not requested and is not needed.

• not requested $\rightarrow$ requested, $\partial f$ changed  
The pencil is not requested, but is needed. The code stops.

• requested $\rightarrow$ not requested, $\partial f$ not changed  
The pencil is requested, but is not needed. The code gives a warning.

• requested $\rightarrow$ not requested, $\partial f$ changed  
The pencil is requested and is needed.

9.6.2 Adding new pencils

Adding a new pencil to the pencil case is trivial but requires a few steps.

• Declare the name of the pencil in the header of the proper physics module. Pencil names must appear come in a ";" separated list, with dimensions in parenthesis after the name [(3) for vector, (3,3) for matrix, etc.].

• Set interdependency of the new pencil (i.e. what other pencils does it depend on) in the subroutine pencil_interdep_modulename

• Make rule for calculating the pencil in calc_pencils_modulename

• Request the new pencil based on the input parameters in any relevant physics module

Remember that the centralized pencilation scheme is partially there to force the users of the code to think in general terms when implementing new physics. Any derived variable can be useful for a number of different physics problems, and it is important that a pencil is accessible in a transparent way to all modules.

9.7 Adding new physics: the Special module

If you want to add new physics to the code, you will in many cases want to add a new Special module. Doing so is relatively straightforward and there is even a special directory for such additions.

To create your own special module, copy ‘nospecial.f90’ from the src/ directory to a new name in the src/special/ directory. It is currently only possible to have one special modules at a time and so several new bits of physics are often put in to one special module. For this reasons a name should be chosen that relates to the problem to be solved rather than the specific physics being implemented.

The first thing to do in your new module is to change the lspecial=.false. header to say lspecial=.true.

The file is heavily commented though all such comments can be removed as you go. You may implement any of the subroutines/function that exist in nospecial.f90 and those routines must have the names and parameters as in nospecial.f90. You do not however need to implement all routines, and you may either leave the dummy routines
copied from nospecial.f90 or delete them all together (provided the "include 'special-
dummy.inc" is kept intact at the end of the file. Beyond that, and data and subroutines
can be added to a special module as required, though only for use within that module.

There are routines in the special interface to allow you to add new equations, modify the
existing equation, add diagnostics, add slices, and many more things. If you feel there is
something missing extra hooks can easily be added - please contact the PENCIL CODE
team for assistance.

You are encouraged to submit/commit your special modules to the Pencil Code source.
When you have added new stuff to the code, don’t forget to mention this in the

9.8 Adding switchable modules

In some cases where a piece of physics is thought to be more fundamental, useful in
many situations or simply more flexibility is required it may be necessary to add a new
module newphysics together with the corresponding nonewphysics module. The special
modules follow the same structure as the rest of the switchable modules and so using a
special module to prototype new ideas can make writing a new switchable module much
easier.

For an example of module involving a new variable (and PDE), the pscalar module is a
good prototype. The grep command

  unix> grep -i pscalar src/*

gives you a good overview of which files you need to edit or add.

9.9 Adding your initial conditions: the InitialCondition module

Although the code has many initial conditions implemented, we now discourage such
practice. We aim to eventually removed most of them. The recommended course of action
is to make use of the InitialCondition module.

InitialCondition works pretty much like the Special module. To implement your own
custom initial conditions, copy the file ‘noinitial_condition.f90’ from the src/ to src/
initial_condition, with a new, descriptive, name.

The first thing to do in your new module is to change the linitialcondition=.false. header
to say linitialcondition=.true. Also, don’t forget to add ../ in front of the file names in
include statements.

This file has hooks to implement a custom initial condition to most vari-
ables. After implementing your initial condition, add the line INITIAL-_CONDITION=initial_condition/myinitialcondition to your ‘src/Makefile.local’
file. Here, myinitialcondition is the name you gave to your initial condition file. Add
also initial_condition_pars to the ‘start.in’ file, just below init_pars. This is a
namelist, which you can use to add whichever quantity your initial condition needs
defined, or passed. You must also un-comment the relevant lines in the subroutines
for reading and writing the namelists. For compiling reasons, these subroutines in
‘noinitial_condition.f90’ are dummies. The lines are easily identifiable in the code.
Check e.g. the samples ‘2d-tests/baroclinic’, ‘2d-tests/spherical_viscous_ring’, or ‘interlocked-fluxrings’, for examples of how the module is used.
10 Testing the code

To maintain reproducibility despite sometimes quite rapid development, the PENCIL CODE is tested nightly on various architectures. The front end for testing are the scripts pc_auto-test and (possibly) pencil-test.

To see which samples would be tested, run

```
unix> pc_auto-test -l
```

, to actually run the tests, use

```
unix> pc_auto-test
```
or

```
unix> pc_auto-test --clean
```

. The latter compiles every test sample from scratch and currently (September 2009) takes about 2 hours on a mid-end Linux PC.

The pencil-test script is useful for cron jobs and allows the actual test to run on a remote computer. See Sect. [10.1] below.

For a complete list of options, run `pc_auto-test --help` and/or `pencil-test --help`.

10.1 How to set up periodic tests

To set up a nightly test of the PENCIL CODE, carry out the following steps.

1. Identify a host for running the actual tests (the work host) and one to initiate the tests and collect the results (the scheduling host). On the scheduling host, you should be able to

   (a) run cron jobs,

   (b) ssh to the work host without password,

   (c) publish HTML files (optional, but recommended),

   (d) send e-mail (optional, but recommended).

   Work host and scheduling host can be the same (in this case, use pencil-test's `-l` option, see below), but often they will be two different computers.

2. [Recommended, but optional:] On the work host, check out a separate copy of the PENCIL CODE to reduce the risk that you start coding in the auto-test tree. In the following, we will assume that you checked out the code as `~/pencil-auto-test`.

3. On the work host, make sure that the code finds the correct configuration file for the tests you want to carry out. [Elaborate on that: PENCIL_HOME/local_config and `-f` option; give explicit example]

   Remember that you can set up a custom host ID file for your auto-test tree under `${PENCIL_HOME}/config-local/hosts/`.

4. On the scheduling host, use `crontab -e` to set up a cron job similar to the following:
30 02 * * * $HOME/pencil-auto-test/bin/pencil-test \
   -D $HOME/pencil-auto-test \n   --use-pc_auto-test \n   -N 15 -Uc -rs \n   -T $HOME/public_html/pencil-code/tests/timings.txt \n   -t 15m \n   -m <email1@inter.net,email2@inter.net,...> \n   <work-host.inter.net> \n   -H > $HOME/public_html/pencil-code/tests/nightly-tests.html

**Note 1:** This has to be one long line. The backslash characters are written only for formatting purposes for this manual *you cannot use them in a crontab file.*

**Note 2:** You will have to adapt some parameters listed here and may want to modify a few more:

- `-D <dir>`: Sets the directory (on the work host) to run in.
- `-T <file>`: If this option is given, append a timing statistics line for each test to the given file.
- `--use-pc`: You want this option (and at some point, it will be the default).
- `-t 15m`: Limit the time for ‘start.x’ and ‘run.x’ to 15 minutes.
- `-N 15`: Run the tests at nice level 15 (may not have an effect for MPI tests).
- `-Uc`: Do svn update and pc_build --cleanall before compiling.
- `-m <email-list>`: If this option is given, send e-mails to everybody in the (comma-separated) list of e-mail addresses if any test fails. As soon as this option is set, the maintainers (as specified in the ‘README’ file) of failed tests will also receive an e-mail.

- `work-host.inter.net|-l`: Replace this with the remote host that is to run the tests. If you want to run locally, write `-l` instead.
- `-H`: Output HTML.

If you want to run fewer or more tests, you can use the ‘-Wa,--max-level’ option:

- `Wa,--max-level=3`

will run all tests up to (and including) level 3. The default corresponds to ‘-Wa,--max-level=2’.

For a complete listing of pencil-test options, run

`unix> pencil-test --help`
11 Useful internals

11.1 Global variables

The following variables are defined in ‘cdata.f90’ and are available in any routine that uses the module Cdata.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>real</td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>simulated time $t$.</td>
</tr>
<tr>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>$n[xyz]grid$</td>
<td>global number of grid points (excluding ghost cells) in $x, y$ and $z$ direction.</td>
</tr>
<tr>
<td>$nx, ny, nz$</td>
<td>number of grid points (excluding ghost cells) as seen by the current processor, i.e. $ny=nygrid/nprocy$, etc.</td>
</tr>
<tr>
<td>$mx, my, mz$</td>
<td>number of grid points seen by the current processor, but including ghost cells. Thus, the total box for the $ivar$th variable (on the given processor) is given by $f(1:mx,1:my,1:mz,ivar)$.</td>
</tr>
<tr>
<td>$l1, l2$</td>
<td>smallest and largest $x$-index for the physical domain (i.e. excluding ghost cells) on the given processor.</td>
</tr>
<tr>
<td>$m1, m2$</td>
<td>smallest and largest $y$-index for physical domain.</td>
</tr>
<tr>
<td>$n1, n2$</td>
<td>smallest and largest $z$-index for physical domain, i.e. the physical part of the $ivar$th variable is given by $f(l1:l2,m1:m2,n1:n2,ivar)$.</td>
</tr>
<tr>
<td>$m, n$</td>
<td>pencil indexing variables: During each time-substep the box is traversed in $x$-pencils of length $mx$ such that the current pencil of the $ivar$th variable is $f(l1:l2,m,n,ivar)$.</td>
</tr>
<tr>
<td>logical</td>
<td></td>
</tr>
<tr>
<td>$lroot$</td>
<td>true only for MPI root processor.</td>
</tr>
<tr>
<td>$lfirst$</td>
<td>true only during first time-substep of each time step.</td>
</tr>
<tr>
<td>$headt$</td>
<td>true only for very first full time step (comprising 3 substeps for the 3rd-order Runge–Kutta scheme) on root processor.</td>
</tr>
<tr>
<td>$headtt$</td>
<td>$= (lfirst .and. lroot)$: true only during very first time-substep on root processor.</td>
</tr>
<tr>
<td>$lfirstpoint$</td>
<td>true only when the very first pencil for a given time-substep is processed, i.e. for the first set of $(m,n)$, which is probably $(3,3)$.</td>
</tr>
<tr>
<td>$lout$</td>
<td>true when diagnostic output is about to be written.</td>
</tr>
</tbody>
</table>

11.2 Subroutines and functions

*output(file,a,nv)* (module *IO*): Write (in each ‘procN’ directory) the content of the global array $a$ to a file called *file*, where $a$ has dimensions $mx \times my \times mz \times nv$, or
$mx \times my \times mz$ if $nv=1$.

`output_pencil(file,a,nv)` (module `IO`): Same as `output()`, but for a pencil variable, i.e. an auxiliary variable that only ever exists on a pencil (e.g. the magnetic field strength $bb$ in ‘magnetic.f90’, or the squared sound speed $cs2$ in ‘entropy.f90’). The file has the same structure as those written by `output()`, because the values of $a$ on the different pencils are accumulated in the file. This involves a quite non-trivial access pattern to the file and has thus been coded in C (‘src/debug_c.c’).

`cross(a,b,c)` (module `Sub`): Calculate the cross product of two vectors $a$ and $b$ and store in $c$. The vectors must either all be of size $mx \times my \times mz \times 3$ (global arrays), or of size $nx \times 3$ (pencil arrays).

`dot(a,b,c)` (module `Sub`): Calculate the dot product of two vectors $a$ and $b$ and store in $c$. The vectors must either be of size $mx \times my \times mz \times 3$ ($a$ and $b$) and $mx \times my \times mz$ ($c$), or of size $nx \times 3$ ($a$ and $b$) and $nx$ ($c$).

`dot2(a,c)` (module `Sub`): Same as `dot(a,a,c)`.
Part III

Appendix

APPENDIX

Date, Revision

A Timings

In the following table we list the results of timings of the code on different machines. Shown is (among other quantities) the wall clock time per mesh point (excluding the ghost zones) and per full 3-stage time step, a quantity that is printed by the code at the end of a run.

As these results were assembled during the development phase of the code (that hasn’t really finished yet,...), you may not get the same numbers, but they should give some orientation of what to expect for your specific application on your specific hardware.

The code will output the timing (in microseconds per grid point per time-step) at the end of a run. You can also specify `walltime` in `print.in` to have the code continuously output the physical time it took to reach the time-steps where diagnostics is done. The time-dependent code speed can then be calculated by differentiating, e.g. in IDL with

```idl
IDL> pc_read_ts, obj=ts
IDL> plot, ts.it, 1/nw*deriv(ts.it,ts.walltime/1.0e-6), psym=2
where nw=nx*ny*nz.
```

<table>
<thead>
<tr>
<th>proc</th>
<th>machine</th>
<th>µs per pt</th>
<th>resol.</th>
<th>what</th>
<th>mem/proc</th>
<th>when</th>
<th>who</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NL3</td>
<td>19</td>
<td>64³</td>
<td>kinematic</td>
<td>10 MB</td>
<td>20-may-02</td>
<td>AB</td>
</tr>
<tr>
<td>1</td>
<td>NL3</td>
<td>30</td>
<td>64³</td>
<td>magn/noentro</td>
<td>20 MB</td>
<td>20-may-02</td>
<td>AB</td>
</tr>
<tr>
<td>1</td>
<td>Nq1</td>
<td>10</td>
<td>64³</td>
<td>magn/noentro</td>
<td>30-may-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Ukaff</td>
<td>9.2</td>
<td>64³</td>
<td>magn/noentro</td>
<td>20-may-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NL6</td>
<td>6.8</td>
<td>64³</td>
<td>magn/noentro</td>
<td>10-mar-03</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NL6</td>
<td>36.3</td>
<td>64x128x64</td>
<td>nomag/entro/dust</td>
<td>19-sep-03</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NL6</td>
<td>42.7</td>
<td>16³x256</td>
<td>nomag/entro/rad6/ion</td>
<td>22-oct-03</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NL6</td>
<td>37.6</td>
<td>16³x256</td>
<td>nomag/entro/rad2/ion</td>
<td>22-oct-03</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NL6</td>
<td>19.6</td>
<td>16³x256</td>
<td>nomag/entro/ion</td>
<td>22-oct-03</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NL6</td>
<td>8.7</td>
<td>16³x256</td>
<td>nomag/entro</td>
<td>22-oct-03</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NL6n</td>
<td>9.8</td>
<td>32³</td>
<td>magn/noentro/pscalar</td>
<td>17-mar-06</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Mhd</td>
<td>7.8</td>
<td>64³</td>
<td>magn/noentro</td>
<td>20-may-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Nq4</td>
<td>14.4</td>
<td>128³</td>
<td>magn/noentro</td>
<td>8-oct-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Nq5</td>
<td>6.7</td>
<td>128³</td>
<td>magn/noentro</td>
<td>8-oct-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>fe1</td>
<td>5.1</td>
<td>128³</td>
<td>magn/noentro</td>
<td>9-oct-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Kabul</td>
<td>4.4</td>
<td>128³</td>
<td>magn/noentro</td>
<td>130 MB</td>
<td>20-jun-02</td>
<td>WD</td>
</tr>
<tr>
<td>1</td>
<td>Hwwax5</td>
<td>3.4</td>
<td>256³</td>
<td>convstar</td>
<td>7.8 GB</td>
<td>29-jan-03</td>
<td>WD</td>
</tr>
<tr>
<td>1</td>
<td>Mac/g95</td>
<td>7.7</td>
<td>32³</td>
<td>magn/noentro</td>
<td>14-jan-07</td>
<td>BD</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Mac/ific</td>
<td>4.5</td>
<td>32³</td>
<td>magn/noentro</td>
<td>14-jan-07</td>
<td>BD</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Kabul</td>
<td>2.5</td>
<td>128³</td>
<td>magn/noentro</td>
<td>80 MB</td>
<td>20-jun-02</td>
<td>WD</td>
</tr>
<tr>
<td>2</td>
<td>Nq4+4</td>
<td>7.4</td>
<td>128³</td>
<td>magn/noentro</td>
<td>8-oct-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Nq4+4</td>
<td>8.9</td>
<td>128³</td>
<td>magn/noentro</td>
<td>8-oct-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Nq4+5</td>
<td>7.3</td>
<td>128³</td>
<td>magn/noentro</td>
<td>8-oct-02</td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Nq5+5</td>
<td>3.7</td>
<td>128³</td>
<td>magn/noentro</td>
<td>8-oct-02</td>
<td>AB</td>
<td></td>
</tr>
</tbody>
</table>

Note that when using 'nompicomm.f90', the timer currently used will overflow on some machines, so you should not blindly trust the timings given by the code.
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2 fe1</td>
<td>3.45</td>
<td>128³ magn/noentro</td>
<td>9-oct-02</td>
<td>AB</td>
</tr>
<tr>
<td>2 Nq2</td>
<td>9.3</td>
<td>64³ magn/noentro</td>
<td>11-sep-02</td>
<td>AB</td>
</tr>
<tr>
<td>2 Nq1+2</td>
<td>8.3</td>
<td>64³ magn/noentro</td>
<td>11-sep-02</td>
<td>AB</td>
</tr>
<tr>
<td>2 Hwwsx5</td>
<td>1.8</td>
<td>256³ convstar</td>
<td>7.9 GB</td>
<td>29-jan-03</td>
</tr>
<tr>
<td>4 Nq1+2</td>
<td>5.4</td>
<td>64³ magn/noentro</td>
<td>11-sep-02</td>
<td>AB</td>
</tr>
<tr>
<td>4 Nq1235</td>
<td>4.1</td>
<td>128³ magn/noentro</td>
<td>11-sep-02</td>
<td>AB</td>
</tr>
<tr>
<td>4 Nq0-3</td>
<td>6.8</td>
<td>256³ magn/noentro</td>
<td>294 MB</td>
<td>10-jun-02</td>
</tr>
<tr>
<td>4 Mhd</td>
<td>2.76</td>
<td>64³ magn/noentro</td>
<td>30-may-02</td>
<td>AB</td>
</tr>
<tr>
<td>4 fe1</td>
<td>3.39</td>
<td>32³ magn/noentro</td>
<td>16-aug-02</td>
<td>AB</td>
</tr>
<tr>
<td>4 Rasm.</td>
<td>2.02</td>
<td>64³ magn/noentro</td>
<td>2x2</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>4 Mhd</td>
<td>8.2</td>
<td>64²×16 nomag/entro</td>
<td>23-jul-02</td>
<td>AB</td>
</tr>
<tr>
<td>4 fe1</td>
<td>6.35</td>
<td>64Ⅹ128Ⅹ64 nomag/entro/dust</td>
<td>19-sep-03</td>
<td>AB</td>
</tr>
<tr>
<td>4 fe1</td>
<td>2.09</td>
<td>128³ magn/noentro</td>
<td>9-oct-02</td>
<td>AB</td>
</tr>
<tr>
<td>4 fe1</td>
<td>1.45</td>
<td>128³ magn/noentro</td>
<td>giga</td>
<td>9-oct-02</td>
</tr>
<tr>
<td>4 fe1</td>
<td>7.55</td>
<td>16²×512 nomag/entro/rad2/ion</td>
<td>4Ⅹ1</td>
<td>1-nov-03</td>
</tr>
<tr>
<td>4 fe1</td>
<td>5.48</td>
<td>16²×512 nomag/entro/rad2/ion</td>
<td>1Ⅹ4</td>
<td>1-nov-03</td>
</tr>
<tr>
<td>4 Luci</td>
<td>1.77</td>
<td>64³ magn/noentro</td>
<td>27-feb-07</td>
<td>AB</td>
</tr>
<tr>
<td>4 Lenn</td>
<td>0.65</td>
<td>64³ nomag/noentro</td>
<td>13-jan-07</td>
<td>AB</td>
</tr>
<tr>
<td>4 Lenn</td>
<td>1.21</td>
<td>64³ magn/noentro</td>
<td>7-nov-06</td>
<td>AB</td>
</tr>
<tr>
<td>4 Kabul</td>
<td>1.5</td>
<td>128³ magn/noentro</td>
<td>47 MB</td>
<td>20-jun-02</td>
</tr>
<tr>
<td>4 Hwwsx5</td>
<td>1.8</td>
<td>256³ convstar</td>
<td>8.2 GB</td>
<td>29-jan-03</td>
</tr>
<tr>
<td>8 Nqall</td>
<td>3.0</td>
<td>128³ magn/noentro</td>
<td>8-oct-02</td>
<td>AB</td>
</tr>
<tr>
<td>8 fe1</td>
<td>3.15</td>
<td>64³ magn/noentro</td>
<td>1Ⅹ8</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>8 fe1</td>
<td>2.36</td>
<td>64³ magn/noentro</td>
<td>2Ⅹ4</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>8 Ukaff</td>
<td>1.24</td>
<td>64³ magn/noentro</td>
<td>20-may-02</td>
<td>AB</td>
</tr>
<tr>
<td>8 Kabul</td>
<td>1.25</td>
<td>64Ⅹ128 nomag/entro</td>
<td>11-jul-02</td>
<td>WD</td>
</tr>
<tr>
<td>8 fe1</td>
<td>1.68</td>
<td>128³ magn/noentro</td>
<td>1Ⅹ8</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>8 fe1</td>
<td>1.50</td>
<td>128³ magn/noentro</td>
<td>2Ⅹ4</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>8 fe1</td>
<td>1.44</td>
<td>128³ magn/noentro</td>
<td>4Ⅹ2</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>8 Kabul</td>
<td>0.83</td>
<td>128³ magn/noentro</td>
<td>28 MB</td>
<td>20-jun-02</td>
</tr>
<tr>
<td>8 Gridur</td>
<td>1.46</td>
<td>128³ magn/noentro</td>
<td>19-aug-02</td>
<td>NE</td>
</tr>
<tr>
<td>8 Kabul</td>
<td>0.87</td>
<td>256³ magn/noentro</td>
<td>160 MB</td>
<td>20-jun-02</td>
</tr>
<tr>
<td>8 fe1</td>
<td>0.99</td>
<td>256³ magn/noentro</td>
<td>2Ⅹ4</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>8 fe1</td>
<td>0.98</td>
<td>256³ magn/noentro</td>
<td>4Ⅹ2</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>8 cetus</td>
<td>0.58</td>
<td>64³ magn/noentro</td>
<td>4Ⅹ2</td>
<td>19-aug-07</td>
</tr>
<tr>
<td>8 cetus</td>
<td>0.73</td>
<td>256³ magn/noentro</td>
<td>4Ⅹ2,156M</td>
<td>19-aug-07</td>
</tr>
<tr>
<td>8 Neolith</td>
<td>0.82</td>
<td>64³ magn/noentro</td>
<td>4Ⅹ2</td>
<td>19-aug-07</td>
</tr>
<tr>
<td>8 Mhd</td>
<td>1.46</td>
<td>160³×40 nomag/entro</td>
<td>46 MB</td>
<td>7-oct-02</td>
</tr>
<tr>
<td>8 Hwwsx5</td>
<td>0.50</td>
<td>256³ convstar</td>
<td>8.6 GB</td>
<td>29-jan-03</td>
</tr>
<tr>
<td>8 Neolith</td>
<td>0.444</td>
<td>128³ magn/noentro</td>
<td>6-dec-07</td>
<td>AB</td>
</tr>
<tr>
<td>8 Ferlin</td>
<td>0.450</td>
<td>64³ ttest/noentro</td>
<td>21-jun-09</td>
<td>AB</td>
</tr>
<tr>
<td>8 Ferlin</td>
<td>0.269</td>
<td>64³ magn/noentro</td>
<td>2-apr-10</td>
<td>AB</td>
</tr>
<tr>
<td>8 Ferlin</td>
<td>0.245</td>
<td>128³ magn/noentro</td>
<td>2-feb-11</td>
<td>AB</td>
</tr>
<tr>
<td>8 nor52</td>
<td>2.00</td>
<td>32³ magn/noentro</td>
<td>2-dec-09</td>
<td>AB</td>
</tr>
<tr>
<td>9 hydra(2)</td>
<td>0.317</td>
<td>72³ magn/noentro</td>
<td>1Ⅹ3Ⅹ3</td>
<td>8-may-16</td>
</tr>
<tr>
<td>9 charybdis</td>
<td>0.169</td>
<td>72³ magn/noentro</td>
<td>1Ⅹ3Ⅹ3</td>
<td>8-may-16</td>
</tr>
<tr>
<td>9 scylla</td>
<td>0.150</td>
<td>72³ magn/noentro</td>
<td>1Ⅹ3Ⅹ3</td>
<td>8-may-16</td>
</tr>
<tr>
<td>12 scylla</td>
<td>0.151</td>
<td>72³ magn/noentro</td>
<td>1Ⅹ3Ⅹ3</td>
<td>8-may-16</td>
</tr>
<tr>
<td>12 janus</td>
<td>6.02</td>
<td>72²×22 coag/noentro</td>
<td>17-dec-15</td>
<td>AB</td>
</tr>
<tr>
<td>16 fe1</td>
<td>1.77</td>
<td>64³ convstar</td>
<td>9-feb-03</td>
<td>AB</td>
</tr>
<tr>
<td>16 copson</td>
<td>0.596</td>
<td>128³ geodynamo/ks95</td>
<td>21-nov-03</td>
<td>DM</td>
</tr>
<tr>
<td>16 fe1</td>
<td>0.94</td>
<td>128³ magn/noentro</td>
<td>4Ⅹ4</td>
<td>8-sep-02</td>
</tr>
<tr>
<td>16 fe1</td>
<td>0.75</td>
<td>128³ magn/noentro</td>
<td>4Ⅹ4/6Ⅹ6</td>
<td>9-may-03</td>
</tr>
<tr>
<td>16 workq</td>
<td>0.88</td>
<td>128³ magn/noentro</td>
<td>4Ⅹ4/6Ⅹ6</td>
<td>21-aug-04</td>
</tr>
<tr>
<td>16 giga</td>
<td>0.76</td>
<td>128³ magn/noentro</td>
<td>4Ⅹ4/6Ⅹ6</td>
<td>21-aug-04</td>
</tr>
<tr>
<td>16 giga2</td>
<td>0.39</td>
<td>128³ magn/noentro</td>
<td>4Ⅹ4/6Ⅹ6</td>
<td>20-aug-04</td>
</tr>
<tr>
<td>16 giga</td>
<td>0.47</td>
<td>128³ chiral</td>
<td>4Ⅹ4/6Ⅹ6</td>
<td>29-may-04</td>
</tr>
<tr>
<td>16 giga</td>
<td>0.43</td>
<td>128³ nomag/noentro</td>
<td>4Ⅹ4/6Ⅹ6</td>
<td>28-apr-03</td>
</tr>
<tr>
<td>16 Mhd</td>
<td>2.03</td>
<td>128³ magn/noentro</td>
<td>26-nov-02</td>
<td>AB</td>
</tr>
<tr>
<td>16 Mhd</td>
<td>0.64</td>
<td>256³ magn/noentro</td>
<td>60 MB</td>
<td>22-may-02</td>
</tr>
<tr>
<td></td>
<td>Lat.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>16</td>
<td>fe1</td>
<td>0.56</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>fe1</td>
<td>6.30</td>
<td>128×256×128</td>
<td>nomag/entro/dust</td>
</tr>
<tr>
<td>16</td>
<td>fe1</td>
<td>1.31</td>
<td>128³×512</td>
<td>nomag/entro/rad2/ion</td>
</tr>
<tr>
<td>16</td>
<td>Ukaff</td>
<td>0.61</td>
<td>128³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Ukaff</td>
<td>0.64</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Kabul</td>
<td>0.80</td>
<td>128³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Kabul</td>
<td>0.51</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Gridur</td>
<td>0.81</td>
<td>128³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Gridur</td>
<td>0.66</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Sander</td>
<td>0.53</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Luci</td>
<td>0.375</td>
<td>128³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Lenn</td>
<td>0.284</td>
<td>128³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Neolith</td>
<td>0.180</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Triolith</td>
<td>0.075</td>
<td>128³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Triolith</td>
<td>0.065</td>
<td>128³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Triolith</td>
<td>0.054</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>16</td>
<td>Coma</td>
<td>0.603</td>
<td>128³</td>
<td>GW/magn/noentro</td>
</tr>
<tr>
<td>24</td>
<td>Gardar</td>
<td>0.44</td>
<td>128² × 48</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>24</td>
<td>Summit</td>
<td>0.041</td>
<td>144³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>32</td>
<td>giga?</td>
<td>0.32</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>32</td>
<td>Ukaff</td>
<td>0.34</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>32</td>
<td>Ukaff</td>
<td>0.32</td>
<td>512³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>32</td>
<td>Hermit</td>
<td>0.200</td>
<td>256×512×256</td>
<td>spherical conv/magn</td>
</tr>
<tr>
<td>32</td>
<td>fe1</td>
<td>0.168</td>
<td>512³</td>
<td>nomag/noentro</td>
</tr>
<tr>
<td>32</td>
<td>fe1</td>
<td>1.26</td>
<td>64²×256</td>
<td>nomag/entro/rad/ion</td>
</tr>
<tr>
<td>32</td>
<td>Luci</td>
<td>0.182</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>32</td>
<td>Lenn</td>
<td>0.147</td>
<td>256³</td>
<td>nomag/entro/cool/fo</td>
</tr>
<tr>
<td>32</td>
<td>Steno</td>
<td>0.076</td>
<td>256³</td>
<td>nomag/entro/cool/fo</td>
</tr>
<tr>
<td>32</td>
<td>Steno</td>
<td>0.081</td>
<td>256³</td>
<td>nomag/entro/cool</td>
</tr>
<tr>
<td>32</td>
<td>Steno</td>
<td>0.085</td>
<td>256³</td>
<td>nomag/entro/cool/sh</td>
</tr>
<tr>
<td>32</td>
<td>Steno</td>
<td>0.235</td>
<td>512²×256</td>
<td>mag/entro</td>
</tr>
<tr>
<td>32</td>
<td>Sanss</td>
<td>0.273</td>
<td>128×256²</td>
<td>nomag</td>
</tr>
<tr>
<td>32</td>
<td>Neolith</td>
<td>0.275</td>
<td>128³</td>
<td>testfield4</td>
</tr>
<tr>
<td>32</td>
<td>Ferlin</td>
<td>0.556</td>
<td>128³</td>
<td>testscalar</td>
</tr>
<tr>
<td>36</td>
<td>Kraken</td>
<td>0.177</td>
<td>192×384×64</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>36</td>
<td>scylla</td>
<td>0.096</td>
<td>72³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>48</td>
<td>janus</td>
<td>0.028</td>
<td>72² + 216</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Coma</td>
<td>0.573</td>
<td>128³</td>
<td>GW/magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>fe1</td>
<td>0.24</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>giga</td>
<td>0.11</td>
<td>256³</td>
<td>nomag/noentro</td>
</tr>
<tr>
<td>64</td>
<td>giga</td>
<td>0.23</td>
<td>256³</td>
<td>nomag/noentro/hyp</td>
</tr>
<tr>
<td>64</td>
<td>fe1</td>
<td>0.164</td>
<td>512³</td>
<td>nomag/noentro/hyp</td>
</tr>
<tr>
<td>64</td>
<td>giga</td>
<td>0.091</td>
<td>512³</td>
<td>nomag/noentro/hyp</td>
</tr>
<tr>
<td>64</td>
<td>giga</td>
<td>0.150</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>giga</td>
<td>0.166</td>
<td>512³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Gridur</td>
<td>0.25</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Ukaff</td>
<td>0.17</td>
<td>512³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Steno</td>
<td>0.075</td>
<td>512³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Neolith</td>
<td>0.0695</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Ferlin</td>
<td>8.51</td>
<td>150×128²</td>
<td>Li mechanism</td>
</tr>
<tr>
<td>64</td>
<td>Ferlin</td>
<td>8.156</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Akka</td>
<td>0.038</td>
<td>256²×512</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Triolith</td>
<td>0.0146</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Triolith</td>
<td>0.0164</td>
<td>256³</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>64</td>
<td>Hermit</td>
<td>0.101</td>
<td>256×512×256</td>
<td>spherical conv/magn</td>
</tr>
<tr>
<td>64</td>
<td>Sisu</td>
<td>0.00205</td>
<td>256×512×256</td>
<td>spherical conv/magn</td>
</tr>
<tr>
<td>72</td>
<td>Kraken</td>
<td>0.093</td>
<td>192×384×64</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>72</td>
<td>Kraken</td>
<td>0.151</td>
<td>96×192×16</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>72</td>
<td>Kraken</td>
<td>0.091</td>
<td>192×384×32</td>
<td>magn/noentro</td>
</tr>
<tr>
<td>72</td>
<td>Kraken</td>
<td>0.071</td>
<td>384×768×64</td>
<td>magn/noentro</td>
</tr>
</tbody>
</table>
The machines we have used can be characterized as follows:

**NI3**: 500 MHz Pentium III single CPU; RedHat Linux 6.2; 256 MB memory

**Nq0**: 931 MHz Pentium III single CPU; RedHat Linux 7.3; 0.5 GB memory

**Nq[1-4]**: 869 MHz Pentium III dual-CPU cluster; RedHat Linux 7.3; 0.77 GB memory per (dual) node

**Nq[5-6]**: 1.2 GHz Athlon dual-CPU cluster; RedHat Linux 7.3; 1 GB memory per (dual) node

**Kabul**: 1.9 GHz Athlon dual-CPU cluster; 1 GB memory per (dual) node; 256 kB cache per CPU; Gigabit ethernet; SuSE Linux 8.0; LAM-MPI

**Cincinnatus**: 1.7 GHz Pentium 4 single CPU; 1 GB memory; 256 kB cache per CPU; SuSE Linux 7.3

**Horseshoe (fe1, giga, and giga2)**: consists of different subclusters. The old one (queue name: workq, referred to as fe1) 2.0 GHz Pentium 512 single CPU; 25x 24-port fast ethernet switches with gigabit ethernet uplink; 1 30-port gigabit ethernet switch; 1 GB memory. The next generation has gigabit switches directly between nodes, and 2.6 GHz processors. The third generation (giga2) has 3.2 GHz processors (most of which have 1 GB, some 2 GB), is organized in 2 blocks interconnected with 2 Gb links, with 10 Gb uplinks within each block.

**Ukaff**: SGI Origin 3000; 400 MHz IP35 CPUs; IRIX 6.5; native MPI

**Mhd**: EV6 Compaq cluster with 4 CPUs per node; 4 GB memory per node (i.e. 1 GB per CPU) OSF1 4.0; native MPI

**Sander and Rasmussen**: Origin 3000

**Steno** 118 node IBM cluster with dual node AMD Opteron processors with 10 Gb infiniband network, compiled with pgf90 -fastsse -tp k8-64e (Copenhagen).

**Gridur**: Origin 3000

**Luci**: (full name Lucidor) is an HP Itanium cluster, each of the 90 nodes has two 900 MHz Itanium 2 "McKinley" processors and 6 GB of main memory. The interconnect is myrinet.

**Lenn**: (full name Lennsgren) is a Dell Xeon cluster with 442 nodes. Each node has two 3.4GHz “Nocona” Xeon processors and 8GB of main memory. A high performance Infiniband network from Mellanox is used for MPI traffic.

**Kraken**: Cray Linux Environment (CLE) 3.1, with a peak performance of 1.17 PetaFLOP; the cluster has 112,896 cores, 147 TB of memory, in 9,408 nodes. Each node has two 2.6 GHz six-core AMD Opteron processors (Istanbul), 12 cores, and 16 GB of memory. Connection via Cray SeaStar2+ router.
**Hermit:** Cray XE6 with 7104 2.3 GHz AMD Interlagos 16 core processors (113,664 cores in total), nodes with either 1 or 2 GB of memory per core.

**Sisu:** Cray XC30 with 1472 2.6 GHz Intel (Xeon) Sandy Bridge 8 core (E5-2670) processors (11,776 cores in total), 2 GB of memory per core.

**Beskow:** Cray XC40 with 2.3 GHz Intel (Xeon) Haswell 16 core (E5-2698v3) processors (67,456 cores in total), 2 GB of memory per core. Theoretical peak performance 2.43 pflops.

Table 8 shows a similar list, but for a few well-defined sample problems. The svn check-in patterns are displayed graphically in Fig 1.

![Figure 12: Scaling results on three different machines. The thin straight line denotes perfectly linear scaling.](image1)

![Figure 13: Scaling results of particle-mesh problem on Blue Gene/P on up to 4096 cores. The different lines denote different particle-mesh schemes (NGP=Nearest Grid Point, TSC=Triangular Shaped Cloud) and whether self-gravity is included (FFT).](image2)
A.1 Test case

In the following test samples, we run isothermal magnetohydrodynamics in a periodic domain. Power spectra are computed during the run, but our current parallelization of the Fourier transform requires that the meshpoint number is an integer multiple of the product of processor numbers in the $y$ and $z$ directions and the product of processor numbers in the $x$ and $y$ directions. In addition, the number of processors in one direction

---

Table 8: Like previous table, but for the versions from the ‘samples’ directory.

<table>
<thead>
<tr>
<th>proc(s)</th>
<th>machine</th>
<th>µs/pt step</th>
<th>resol.</th>
<th>mem./proc</th>
<th>when</th>
<th>who</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>conv-slab</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Mhd</td>
<td>6.45</td>
<td>32³</td>
<td>4 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
<tr>
<td>1</td>
<td>Cincinnatus</td>
<td>4.82</td>
<td>32³</td>
<td>3 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
<tr>
<td>1</td>
<td>Cincinnatus</td>
<td>11.6</td>
<td>64³</td>
<td>14 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
<tr>
<td>1</td>
<td>Cincinnatus</td>
<td>20.8</td>
<td>128³</td>
<td>93 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
<tr>
<td>1</td>
<td>Kabul</td>
<td>3.91</td>
<td>32³</td>
<td>23-jul-02</td>
<td>wd</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Kabul</td>
<td>3.88</td>
<td>64³</td>
<td>23-jul-02</td>
<td>wd</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Kabul</td>
<td>4.16</td>
<td>128³</td>
<td>93 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
<tr>
<td><strong>conv-slab-flat</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Kabul</td>
<td>3.02</td>
<td>128³×32</td>
<td>29 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
<tr>
<td>2</td>
<td>Kabul</td>
<td>1.81</td>
<td>128³×32</td>
<td>18 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
<tr>
<td>4</td>
<td>Kabul</td>
<td>1.03</td>
<td>128³×32</td>
<td>11 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
<tr>
<td>8</td>
<td>Kabul</td>
<td>0.87</td>
<td>128³×32</td>
<td>9 MB</td>
<td>23-jul-02</td>
<td>wd</td>
</tr>
</tbody>
</table>

should not be so large that the number of mesh points per processor becomes comparable to or less than the number of ghost zones (which is 6).

A.2 Running the code

To run the code, get one of the sample run directories, e.g.,
The relevant file to be changed is `src/cparam.local`
in particular the values of ncpus, nprocx, nprocy, and nxgrid. Once they are chosen, say make, and submit start_run.csh.

A.3 Triolith

On Triolith, strong scaling tests have been performed for three mesh sizes. The time per time step and mesh point is given for different processor numbers and layouts. Generally, it is advantageous to keep the number of processors in the $x$ direction small.

Comments. Although on Triolith the number of processors per node is 16, resolutions with one or two powers of 3 (such as 576) still work well. Furthermore, the number of processors above which the scaling becomes poor increases quadratically with the number of mesh points. This implies that the RAM per processor increases linearly with the problem size per direction. However, this will not be a limitation, because even for $2304^3$ meshes, the required RAM is still below 100 MB.

A.4 Lindgren

On Lindgren, we have performed weak scaling tests and compare with weak scaling results for Triolith. Triolith is about twice as fast as Lindgren.
Figure 16: Strong scaling on Triolith (2014).

Figure 17: Comparison Triolith (black, plus signs) and Lindgren (red, triangles). Weak scaling (2014).
### Table 9: Triolith timings

<table>
<thead>
<tr>
<th>proc</th>
<th>µs</th>
<th>resol.</th>
<th>layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0.075 µs</td>
<td>128³</td>
<td>2x2x4</td>
</tr>
<tr>
<td>16</td>
<td>0.065 µs</td>
<td>128³</td>
<td>1x4x4</td>
</tr>
<tr>
<td>16</td>
<td>0.0544 µs</td>
<td>256³</td>
<td>1x4x4</td>
</tr>
<tr>
<td>64</td>
<td>0.0146 µs</td>
<td>256³</td>
<td>1x8x8</td>
</tr>
<tr>
<td>64</td>
<td>0.0164 µs</td>
<td>256³</td>
<td>2x4x8</td>
</tr>
<tr>
<td>256</td>
<td>0.0049 µs</td>
<td>256³</td>
<td>1x16x16</td>
</tr>
<tr>
<td>512</td>
<td>0.0035 µs</td>
<td>256³</td>
<td>2x16x16</td>
</tr>
<tr>
<td>1024</td>
<td>0.00236 µs</td>
<td>256³</td>
<td>2x16x32</td>
</tr>
<tr>
<td>1024</td>
<td>0.00127 µs</td>
<td>512³</td>
<td>2x16x32</td>
</tr>
<tr>
<td>1024</td>
<td>0.00129 µs</td>
<td>512³</td>
<td>4x16x16</td>
</tr>
<tr>
<td>2048</td>
<td>9.34×10⁻⁴ µs</td>
<td>512³</td>
<td>4x16x32</td>
</tr>
<tr>
<td>2304</td>
<td>0.00107 µs</td>
<td>576³</td>
<td>4x18x32</td>
</tr>
<tr>
<td>4096</td>
<td>3.6×10⁻⁴ µs</td>
<td>1024³</td>
<td>4x32x32</td>
</tr>
<tr>
<td>4096</td>
<td>3.8×10⁻⁴ µs</td>
<td>1024³</td>
<td>8x16x32</td>
</tr>
<tr>
<td>4096</td>
<td>4.2×10⁻⁴ µs</td>
<td>1024³</td>
<td>4x16x64</td>
</tr>
<tr>
<td>4608</td>
<td>7.38×10⁻⁴ µs</td>
<td>576³</td>
<td>8x18x32</td>
</tr>
<tr>
<td>4608</td>
<td>2.66×10⁻⁴ µs</td>
<td>1152³</td>
<td>4x32x36</td>
</tr>
<tr>
<td>4608</td>
<td>3.03×10⁻⁴ µs</td>
<td>1152³</td>
<td>4x36x32</td>
</tr>
<tr>
<td>4608</td>
<td>3.12×10⁻⁴ µs</td>
<td>1152³</td>
<td>4x18x64</td>
</tr>
<tr>
<td>4608</td>
<td>2.36×10⁻⁴ µs</td>
<td>2304³</td>
<td>2x32x72</td>
</tr>
<tr>
<td>8192</td>
<td>1.475×10⁻⁴ µs</td>
<td>2048³</td>
<td>4x32x64</td>
</tr>
<tr>
<td>9216</td>
<td>0.00104 µs</td>
<td>576³</td>
<td>16x18x32</td>
</tr>
<tr>
<td>9216</td>
<td>1.276×10⁻⁴ µs</td>
<td>2304³</td>
<td>4x36x64</td>
</tr>
<tr>
<td>9216</td>
<td>1.30×10⁻⁴ µs</td>
<td>2304³</td>
<td>4x32x72</td>
</tr>
</tbody>
</table>

### Table 10: Lindgren timings

<table>
<thead>
<tr>
<th>proc</th>
<th>µs</th>
<th>resol.</th>
<th>layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>1536</td>
<td>0.00171 µs</td>
<td>512²×384</td>
<td>2x32x24</td>
</tr>
<tr>
<td>2048</td>
<td>0.00129 µs</td>
<td>512²×1024</td>
<td>1x32x64</td>
</tr>
<tr>
<td>2048</td>
<td>0.00129 µs</td>
<td>1024²×2048</td>
<td>1x32x64</td>
</tr>
<tr>
<td>4096</td>
<td>4.6×10⁻⁴ µs</td>
<td>2048³</td>
<td>4x16x64</td>
</tr>
<tr>
<td>9216</td>
<td>2.36×10⁻⁴ µs</td>
<td>2304³</td>
<td>4x48x48</td>
</tr>
</tbody>
</table>
B Coding standard

The numerous elements that make up the PENCIL CODE are written in a consistent style that has evolved since it was first created. Many people have contributed their knowledge and experience with in this and the result is what we believe is and extremely readable and manageable code.

As well as improving the readability of the code, by having some naming conventions for example aids greatly in understanding what the code does.

There is a standard for all aspects of the code, be it Fortran source, shell scripts, Perl scripts, LaTeX source, Makefiles, or otherwise. Where nothing has been explicitly stated it is recommended that similar existing examples found in the code are used as a template.

B.1 File naming conventions

All files with the exception of the ‘Makefile’s are given lowercase filenames.

Fortran source files all have the ‘.f90’ extension. Files that contain ‘non-executable code’ i.e. declarations that are included into other files are given the extension ‘.h’ and those that are generated dynamically at compile time have an ‘.inc’ extension.

Fortran source code defining a module is placed in files whose names begin with the Fortran module name in all lowercase. Where there exist multiple implementations of a specific module the filenames are extended using and with a brief name relating to what they do.

Text files containing parameters to be read by the code at run time are placed in files with the extension ‘.in’

B.2 Fortran Code

The code should remain fully compatible with the Fortran90 standard. This ensures that the code will run on all platforms. Indeed, an important aspect of PENCIL CODE philosophy is to be maximally flexible. This also means that useful non-standard extensions to the code should be hidden in and be made accessible through suitable non-default modules.

Fortran is not case-sensitive but in almost all instances we prescribe some form of capitalization for readability.

In general all Fortran code including keywords, variable names etc. are written in lowercase. Some of the coding standard has already been discussed in Sect. [9.1]. Here we discuss and amplify some remaining matters.

B.2.1 Indenting and whitespace

Whitespace should be removed from the end of lines.

Blank lines are kept to a minimum, and when occurring in subroutines or functions are replaced by a single ‘!’ in the first column.
Tab characters are not used anywhere in the code. Tab characters are not in fact allowed by the Fortran standard and compilers that accept them do so as an extension.

All lines are kept to be not more than 80 characters long. Where lines are longer they must be explicitly wrapped using the Fortran continuation character ‘&’. Longer lines (up to 132 characters) and additional spaces are allowed in cases where the readability of the code is enhanced, e.g. when one line is followed by a similar one with minor differences in some places.

Code in syntactic blocks such as if–endif, do–enddo, subroutine–endsubroutine etc. is always indented by precisely two spaces. The exception to this is that nested loops where only the innermost loop contains executable code should be written with the do–enddo pairs at the same level of indentation,

```fortran
  do n=n1,n2
    do m=m1,m2
      [...]  
    enddo
  enddo
```

Alternatively nested loops may be written on a single line, i.e.

```fortran
  do n=n1,n2; do m=m1,m2
    [...]  
  enddo; enddo
```

**B.2.2 Comments**

Descriptive comments are written on their own lines unless there is a strong reason to do otherwise. Comments are never indented and the ‘!’ should appear in the first column followed by two spaces and then the text of the comment. Extremely short comments may follow at the end of a line of code, provided there is space.

Comments also must not exceed the 78 character line length and should be wrapped onto more lines as needed.

Typically comments should appear with a blank commented line above and below the wrapped text of the comment.

All subroutine/functions begin with a standard comment block describing what they do, when and by whom they were created and when and by whom any non-trivial modifications were made.

Comments should be written in sentences using the usual capitalization and punctuation of English, similar to how text is formatted in an e-mail or a journal article.

For example:

```fortran
  some fortran code
  some more fortran code
  !
  ! A descriptive comment explaining what the following few lines
  ! of code do.
  !
  the fortran code being described
```
Subroutines and functions are started with a comment block describing what they do, when and by whom they were created and when and by whom any non-trivial modifications were made. The layout of this comment block is a standard, for example:

```fortran
!***********************************************************************
! subroutine initialize_density(f,lstarting)
!
! Perform any post-parameter-read initialization i.e. calculate derived
! parameters.
!
! For compatibility with other applications, we keep the possibility
! of giving diffrho units of dxmin*cs0, but cs0 is not well defined general.
!
! 24-nov-02/tony: coded
! 1-aug-03/axel: normally, diffrho should be given in absolute units
!
```

where dates are written in dd-mmm-yy format as shown and names appearing after the '/` are either the users cvs login name or, where such exists amongst the PENCIL CODE community, the accepted short form (≈ 4 characters) of the authors name.

### B.2.3 Module names

The names of modules are written with initial letter capitalization of each word and the multiple words written consecutively without any separator.

### B.2.4 Variable names

Variable are given short but meaningful names and written in all lowercase. Single character names are avoided except for commonly used loop indices and the two code data structures of the PENCIL CODE: ‘f’ the main state array (see 9.3) and ‘p’ the pencil case structure (see 9.6).

Quantities commonly represented by a particular single character in mathematics are typically given names formed by repeating the character (usually in lowercase), e.g. the velocity $u$ becomes ‘uu’, specific entropy $s$ becomes ‘ss’ etc.

Temperature in variable names is denoted with a capital $T$ so as not to be confused with time as represented by a lowercase $t$. Note however the since Fortran is not case sensitive the variables for example ‘TT’ and ‘tt’ are the same so distinct names must be used. For this reason time is usually represented by a single $t$ contrary to the above guideline.
The natural log of a quantity is represented by using adding ‘ln’ to its name, for example log of temperature would be ‘lnTT’.

There are some standard prefixes used to help identify the type and nature of variables they are as follows:

- **i** – Denotes integer variables typically used as array indices.
- **i** – Denotes pencil case array indices.
- **idiag** – Denotes diagnostic indices.
- **l** – Denotes logical/boolean flags
- **cdt** – Denotes timestep constraint parameters.
- **unit** – Denotes conversion code/physics unit conversion parameters.

### B.2.5 Emacs settings

Here are some settings from wd’s ‘~/.emacs’ file:

```emacs
;;; ~/.f90.emacs
;;; Set up indentation and similar things for coding the \{sc Pencil Code\}.
;;; Most of this can probably be set through Emacs’ Customize interface
;;; as well.
;;; To automatically load this file, put the lines
;;; (if (file-readable-p "~/.f90.emacs")
;;;     (load-file "~/.f90.emacs"))
;;; into your ~/.emacs file.

;;; F90-mode indentation widths
(setq f90-beginning-ampersand nil) ; no 2nd ampersand at continuation line
(setq f90-do-indent 2)
(setq f90-if-indent 2)
(setq f90-type-indent 2)
(setq f90-continuation-indent 4)

;;; Don’t use any tabs for indentation (with TAB key).
;;; This is actually already set for F90-mode.
(setq-default indent-tabs-mode nil)

;;; Ensure Emacs uses F90-mode (and not Fortran-mode) for F90 files:
(setq auto-mode-alist
  (append
   '("\.[fF]90$" . f90-mode)
   ("\.[iI]nc$" . f90-mode)
  )
  auto-mode-alist))
```

;;; Make M-Backspace behave in Xemacs as it does in GNU Emacs. The default
;;; behavior is apparently a long-known bug the fix for which wasn’t
;; propagated from fortran.el to f90.el.
;; (http://list-archive.xemacs.org/xemacs-patches/200109/msg00026.html):
(add-hook 'f90-mode-hook
  (function (lambda ()
    (define-key f90-mode-map [(meta backspace)] 'backward-kill-word)
  )))

B.3 Other best practices

When implementing IF or SELECT blocks always write code for all cases – including the default or else case. This should be done even when that code is only a call to raise an error that the case should not have been reached. If you see a missing case anywhere then do add it. These failsafes are essential in a large multi-purpose multi-user code like the PENCIL CODE.

If a case is supposed to do nothing and it may be unclear that the coder has recognized this fact then make it explicit by adding the default case with a comment like

! Do Nothing. The compiler will clean away any such empty blocks.

B.4 General changes to the code

It is sometimes necessary to do major changes to the code. Since this may affect many people and may even be controversial among the developers, such changes are restricted to the time of the next Pencil Code User Meeting. Such meetings are advertised on http://www.nordita.org/software/pencil-code/ under the news section. Notes about previous such meetings can be found under http://www.nordita.org/software/pencil-code/UserMeetings/.

Major changes can affect those developers who have not checked in their latest changes for some time. Before doing such changes it is therefore useful to contact the people who have contributed to the latest developments on that module. If it is not functional or otherwise in bad shape, it should be moved to ‘experimental’, i.e. one says svn mv file.f90 experimental/file.f90. However, any such directory change constitutes a major change in itself and should be performed in agreement with those involved in the development. Otherwise any file that has been changed in the mean time will end up being outside revision control, which is to be avoided at all cost.
C Some specific initial conditions

C.1 Random velocity or magnetic fields

Obtained with inituu='gaussian-noise' (or initaa='gaussian-noise'). The vector $u$ (or $A$) is set to normally distributed, uncorrelated random numbers in all meshpoints for all three components. The power spectrum of $u$ ($A$) increases then quadratically with wavenumber $k$ (without cutoff) and the power spectrum of $\omega$ ($B$) increases like $k^4$.

Note that a random initial condition contains significant power at the Nyquist frequency ($k_{Ny} = \pi/N$, where $N$ is the number of mesh points). In a decay calculation, because of the discretization error, such power decays slower than it ought to; see Fig. 18 where we show the evolution for a random initial velocity field for $64^3$ meshpoints, $\nu = 5 \times 10^{-2}$ (fairly large!), and nfilter=30.

It is clearly a good idea to filter the initial condition to prevent excess power at $k_{Ny}$. On the other hand, such excess power is weak by comparison with the power at the energy carrying scale, so one does not see it in visualizations in real space. Furthermore, as seen from Fig. 18 for $k < k_{Ny}/2$ the power spectra for filtered and unfiltered initial conditions is almost the same.

![Figure 18: Velocity power spectra at three different times with and without filtering of the initial condition.](image)

C.2 Turbulent initial with given spectrum

The most general procedure for producing an initial condition with a turbulent spectrum is inituu='power_randomphase_hel', which allows one to set two different slopes, together with an exponential cutoff as well as a Gaussian peak within the spectrum. By default, the field is solenoidal unless one puts lskip_projection=.true. and can have fractional helicity by setting rehelic_uu to a value between $-1$ and $1$. By default it is 0, which means it is nonhelical.
The spectral indices initpower and initpower2 refer to energy spectral indices. By default, initpower2=-5/3, corresponding to a Kolmogorov spectrum. For a delta-correlated spectrum, we have to put initpower2=2, corresponding to a $k^2$ energy spectrum for kinetic energy. This would be suitable for the subinertial range from $k = 1$ to $k = k_p$ (corresponding to the variable kpeak).

If cutoff=0, no cutoff will be imposed. Otherwise, the spectrum will be multiplied by an exponential function with $\exp(-k^{2n})$, where $n = \text{'ncutoff=1'}$ by default.

Example, for ampluu=1e-1, initpower=4., and kpeak=3., we get urms=3.981E-01 when relhel_uu=1 and urms=3.981E-01 when relhel_uu=0 and urms=5.560E-01 when relhel_uu=1. The urms values scale linearly with ampluu and for initpower=2 also approximately linearly with kpeak. For the magnetic field, we initialize the magnetic vector potential, so to get a $k^4$ spectrum, we have to put initpower_aa=2. Everything else is analogous; see, e.g.,

```plaintext
&hydro_init_pars
  inituu='power_randomphase_hel', ampluu=1e-1, initpower=4., kpeak=3.
  relhel_uu=0., cutoff=30.
/
&magnetic_init_pars
  initaa='power_randomphase_hel', amplaa=1e-1, initpower_aa=2., kpeak_aa=3.
  relhel_aa=0., cutoff_aa=30.
/
```

for which we get urms=3.981E-01 and brms=3.871E-01.

### C.3 Beltrami fields

Obtained with inituu='Beltrami-z' or initaa='Beltrami-z'.

$$\mathbf{A} = (\cos z, \sin z, 0), \quad \text{or} \quad \mathbf{u} = (\cos z, \sin z, 0)$$ (116)

### C.4 Magnetic flux rings: initaa='fluxrings'

This initial condition sets up two interlocked thin magnetic tori (i.e. thin, torus-shaped magnetic flux tubes). One torus of radius $R$ lying in the plane $z = 0$ can be described in cylindrical coordinates, $(r, \varphi, z)$, by the vector potential

$$\mathbf{A} = \Phi_m \begin{pmatrix} 0 \\ 0 \\ -\theta(r-R)\delta(z) \end{pmatrix},$$ (117)

resulting in a magnetic field

$$\mathbf{B} = \Phi_m \begin{pmatrix} 0 \\ \delta(r-R)\delta(z) \\ 0 \end{pmatrix}.$$ (118)

Here $\Phi_m$ is the magnetic flux through the tube, $\theta(x)$ denotes the Heaviside function, and $\delta(x) = \theta'(x)$ (119)
is Dirac’s delta function.

Any smoothed versions of $\theta(x)$ and $\delta(x)$ will do, as long as the consistency condition (119) is satisfied. E.g. the pairs

\[
\delta_\varepsilon(x) = \frac{1}{\sqrt{2\pi\varepsilon^2}} e^{-\frac{x^2}{2\varepsilon^2}}, \quad \theta_\varepsilon(x) = \frac{1}{2} \left( 1 + \text{erf} \left( \frac{x}{\sqrt{2}\varepsilon} \right) \right)
\]

or

\[
\delta_\varepsilon(x) = \frac{1}{2\varepsilon} \frac{1}{\cosh^2 \frac{x}{\varepsilon}}, \quad \theta_\varepsilon(x) = \frac{1}{2} \left( 1 + \tanh \frac{x}{\varepsilon} \right)
\]

are quite popular. Another possibility is a constant or box-like profile with

\[
\delta_\varepsilon(x) = \frac{1}{2\varepsilon} \theta(|x| - \varepsilon), \quad \theta_\varepsilon(x) = \frac{1}{2} \left\{ 1 + \max[-1, \min(x/\varepsilon), 1] \right\}
\]

Note, however, that the Gaussian profile (120) is the only one that yields a radially symmetric (with respect to the distance from the central line of the torus) magnetic field profile $B_\phi = B_\phi(\sqrt{r-R})^2 + z^2)$ if $\varepsilon$ is sufficiently small.

In Cartesian coordinates, the vector potential (117) takes the form

\[
A = \Phi_{in} \begin{pmatrix}
0 \\ 0 \\ -\theta \left( \sqrt{x^2 + y^2} - R \right) \delta(z)
\end{pmatrix}.
\]

C.5 Vertical stratification

Gravity, $g = -\nabla \Phi$, is specified in terms of a potential $\Phi$. In slab geometry, $\Phi = \Phi(z)$, we have $g = (0, 0, g_z)$ and $g_z = -d\Phi/dz$.

Use `grav_profile='const'` together with `gravz=-1` to get

\[
\Phi = (z - z_\infty)(-g_z), \quad (-g_z) > 0.
\]

Use `grav_profile='linear'` to get

\[
\Phi = \frac{1}{2}(z^2 - z_\infty^2)\nu_g^2, \quad g_z = -\nu_g^2 z
\]

where $\nu_g$ is the vertical epicyclic frequency. For a Keplerian accretion disc, $\nu_g = \Omega$. For galactic discs, $\nu_g = 0.5\Omega$ is representative of the solar neighborhood.

The value of $z_\infty$ is determined such that $\rho = \rho_0$ and $c_s^2 = c_{s0}^2$ at $z = z_{ref}$. This depends on the values of $\gamma$ and the polytropic index $m$ (see below).

C.5.1 Isothermal atmosphere

Here we want $c_s = c_{s0} = \text{const}$. Using `initlnrho='isothermal'` means

\[
\ln \frac{\rho}{\rho_0} = -\gamma \frac{\Phi}{c_{s0}^2}.
\]

The entropy is then initialized to

\[
\frac{s}{c_p} = (\gamma-1) \frac{\Phi}{c_{s0}^2}.
\]

In order that $\rho = \rho_0$ and $c_s^2 = c_{s0}^2$ at $z = z_{ref}$, we have to choose $z_\infty = z_{ref}$. 

\[\text{C.5 Vertical stratification 129} \]
C.5.2 Polytropic atmosphere

For a polytropic equation of state, \( p = K \rho^\Gamma \), where generally \( \Gamma \neq \gamma \), we can write

\[
- \nabla h + T \nabla s = - \frac{1}{\rho} \nabla p = - \nabla \left( \frac{\Gamma K}{\Gamma - 1} \rho^{\Gamma - 1} \right) \equiv - \nabla \tilde{h},
\]

where we have introduced a pseudo enthalpy \( \tilde{h} \) as

\[
\tilde{h} = \frac{\Gamma K}{\Gamma - 1} \rho^{\Gamma - 1} = \left[ \left( 1 - \frac{1}{\gamma} \right) \left/ \left( 1 - \frac{1}{\Gamma} \right) \right. \right] h. \tag{129}
\]

Obviously, for \( \Gamma = \gamma \), the pseudo enthalpy \( \tilde{h} \) is identical to \( h \) itself. Instead of specifying \( \Gamma \), one usually defines the polytropic index \( m = 1/(\Gamma - 1) \). Thus, \( \Gamma = 1 + 1/m \), and

\[
\tilde{h} = (m+1) \left( 1 - \frac{1}{\gamma} \right) h \tag{130}
\]

This is consistent with a fixed entropy dependence, where \( s \) only depends on \( \rho \) like

\[
\frac{s}{c_p} = \left( \frac{\Gamma}{\gamma} - 1 \right) \ln \frac{\rho}{\rho_0}, \tag{131}
\]

and implies that

\[
\ln \frac{c_s^2}{c_{s0}^2} = (\Gamma - 1) \ln \frac{\rho}{\rho_0}. \tag{132}
\]

For hydrostatic equilibrium we require \( \tilde{h} + \Phi = \tilde{h}_0 = \text{const.} \) For gravity potentials that vanish at infinity, we can have \( \tilde{h}_0 \neq 0 \), i.e. a finite pseudo enthalpy at infinity. For \( g_z = -1 \) or \( g_z = -z \), this is not the case, so we put \( \tilde{h}_0 = 0 \), and therefore \( \tilde{h} = -\Phi \). Using \( c_s^2 = (\gamma - 1)h \) together with (130) we find

\[
c_s^2 = -\frac{\gamma}{m+1} \Phi. \tag{133}
\]

In order that \( \rho = \rho_0 \) and \( c_s^2 = c_{s0}^2 \) at \( z = z_{\text{ref}} \), we have to choose (remember that \( g_z \) is normally negative!)

\[
z_{\infty} = z_{\text{ref}} + (m+1) \frac{c_{s0}^2}{\gamma(-g_z)} \quad \text{for } \text{grav_profile} = '\text{const}', \tag{134}
\]

and

\[
z_{\infty}^2 = z_{\text{ref}}^2 + (m+1) \frac{c_{s0}^2}{2\gamma v_s^2} \quad \text{for } \text{grav_profile} = '\text{linear}'. \tag{135}
\]

Thus, when using \( \text{initlnrho} = '\text{polytropic_simple}' \) we calculate

\[
\ln \frac{c_s^2}{c_{s0}^2} = \ln \left[ -\frac{\gamma \Phi}{(m+1)c_{s0}^2} \right] \tag{136}
\]

and so the stratification is given by

\[
\ln \frac{\rho}{\rho_0} = m \ln \frac{c_s^2}{c_{s0}^2}, \quad \frac{s}{c_p} = \left( \frac{\Gamma}{\gamma} - 1 \right) m \ln \frac{c_s^2}{c_{s0}^2}. \tag{137}
\]
C.5.3 Changing the stratification

Natural: measure length in units of \( \frac{c^2 s_0}{g_z} \). Can increase stratification by moving \( z_{\text{top}} \) close to \( z_\infty \) or, better still, keeping \( z_{\text{top}} = 0 \) and moving \( z_{\text{bot}} \to -\infty \). Disadvantage: in the limit of weak stratification, the box size will be very small (in nondimensional units).

Box units: measure length in units of \( d \). Can increase stratification by increasing \( g_z \) to \( g_{\text{max}} \), which can be obtained by putting \( z_{\text{top}} = z_\infty \) in (134), so

\[
g_{\text{max}} = \frac{m+1}{\gamma} \frac{c^2 s_0}{z_{\text{top}} - z_{\text{ref}}}. \tag{138}
\]

For \( m = 1, \gamma = 5/3, z_{\text{top}} = 1, \) and \( z_{\text{ref}} = 0 \), for example, we have \( g_{\text{max}} = 6/5 = 1.2 \).

Gravitational box units: measure speed in units of \( \sqrt{g_z d} \). The limit of vanishing stratification corresponds to \( c^2 s_0 \to \infty \). This seems optimal if we want to approach the Boussinesq case.

In Hurlburt et al. (1984), \( z \) increased downward and the atmosphere always terminated at \( z = 0 \). In order to reproduce their case most directly, we put \( z_\infty = 0 \) and consider only negative values of \( z \). To reproduce their case with a density stratification of 1:1.5, we place the top of the model at \( z_{\text{top}} = -2 \) and the bottom at \( z_{\text{bot}} = -3 \). In addition, the reference height, \( z_{\text{ref}} \), is chosen to be at the top of the box, i.e. \( z_{\text{ref}} = -2 \). From Eq. (134) we have \( c^2 s_0 = \gamma (-g_z) (-z_{\text{ref}})/(m+1) \). Using \( (-g_z) = 1 \) and \( m = 1 \) we find \( c^2 s_0 = \gamma \), so \( c^2 s_0 = 1.291 \) (for \( \gamma = 5/3 \)). Values for other combinations are listed in Table 11.

<table>
<thead>
<tr>
<th>( \rho_{\text{bot}}/\rho_{\text{top}} )</th>
<th>( z_{\text{bot}} )</th>
<th>( z_{\text{top}} )</th>
<th>( c_{s0} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>3</td>
<td>2</td>
<td>1.291</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>0.5</td>
<td>0.645</td>
</tr>
<tr>
<td>6</td>
<td>1.2</td>
<td>0.2</td>
<td>0.408</td>
</tr>
<tr>
<td>11</td>
<td>1.1</td>
<td>0.1</td>
<td>0.289</td>
</tr>
<tr>
<td>21</td>
<td>1.05</td>
<td>0.05</td>
<td>0.204</td>
</tr>
</tbody>
</table>

C.5.4 The Rayleigh number

In Ref. [9] the Rayleigh number is defined as

\[
\text{Ra} = \frac{g d^4}{\overline{\nu} \overline{\chi}} \left( -\frac{d s/c_p}{dz} \right)_{\text{hydrostat}}, \tag{139}
\]

where the (negative) entropy gradient was evaluated in the middle of the box for the associated hydrostatic reference solution, and \( \overline{\chi} = K/(\overline{\rho} c_p) \) and either \( \overline{\nu} = \nu \) (if \( \nu \) was assumed constant) or \( \overline{\nu} = \mu/\overline{\rho} \) (if \( \mu \) was assumed constant). Note that \( \overline{\rho} \) is the average mass in the box per volume, which is conserved. For a polytrope we have

\[
\left( -\frac{d s/c_p}{dz} \right)_{\text{hydrostat}} = \left[ 1 - (m+1) \left( 1 - \frac{1}{\gamma} \right) \right] \frac{1}{z_\infty - z_m}, \tag{140}
\]

where \( z_m = (z_1 + z_2)/2 \). This factor was also present in the definition of Hurlburt et al. [20], but their definition differs slightly from Eq. (139), because they normalized the
density not with respect to the average value (which is constant for all times), but with respect to the value at the top of the initial hydrostatic solution. Since the Rayleigh number is proportional to $\rho^2$, their definition included the extra factor $[(z_\infty - z_m)/d]^2$. Therefore

$$Ra_{HTM} = \left(\frac{z_\infty - z_m}{d}\right)^{2m} \left(\frac{\rho_{top}}{\rho}\right)^2 Ra$$  \hspace{1cm} (141)

In the first model of Hurlburt et al. (1984), the Rayleigh number, $Ra_{HTM}$, was chosen to be 310 times supercritical, and the critical Rayleigh number was around 400, so $Ra_{HTM} = 1.25 \times 10^5$. In their model the density contrast was 1:1.5 and $m = 1$. This turns out to correspond to $Ra = 4.9 \times 10^4$, $F_{bot} = 0.0025$, and $K = 0.002$.

Another model that was considered by Hurlburt & Toomre (1988) had $Ra_{HTM} = 10^5$, a density contrast of 11, and had a vertical imposed magnetic field (Chandrasekhar number $Q = 72$). This corresponds to $Ra = 3.6 \times 10^8$, $K = 0.0011$, $F_{bot} = 0.0014$.

C.5.5 Entropy boundary condition

This discussion only applies to the case of convection in a slab. A commonly used lower boundary condition is to prescribe the radiative flux at the bottom, i.e. $F_{bot} = -KdT/dz$. Assuming that the density in the ghost zones has already been updated, we can calculate the entropy gradient from

$$F_{bot} = -K \frac{c_p^2 s}{c_p \gamma - 1} \left( (\gamma - 1) \frac{d\ln \rho}{dz} + \frac{d s/c_p}{dz} \right),$$  \hspace{1cm} (142)

which gives

$$\frac{ds/c_p}{dz} = -\frac{\gamma - 1}{\gamma} \left( \frac{c_p^2 F_{bot} K c_s^2}{c_p K c_s^2 + d\ln \rho dz} \right),$$  \hspace{1cm} (143)

for the derivative of the entropy at the bottom. This is implemented as the ‘c1’ boundary condition at the bottom.

C.5.6 Temperature boundary condition at the top

In earlier papers the temperature at the top was set in terms of the quantity $\xi_0$, which is the ratio of the pressure scale height relative to the depth of the unstable layer. Expressed in terms of the sound speed at the top we have

$$c_{s,\text{top}}^2 = \gamma \xi_0 gd.$$  \hspace{1cm} (144)

$$\xi_{s,\text{bot}}^2 = \left(\xi_0 + \frac{1}{m + 1}\right) \gamma gd.$$  \hspace{1cm} (145)

C.6 Potential-field boundary condition

The ‘pot’ [or currently rather the ‘pwd’] boundary condition for the magnetic vector potential implements a potential-field boundary condition in $z$ for the case of an $x$-$y$-periodic box. In this section, we discuss the relevant formulas and their implementation in the PENCIL CODE.
Table 12: Correspondence between $\xi_0$ and $c^2_{s,\text{bot}}$ in single layer polytropes.

<table>
<thead>
<tr>
<th>$\xi_0$</th>
<th>$c^2_{s,\text{bot}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.00</td>
<td>17.500</td>
</tr>
<tr>
<td>0.20</td>
<td>1.167</td>
</tr>
<tr>
<td>0.10</td>
<td>1.000</td>
</tr>
<tr>
<td>0.05</td>
<td>0.917</td>
</tr>
<tr>
<td>0.02</td>
<td>0.867</td>
</tr>
</tbody>
</table>

If the top boundary is at $z = 0$, the relevant potential field for $z > 0$ is given by

$$
\tilde{A}_x(k_x, k_y, z) = C_x(k_{xy}) e^{-\kappa z},
\tag{146}
$$

$$
\tilde{A}_y(k_x, k_y, z) = C_y(k_{xy}) e^{-\kappa z},
\tag{147}
$$

$$
\tilde{A}_z(k_x, k_y, z) = C_z(k_{xy}) e^{-\kappa z},
\tag{148}
$$

where

$$
\tilde{A}_i(k_x, k_y, z) \equiv \int e^{-ik_{xy} \cdot x} A_i(x, y, z) \, dx \, dy
\tag{149}
$$

is the horizontal Fourier transform with $k_{xy} \equiv (k_x, k_y, 0)$, and $k \equiv |k_{xy}|$. Note that this implies a certain gauge and generally speaking the $z$ dependence in Eq. (148) is completely arbitrary, but the form used here works well in terms of numerical stability.

At the very boundary, the potential field (146)–(148) implies

$$
\frac{\partial \tilde{A}}{\partial z} + \kappa \tilde{A} = 0,
\tag{150}
$$

and, due to natural continuity requirements on the vector potential, these conditions also hold for the interior field at the boundary.

**Robin boundary conditions and ghost points** To implement a homogeneous Robin boundary condition, i.e. a condition of the form

$$
\frac{df}{dz} + \kappa f = 0
\tag{151}
$$

using ghost points, we first write it as

$$
\frac{d}{dz} (f e^{\kappa z}) = 0
\tag{152}
$$

and implement this as symmetry condition for the variable $\phi(z) \equiv f(z) e^{\kappa z}$:

$$
\phi_{N-j} = \phi_{N+j}, \quad j = 1, 2, 3
\tag{153}
$$

(where $z_N$ is the position of the top boundary and $z_{N+1}, \ldots$ are the boundary points). In terms of $f$, this becomes

$$
f_{N+j} = f_{N-j} e^{-\kappa (z_{N+j} - z_{N-j})}.
\tag{154}
$$

Note that although the exponential term in Eq. (154) looks very much like the exterior potential field (146)–(148), our ghost-zone values do not represent the exterior field — they are rather made-up values that allow us to implement a local boundary condition at $z = 0$. 
C.7 Planet solution in the shearing box

In order to test the setup for accretion discs and the sliding periodic shearing sheet boundary condition, a useful initial condition is the so-called planet solution of Goodman, Narayan, & Goldreich [17].

Assume $s = 0$ (isentropy), so the equations in 2-D are

\begin{align}
    u_x u_{x,x} + (u_y^{(0)} + u_y) u_{x,y} &= 2\Omega u_y - h_{,x} \tag{155} \\
    u_x u_{y,x} + (u_y^{(0)} + u_y) u_{y,y} &= -(2 - q)\Omega u_x - h_{,y} \tag{156}
\end{align}

where $u_y^{(0)} = -q\Omega x$. Express $u$ in terms of a stream function, so $u = \nabla \times (\psi \hat{z})$, or

\begin{align}
    u_x &= \psi_{,y}, \quad u_y = -\psi_{,x}. \tag{157}
\end{align}

Ansatz for enthalpy

\begin{align}
    h &= \frac{1}{2}\sigma^2 \Omega^2 (R^2 - x^2 - \epsilon^2 y^2 - z^2 / \delta^2) \tag{158} \\
    \psi &= -\frac{1}{2}\sigma \Omega (R^2 - x^2 - \epsilon^2 y^2) - \frac{1}{2}q\Omega x^2 \tag{159}
\end{align}

This implies

\begin{align}
    u_x &= \sigma \Omega \epsilon^2 y, \quad u_y = (q - \sigma)\Omega x \tag{160}
\end{align}

and $u_{x,x} = u_{y,y} = 0$. Inserting into Eqs (155) and (156) yields

\begin{align}
    (-q + q - \sigma)\sigma \epsilon^2 &= 2(q - \sigma) + \delta^2 \tag{161} \\
    \sigma(q - \sigma) &= -(2 - q)\sigma + \delta^2 \tag{162}
\end{align}

where we have already canceled common $\Omega^2$ factors in both equations and common $\epsilon^2$ factors in the last equation. Simplifying both equations yields

\begin{align}
    -\sigma^2 \epsilon^2 &= 2(q - \sigma) + \delta^2 \tag{163} \\
    -\sigma^2 &= -2\sigma + \delta^2 \tag{164}
\end{align}

The second equation yields

\begin{align}
    \delta^2 &= (2 - \sigma)\sigma \tag{165}
\end{align}

and subtracting the two yields

\begin{align}
    \sigma^2 &= 2q / (1 - \epsilon^2) \tag{166}
\end{align}

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$\sigma$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.74</td>
<td>0.67</td>
</tr>
<tr>
<td>0.2</td>
<td>1.77</td>
<td>0.64</td>
</tr>
<tr>
<td>0.3</td>
<td>1.82</td>
<td>0.58</td>
</tr>
<tr>
<td>0.4</td>
<td>1.89</td>
<td>0.46</td>
</tr>
<tr>
<td>0.48</td>
<td>1.97</td>
<td>0.22</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 13: Dependence of $\epsilon$ and $\delta$ on $\epsilon$. 
D. Some specific boundary conditions

In this section, we formulate and discuss the implementation of some common boundary conditions in spherical and cylindrical coordinates.

D.1 Perfect-conductor boundary condition

This is a popular boundary condition for the magnetic field; it implies that

\[ B_n = 0 \]  

(167)

and

\[ E_t = 0 \]  

(168)

on the boundary, where the subscript \( n \) denotes the normal component, and \( E_t \) denotes the tangential components of the electric field.

In Cartesian geometry, these conditions can be implemented by setting the two tangential components of the vector potential \( A \) to zero on the boundary. It is easy to see that this also works in arbitrary curvilinear coordinates.

In particular, for spherical coordinates on a radial boundary we must have

\[ r \sin \theta B_r = \partial_\theta (\sin \theta A_\phi) - \partial_\phi A_\theta = 0 \]  

(169)

This can be achieved by setting

\[ A_\phi = A_\theta = 0 \]  

(170)

everywhere on the boundary. Note that this does not impose any condition on the radial component of the vector potential.

Next, in spherical coordinates on a boundary with constant \( \theta \), we must have

\[ B_\theta = \frac{1}{r \sin \theta} \partial_\phi A_r - \frac{1}{r} \partial_r (r A_\phi) = 0 \]  

(171)

Again this can be achieved by \( A_r = A_\phi = 0 \).

D.2 Stress-free boundary condition

On an impenetrable, stress-free boundary, we have

\[ u_n = 0 \]  

(172)

and the shear stress components \( S_{nt} \) must vanish for any tangential direction \( t \). At the radial boundary, the relevant components of the strain tensor (required to vanish at the boundary) are:

\[ S_{r\theta} = \frac{1}{r} \partial_\theta u_r + r \partial_r \left( \frac{u_\theta}{r} \right) \]  

(173)

\[ S_{r\phi} = \frac{1}{r \sin \theta} \partial_\phi u_r + r \partial_r \left( \frac{u_\phi}{r} \right) \]  

(174)

Both of them vanish if we require

\[ u_r = 0 \; , \; \partial_r (u_\theta/r) = 0 \; , \; \partial_r (u_\phi/r) = 0 \]  

(175)
We implement this by requiring \( u_r \) to be antisymmetric and \( u_\theta/r \) and \( u_\phi/r \) to be symmetric with respect to the boundary.

The more general condition

\[
r^\alpha \partial_r \left( u_\theta/r^\alpha \right) = \partial_r u_\theta - \frac{\alpha}{r} u_\theta = 0
\]

(176)

(where \( \alpha \) is a constant) can be implemented by requiring \( u_\theta/r^\alpha \) to be symmetric.

At a boundary \( \theta = \text{const} \), the stress-free boundary condition will take the form

\[
S_{r\theta} = \frac{1}{r} \partial_\theta u_r + r \partial_r \left( \frac{u_\theta}{r} \right) = 0 ,
\]

(177)

\[
S_{\theta\phi} = \frac{1}{r \sin \theta} \partial_\phi u_\theta + \sin \theta \partial_\theta \left( \frac{u_\phi}{r \sin \theta} \right) = 0 .
\]

(178)

With \( u_\theta = 0 \), the first condition gives \( \partial_\theta u_r = 0 \), i.e. we require \( u_r \) to be symmetric with respect to the boundary. The second condition requires

\[
\frac{\sin \theta}{r} \partial_\theta \left( \frac{u_\phi}{\sin \theta} \right) = 0
\]

(179)

and is implemented by requiring \( u_\phi/\sin \theta \) to be symmetric.

### D.3 Normal-field-radial boundary condition

While unphysical, this boundary condition is often used as a cheap replacement for a potential-field condition for the magnetic field. It implies that the two tangential components of the magnetic field are zero at the boundary, while the normal component is left unconstrained.

At a radial boundary, this gives:

\[
B_\theta = \frac{1}{r \sin \theta} \partial_\phi A_r - \frac{1}{r} \partial_r (r A_\phi) = 0 ,
\]

(180)

\[
B_\phi = \frac{1}{r} \partial_r (r A_\theta) - \frac{1}{r} \partial_\theta A_r = 0 .
\]

(181)

Which are satisfied by setting

\[
A_r = 0 , \quad \partial_r (r A_\theta) = 0 , \quad \partial_r (r A_\phi) = 0 ,
\]

(182)

and these are implemented by requiring \( A_r \) to be antisymmetric, and \( r A_\theta \) and \( r A_\phi \) to be symmetric.

On a boundary \( \theta = \text{const} \), we have

\[
r \sin \theta B_r = \partial_\theta (\sin \theta A_\phi) - \partial_\phi A_\theta = 0 ,
\]

(183)

\[
r B_\phi = \partial_r (r A_\theta) - \partial_\theta A_r = 0
\]

(184)

which can be achieved by setting

\[
\partial_\theta A_r = 0 , \quad A_\theta = 0 , \quad \partial_\theta (\sin \theta A_\phi) = 0 .
\]

(185)

We thus require \( A_r \) and \( \sin \theta A_\phi \) to be symmetric, and \( A_\theta \) to be antisymmetric.
E. High-frequency filters

Being high order, PENCIL CODE has much reduced numerical dissipation. In order to perform inviscid simulations, high-frequency filters can be used to provide extra dissipation for modes approaching the Nyquist frequency. Usual Laplacian viscosity $\nu \nabla^2 u$ is equivalent to a multiplication by $k^2$ in Fourier space, where $k$ is the wavenumber. Another tool is hyperviscosity, which replaces the $k^2$ dependency by a higher power-law, $k^n$, $n>2$. The idea behind it is to provide large dissipation only where it is needed, at the grid scale (high $k$), while minimizing it at the largest scales of the box (small $k$). In principle, one can use as high $n$ as desired, but in practice we are limited by the order of the code. A multiplication by $k^n$ is equivalent to an operator $\nabla^n$ in real space. As PENCIL CODE is of sixth order, three ghost cells are available in each direction, thus the sixth-order derivative is the highest we can compute. The hyperdissipation we use is therefore $\nabla^6$, or $k^6$ is Fourier space. Figure 19 illustrates how such tool maximizes the inertial range of a simulation.

Simplified hyperdiffusivity has been implemented for many dynamical variables and can be found in the respective modules. A strict generalization of viscosity and resistivity to higher order is implemented in the modules ‘hypervisc_strict_2nd’ and ‘hyperresi_strict_2nd’.

Hyperdiffusivity is meant purely as a numerical tool to dissipate energy at small scales and comes with no guarantee that results are convergent with regular second order dissipation. See Haugen & Brandenburg (2004) for a discussion. In fact, large-scale dynamo action is known to be seriously altered in simulations of closed systems where magnetic helicity is conserved: this results in prolonged saturation times and enhanced saturation amplitudes (Brandenburg & Sarson 2002).

E.1 Conservative hyperdissipation

It is desirable to have this high-frequency filter obeying the conservation laws. So, for density we want a mass conserving term, for velocities we want a momentum conserving term, for magnetic fields we want a term conserving magnetic flux, and for entropy we want an energy conserving term. These enter as hyperdiffusion, hyperviscosity, hyperresistivity, and hyper heat conductivity terms in the evolution equations. To ensure conservation under transport, they must take the form of the divergence of the flux $\mathcal{J}$ of the quantity $\psi$, so that Gauss theorem applies and we have

$$\frac{\partial \psi}{\partial t} + \nabla \cdot \mathcal{J} = 0 \quad (186)$$

For density, the flow due to mass diffusion is usually taken as the phenomenological Fick’s Law

$$\mathcal{J} = -D \nabla \rho \quad (187)$$

i.e., proportional to the density gradient, in the opposite direction. This leads to the usual Laplacian diffusion

$$\frac{\partial \rho}{\partial t} = D \nabla^2 \rho \quad (188)$$
Figure 19: Dissipation acting on a scalar field $\psi$, for $n=1$ (Laplacian dissipation) and $n=3$ (third-order hyperdissipation). The field is initially seeded with noise (upper panel). For $n=3$ the large scale is not affected as much as in the $n=1$ case, which is seen by the larger wiggling of the latter in the middle panel. In Fourier space (lower panel) we see that near the grid scale both formulations give strong dissipation. It also illustrates that at the large scales ($k\approx 1$), the effect of $n=3$ is indeed negligible.

under the assumption that the diffusion coefficient $D$ is isotropic. Higher order hyperdiffusion of order $2n$ involves a generalization of Eq. (187), to

$$\mathcal{J}^{(n)} = (-1)^n D^{(n)} \nabla^{2n-1} \rho. \quad (189)$$

In our case, we are interested in the case $n = 3$, so that the hyperdiffusion term is

$$\frac{\partial \rho}{\partial t} = D^{(3)} \nabla^6 \rho. \quad (190)$$

The hyperdiffusion coefficient $D^{(3)}$ can be calculated from $D$ assuming that at the Nyquist frequency the two formulations (188) and (190) yield the same quenching. Considering a wave as a Fourier series in one dimension ($x$), one element of the series is expressed as

$$\psi_k = Ae^{i(kx + \omega t)} \quad (191)$$
Plugging it into the second order diffusion equation (188) we have the dispersion condition \(i\omega = -Dk^2\). The sixth order version (190) yields \(i\omega = -D^{(3)}k^6\). Equating both we have \(D^{(3)} = Dk^{-4}\). This condition should hold at the grid scale, where \(k = \pi/\Delta x\), therefore

\[
D^{(3)} = D \left( \frac{\Delta x}{\pi} \right)^4 \quad (192)
\]

For the magnetic potential, resistivity has the same formulation as mass diffusion

\[
\frac{\partial A}{\partial t} = -\eta \nabla \times B = \eta \nabla^2 A, \quad (193)
\]

where we used the Coulomb gauge \(\nabla \cdot A = 0\). The algebra is the same as above, also yielding \(\eta^{(3)} = \eta (\Delta x/\pi)^4\). For entropy, the heat conduction term is

\[
\frac{\partial S}{\partial t} = \frac{1}{\rho T} \nabla \cdot (K \nabla T), \quad (194)
\]

and requiring that \(K\) be constant, we substitute it by

\[
\frac{\partial S}{\partial t} = \frac{K^{(3)}}{\rho T} \nabla^6 T. \quad (195)
\]

also with \(K^{(3)} = K (\Delta x/\pi)^4\).

## E.2 Hyperviscosity

Viscosity has some caveats where subtleties apply. The difference is that the momentum flux due to viscosity is not proportional to the velocity gradient, but to the rate-of-strain tensor

\[
S_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \nabla \cdot \mathbf{u} \right), \quad (196)
\]

which only allows the viscous acceleration to be reduced to the simple formulation \(\nu \nabla^2 \mathbf{u}\) under the condition of incompressibility and constant dynamical viscosity \(\mu = \nu \rho\). Due to this, the general expression for conservative hyperviscosity involves more terms. In some cases, it is no great overhead, but for others, simpler formulations can be applied.

### E.2.1 Conservative case

In the general case, the viscous acceleration is

\[
f_{\text{visc}} = \rho^{-1} \nabla \cdot (2\rho \nu \mathbf{S}) \quad (197)
\]

So, for the hyperviscous force, we must replace the rate-of-strain tensor by a high order version

\[
f_{\text{visc}}^{(\text{hyper})} = \rho^{-1} \nabla \cdot (2\rho \nu_n \mathbf{S}^{(n)}) \quad (198)
\]

where the \(n\)th-order rate of strain tensor is

\[
\mathbf{S}^{(n)} = (-\nabla^2)^{n-1} \mathbf{S}. \quad (199)
\]
For the \( n = 3 \) case it is

\[
S_{ij}^{(3)} = \frac{1}{2} \left( \frac{\partial^5 u_j}{\partial x_i^5} + \frac{\partial^4 u_i}{\partial x_j^4} - \frac{1}{3} \frac{\partial^4}{\partial x_i^4} (\nabla \cdot \mathbf{u}) \right). \tag{200}
\]

Plugging it into Eq. (198), and assuming \( \mu_3 = \rho \nu_3 = \text{const} \)

\[
f^{(\text{hyper})}_{\text{visc}} = \nu_3 \left( \nabla^6 \mathbf{u} + \frac{1}{3} \nabla^4 (\nabla (\nabla \cdot \mathbf{u})) \right). \tag{201}
\]

For \( \nu_3 = \text{const} \), we have to take derivatives of density as well

\[
f^{(\text{hyper})}_{\text{visc}} = \nu_3 \left( \nabla^6 \mathbf{u} + \frac{1}{3} \nabla^4 (\nabla (\nabla \cdot \mathbf{u})) + 2S_{ij}^{(3)} \cdot \nabla \ln \rho \right) \tag{202}
\]

### E.2.2 Non-conservative cases

Equations (201) and (202) explicitly conserve linear and angular momentum. Although desirable properties, such expressions are cumbersome and numerically expensive, due to the fourth order derivatives of \( \nabla (\nabla \cdot \mathbf{u}) \).

This term, however, is only important when high compressibility is present (since it depends on the divergence of \( \mathbf{u} \)). In practice we drop this term and use a simple hyperviscosity

\[
f_{\text{visc}} = \begin{cases} 
\nu_3 \nabla^6 \mathbf{u} & \text{if } \mu = \text{const} \\
\nu_3 \left( \nabla^6 \mathbf{u} + 2S_{ij}^{(3)} \cdot \nabla \ln \rho \right) & \text{if } \nu = \text{const}
\end{cases} \tag{203}
\]

Notice that this can indeed be expressed as the divergence of a simple rate-of-strain tensor

\[
S_{ij}^{(3)} = \frac{\partial^5 u_i}{\partial x_j^5}, \tag{204}
\]

so it does conserve linear momentum. It does not, however, conserve angular momentum, since the symmetry of the rate-of-strain tensor was dropped. Thus, vorticity sinks and sources may be spuriously generated at the grid scale.

A symmetric tensor can be computed, that conserves angular momentum and can be easily implemented

\[
S_{ij} = \frac{1}{2} \left( \frac{\partial^5 u_i}{\partial x_j^5} + \frac{\partial^5 u_j}{\partial x_i^5} \right) \tag{205}
\]

This tensor, however, is not traceless, and therefore accurate only for weak compressibility. It should work well if the turbulence is subsonic. Major differences are not expected, since the spectral range in which hyperviscosity operates is very limited: as a numerical tool, only its performance as a high-frequency filter is needed. This also supports the usage of the highest order terms only, since these are the ones that provide quenching at high \( k \). Momentum conservation is a cheap bonus. Angular momentum conservation is perhaps playing it too safe, at great computational expense.
E.2.3 Choosing the coefficient

When changing the resolution, one wants to keep the grid Reynolds number, here defined as

$$\text{Re}_{\text{grid}} = \frac{u_{\text{rms}}}{(\nu_k k_{\text{Ny}}^{2n-1})} \quad (206)$$

approximately constant. Here, $k_{\text{Ny}} = \pi/\delta x$ is the Nyquist wavenumber and $\delta x$ is the mesh spacing. Thus, when doubling the number of meshpoints, we can decrease the viscosity by a factor of about $2^5 = 32$ (Haugen & Brandenburg 2004). This shows that hyperviscosity can allow a dramatic increase of the Reynolds number based on the scale of the box.

By choosing $\text{idiff='hyper3\_mesh'}$ in density_run_pars the hyperdiffusion for density is being set automatically in a mesh-independent way. A hyper-mesh Reynolds number of 30 corresponds to a coefficient $\text{diffrho\_hyper3\_mesh}=2$ if maxadvec is about 1, but in practice we need a bit more (5 is currently the default).

E.2.4 Turbulence with hyperviscosity

When comparing hyperviscous simulations with non-hyperviscous ones, it turns out that the Reynolds number at half the Nyquist frequency is usually in the range 5–7, i.e.

$$\text{Re}_{\text{half-grid}} = \frac{u_{\text{rms}}}{[\nu_n (k_{\text{Ny}}/2)^{2n-1}] \approx 5–7} \quad (207)$$

The following table gives some typical values used in simulations with forcing wavenumber $k_r = 1.5$ and a forcing amplitude of $f_0 = 0.02$. If hyperdiffusion $D_3$ is used in the continuity equation, the corresponding values are about 30 times smaller than those of $\nu_3$; see Table 14.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\nu_1$</th>
<th>$\nu_2$</th>
<th>$\nu_3$</th>
<th>$D_3$</th>
<th>$\nu_3^{\text{CFL}=0.4}$</th>
<th>$\nu_3^{\text{CFL}=0.9}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>$1 \times 10^{-2}$</td>
<td>$3 \times 10^{-4}$</td>
<td>$2 \times 10^{-5}$</td>
<td>$6 \times 10^{-7}$</td>
<td>$7 \times 10^{-4}$</td>
<td>$1 \times 10^{-4}$</td>
</tr>
<tr>
<td>32</td>
<td>$5 \times 10^{-3}$</td>
<td>$4 \times 10^{-5}$</td>
<td>$6 \times 10^{-7}$</td>
<td>$2 \times 10^{-8}$</td>
<td>$1 \times 10^{-6}$</td>
<td>$2 \times 10^{-5}$</td>
</tr>
<tr>
<td>64</td>
<td>$2 \times 10^{-3}$</td>
<td>$5 \times 10^{-6}$</td>
<td>$2 \times 10^{-8}$</td>
<td>$6 \times 10^{-10}$</td>
<td>$2 \times 10^{-7}$</td>
<td>$3 \times 10^{-6}$</td>
</tr>
<tr>
<td>128</td>
<td>$1 \times 10^{-3}$</td>
<td>$6 \times 10^{-7}$</td>
<td>$6 \times 10^{-10}$</td>
<td>$2 \times 10^{-11}$</td>
<td>$3 \times 10^{-8}$</td>
<td>$4 \times 10^{-7}$</td>
</tr>
<tr>
<td>256</td>
<td>$5 \times 10^{-4}$</td>
<td>$8 \times 10^{-8}$</td>
<td>$2 \times 10^{-11}$</td>
<td>$6 \times 10^{-13}$</td>
<td>$4 \times 10^{-9}$</td>
<td>$5 \times 10^{-8}$</td>
</tr>
<tr>
<td>512</td>
<td>$2 \times 10^{-4}$</td>
<td>$1 \times 10^{-8}$</td>
<td>$6 \times 10^{-13}$</td>
<td>$2 \times 10^{-14}$</td>
<td>$5 \times 10^{-10}$</td>
<td>$6 \times 10^{-9}$</td>
</tr>
<tr>
<td>1024</td>
<td>$1 \times 10^{-4}$</td>
<td>$1 \times 10^{-9}$</td>
<td>$2 \times 10^{-14}$</td>
<td>$6 \times 10^{-16}$</td>
<td>$6 \times 10^{-11}$</td>
<td>$8 \times 10^{-10}$</td>
</tr>
</tbody>
</table>

For comparison, we give in Table 14 estimates of the numerical 4th order hyperdiffusion resulting from a third order time step, for which we have

$$\kappa_{\text{CFL}}^{2} = \frac{1}{24} \frac{u_{\text{rms}}}{(C_{\text{CFL}} \delta x)^3} \quad (208)$$

where $C_{\text{CFL}}$ is the CFL parameter which is either 0.4 in the conservative case or 0.9 in the more progressive case.
E.3 Anisotropic hyperdissipation

As we want quenching primarily at the Nyquist frequency, hyperdissipation depends intrinsically on the resolution, according to Eq. (192). Because of this, isotropic hyperdissipation only gives equal quenching in all spatial directions if $\Delta x=\Delta y=\Delta z$, i.e., if the cells are cubic. For non-cubic cells, anisotropic dissipation is required as different directions may be better/worse sampled, thus needing less/more numerical smoothing. Such generalization is straightforward. For that, we replace Eq. (189) by

$$\mathbf{J} = \left( D_x \frac{\partial^5 \rho}{\partial x^5}, D_y \frac{\partial^5 \rho}{\partial y^5}, D_z \frac{\partial^5 \rho}{\partial z^5} \right),$$

so that different diffusion operates in different directions. Since $D_x, D_y$ and $D_z$ are constants, the divergence of this vector is

$$\nabla \cdot \mathbf{J} = D_x \frac{\partial^6 \rho}{\partial x^6} + D_y \frac{\partial^6 \rho}{\partial y^6} + D_z \frac{\partial^6 \rho}{\partial z^6}. \quad (210)$$

The formulation for resistivity and heat conductivity are strictly the same. For viscosity it also assumes the same form if we consider the simple non-conservative rate-of-strain tensor (204).

Mathematically, these operations can be written compactly by noticing that the coefficients in Eq. (210) transform like diagonal tensors $\chi^{(3)}_{ij} = \chi^{(3)}_k \delta_{ijk}$, where $\delta_{ijk}$ is the unit diagonal third order tensor, $\chi^{(3)}$ is the vector containing the dissipative coefficients (diffusion, viscosity, resistivity, or heat conductivity) in $x$, $y$, and $z$, and summation over repeated indices applies.

Therefore, for a scalar quantity $\psi$ (density, any of the three components of the velocity or magnetic potential), we can write

$$\frac{\partial \psi}{\partial t} = -\chi^{(3)}_{ij} \partial_i \partial_j^5 \psi = -\sum_q \chi^{(3)}_q \partial_{x^q}^6 \psi. \quad (211)$$

E.4 Hyperviscosity in Burgers shock

![Figure 20: Left: Burgers shock from teach/PencilCode/material/BurgersShock (in the teaching material) with $-20 \leq x \leq 20$, $n_x = 64$ mesh points, $u_z = \mp 1$ on the two ends, $\nu = 0.4$ and either $\nu_3 = 0$ (solid line) or $\nu_3 = 0.05$ (dashed line). Right: similar to the left hand side, but with $\nu = 0$ and $\nu_3 = 0.05$ (dashed line), compared with the case $\nu = 0.4$ and $\nu_3 = 0$ (solid line).]
Hyperviscosity has the unfortunate property of introducing (numerically stable) wiggles, even if one just adds a little bit of hyperviscosity to a run with normal viscosity; see left hand side of Fig. [20]. Running with just hyperviscosity give strong wiggles.
F Special techniques

F.1 After changing REAL_PRECISION

To continue working in double precision (REAL_PRECISION=double), you just say lread_from_other_prec=T in run_pars. Afterwards, remember to put lread_from_other_prec=F. If continuation is done in a new run directory, first execute start.csh there and then copy the files var.dat (and if present global.dat) from the old to the new directory, using pc-copyvar.

F.2 Remeshing (regridding)

[This should be written up in a better way and put somewhere else. But currently, remeshing is only available for the Pencil developers anyway.]

Suppose you have a directory run_64 with a 64³ run (running on N₀ = ny×nz=2×1 CPUs) that you want to continue from ‘VAR1’ at 128³ (on ny×nz=4×4 CPUs).

1. Get the remeshing stuff from repository:

```bash
unix> cd $PENCIL_HOME; cvs co -d remesh pencil-remesh
unix> setenv PATH \$\{$PATH\}:\$PENCIL_HOME/remesh/bin
```

2. Create another run directory with current ‘VAR1’ as ‘var.dat’ (remesh.csh so far only works with ‘var.dat’):

```bash
unix> cd run_64
run_64> pc_newrun ../tmp_64 or new tmp_64
run_64> mkdir -p ../tmp_64/data or (cd ../tmp_64/; ctmp)
run_64> (cd ../tmp_64/data ; mkproc-tree N₀)
run_64> restart-new-dir-VAR ../tmp_64 1
```

3. Create the new run directory (linking the executables with -s):

```bash
run_64> cd ../tmp_64
tmp_64> pc_newrun -s ../run_128 or new run_128
tmp_64> vi ../run_128/src/cparam.local
# set nxgrid=128, ncpus=16, nprocy=4
tmp_64> (cd ../run_128; ctmp; pc_setupsrc; make)
```

4. Setup and do remeshing

```bash
tmp_64> setup-remesh
tmp_64> vi remesh/common.local
# set muly=2, mulz=4, remesh_par=2
tmp_64> (cd remesh; make)
tmp_64> vi remesh.in
# Replace line by ../run_128
tmp_64> remesh.csh
# Answer ‘yes’
```
F.3 Restarting from a run with less physics

First, prepare a new run directory with the new physics included. By new physics, we mean that the new run wants to read in more fields (e.g. magnetic fields, if the old run didn’t have magnetic fields).

Example for test fields:

1. Prepare ‘src/cparam.local’
   Add the following 2 fragments into the ‘cparam.local’ file. The first piece comes in the beginning and the second in the end of the file.

   ```
   !*** AUTOMATIC CPARAM.INC GENERATION ****************************************
   ! Declare (for generation of cparam.inc) the number of f array
   ! variables and auxiliary variables added by this module
   ! Use MVAR to reserve the appropriate workspace for testfield_z.f90
   ! The MAUX number must be equally big and is used for uxb in the f-array.
   ! At the end of this file, njtest must be set such that 3*njtest=MVAR.
   !
   ! MVAR CONTRIBUTION 12
   ! MAUX CONTRIBUTION 12
   !***************************************************************************
   
   ! note that MVAR=MAUX=3*njtest must be obeyed
   
   integer, parameter :: njtest=4
   !
   ```

2. Prepare ‘src/Makefile.local’
   Add the line TESTFIELD=testfield_z to the file. Finally, compile the code.

3. Prepare restart data
   Go into data directory of the new run and prepare the directory tree using, e.g., the command `pc_mkproctree 16`. [In principle this could be automatized, but it isn’t yet.]

   Next, go into old run directory and say `restart-new-dir ../32c`, if ‘../32c’ is the name of the new run directory. This procedure copies all the files from the processor tree, plus files like ‘param.nml’, but this file may need some manual modification (or you could just use one from another runs with the new physics included, which is definitely the simplest!).

4. Prepare ‘run.in’
   Set lread_oldsnap_notestfield=T in `run_pars`. This means (as the name says) that one reads an old snapshot that did not have test fields in it.

   Reset boundary conditions and add stuff for the newly added fields, e.g., `bcz='a:s','a','a:s','a2','a','a','s','a','a','s','a','a','s','a','a','s'` in `run_pars`. If you don’t do this, you would effectively use periodic boundary conditions for the response to the test field, which is hardly correct once you set non-periodic boundary conditions for the other variables.
Add something like the following text fragments in the right position (after `grav_run_pars` and `magnetic_run_pars`, but before `shear_run_pars` and `viscosity_run_pars`.

```fortran
&testfield_run_pars
  !linit_aatext=T, daainit=100.
  itestfield='B11-B22'
  etatest=1e-4
  lsoca=F
/
```

Make sure that the data above are correct. You may want to change the values of `daainit` or `etatest`.

If you now run, and if you didn’t fix the file ‘data/param.nml’ you might get something like the following error:

```fortran
forrtl: severe (24): end-of-file during read, unit 1, file /wkspace/brandenb/pencil-code/axel/Poiseuille/2d/32c/data/param.nml
```

The reason for this is that it reads the old boundary data, but the corresponding array is too short. This includes stuff like `FBCX1` to `FBCX2_2`, but it is still not enough. Therefore it is easiest to use the ‘data/param.nml’ file from another run. You may well just use one from a single processor run with a different mesh. But remember to fix the ‘start.in’ file by correcting the boundary conditions and adding things like

```fortran
&testfield_init_pars
  luxb_as_aux=T
/
```

5. Prepare ‘print.in’, ‘xyaver.in’, and other obvious files such as ‘video.in’.

6. Once it works and is running, you must say *explicitly*

```fortran
&run_pars
  ...
  lread_oldsnap_notestfield=F
/
```

because otherwise you won’t read in your precious test field data next time you restart the code! (If you instead just remove this line, it will remember `lread_oldsnap_notestfield=T` from the previous run, which is of course wrong!)

Comments: For large magnetic Reynolds numbers the solutions to the test-field equations can show a linear instability, which can introduce large fluctuations. In that case it is best to reset the dependent test-field variable to zero in regular intervals. This is done by setting `linit_aatext=T`. Note that `daainit=100` sets the reset interval to 100.
G Runs and reference data

For reference purposes we document here some results obtained with various samples of the code.

G.1 Shock tests

G.1.1 Sod shock tube problem

Table 15: Combinations of $\rho$, $p$, and $s/c_p$ that are relevant for the Sod shock tube problem with constant temperature and different pressure ratios on the left and right hand sides of the shock.

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$p$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.3065</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>1.2275</td>
</tr>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>2.1486</td>
</tr>
</tbody>
</table>

G.1.2 Temperature jump

Table 16: Combinations of $c_s^2$, $p$, and $s/c_p$ that are relevant for the temperature shock problem with constant density, $\rho = 1$, and different temperature ratios on the left and right hand sides of the shock.

<table>
<thead>
<tr>
<th>$c_s^2$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.1</td>
<td>-2.3</td>
</tr>
<tr>
<td>0.01</td>
<td>-4.6</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>-9.2</td>
</tr>
</tbody>
</table>

G.2 Random forcing function

A solenoidal random forcing function $f$ can be invoked by putting `iforce='helical'` in the `forcing_run_pars` namelist. This produces the forcing function $f$ of the form

$$f(x,t) = \text{Re}\{N f_{k(t)} \exp[i k(t) \cdot x + i \phi(t)]\},$$

where $k(t) = (k_x, k_y, k_z)$ is a time dependent wave vector, $x = (x, y, z)$ is position, and $\phi(t)$ with $|\phi| < \pi$ is a random phase. On dimensional grounds the normalization factor is chosen to be $N = f_0 c_s (k c_s / \delta t)^{1/2}$, where $f_0$ is a nondimensional factor, $k = |k|$, and $\delta t$ is the length of the timestep. The $\delta t^{-1/2}$ dependence ensures that the forcing, which is delta-correlated in time, is properly normalized such that the correlator of the forcing function is independent of the length of the time step, $\delta t$. We focus on the case where $|k|$ is around 5, and select at each timestep randomly one of the 350 possible vectors in $4.5 < |k| < 5.5$. We force the system with eigenfunctions of the curl operator,

$$f_k = \frac{i k \times (k \times e) - \sigma |k|(k \times e)}{\sqrt{1 + \sigma^2 k^2} \sqrt{1 - (k \cdot e)^2/k^2}},$$

where $\sigma$, $\rho$, and $p$ are the density, pressure, and specific heat ratio of the gas, respectively.
where \( e \) is an arbitrary unit vector needed in order to generate a vector \( k \times e \) that is perpendicular to \( k \). Note that \( |f_k|^2 = 1 \) and, for \( \sigma = 1 \), \( ik \times f_k = |k| f_k \), so the helicity density of this forcing function satisfies

\[
f \cdot \nabla \times f = |k| f^2 > 0 \quad \text{(for } \sigma = 1 \text{)}
\]

at each point in space. We note that since the forcing function is like a delta-function in \( k \)-space, this means that all points of \( f \) are correlated at any instant in time, but are different at the next timestep. Thus, the forcing function is delta-correlated in time (but the velocity is not). This is the forcing function used in Brandenburg (2001), Brandenburg & Dobler (2001), and other papers in that series.

For \( \sigma = 0 \), the forcing function is completely nonhelical and reduces to the simpler form

\[
f_k = (k \times e) / \sqrt{k^2 - (k \cdot e)^2}.
\]

(215)

For \( 0 < |\sigma| < 1 \), the forcing function has fractional helicity, where \( \sigma \approx \langle \omega \cdot u \rangle / (k_t \langle u^2 \rangle) \); see Sect. 4.5 of Ref. [7]. In the code and the forcing_run.pars namelist, \( \sigma \) is called relhel.

In the code, the possible wavevectors are pre-calculated and stored in ‘k.dat’, which is being read in the beginning the code runs. To change the wavevectors (e.g. the typical value of \( k_t \), you need to change the file. In the directory ‘$PENCIL_HOME/samples/helical-MHDturb/K_VECTORS/’ there are several such files prepared:

\[
\text{k10.dat k1.dat k2.dat k3.dat k5.dat k15.dat k27.dat k30.dat k4.dat k8.dat}
\]

and more can be prepared in IDL with the procedure ‘$PENCIL_HOME/samples/helical-MHDturb/idl/generate_kvectors.pro’ There is also more help in the ‘README’ file in ‘helical-MHDturb’.

G.3 Three-layered convection model

In Sect. 3 we have shown the early stages of the convection model located in ‘samples/conv-slab’. To arrive at fully developed convection, you will need to run the

![Figure 21](image-url)
code for many more time steps. Figure 21 shows the vertical profiles of four basic quantities at time \( t = 50 \). Figure 22 shows the time evolution of rms and maximum velocity for the model for \( 0 < t < 50 \).

Figures 23 and 24 show vertical and horizontal sections for time \( t = 50 \).

To test magnetic helicity evolution in the shearing shear, we can choose as initial condition initaa='Beltrami-y' with amplaa=1. in magnetic_init_pars together with Sshear=-1. in shear_run_pars.

\[ G.4 \text{ Magnetic helicity in the shearing sheet} \]

**Figure 22**: Time evolution of rms and maximum velocity for the model ‘samples/conv-slab’. Similar plots can be produced by running the IDL script ‘ts.pro’.

**Figure 23**: Horizontal sections for \( t = 50 \). Top: velocity field. Bottom: entropy (color coded) and density (isocontours). Plots of this type can be produced by running the IDL script ‘hsections.pro’.
Thus, in ‘src/Makefile.local’ we just use

MAGNETIC=magnetic
HYDRO=nohydro
EOS=neos
DENSITY=nodensity
SHEAR=shear
VISCOSITY=noviscosity

and put

&init_pars
cvsid='$Id$',
/
&magnetic_init_pars
  initaa='Beltrami-y', amplaa=1.
/
&shear_init_pars

Figure 24: Vertical section $y = 0.516$ at $t = 50$. Top: velocity field. Bottom: entropy (color coded) and density (isocontours). Plots of this type can be produced by running the IDL scripts ‘vsections.pro’ or ‘vsections2.pro’.
Figure 25: Wind-up of the magnetic field leads to a linear increase in the rms magnetic field strength until Ohmic diffusion begins to become important (top panel). During this time the magnetic helicity is conserved. With Ohmic diffusion, the decay of $\langle A \cdot B \rangle$ is well described by integrating $-2\eta \langle J \cdot B \rangle$ (indicated by “from rhs” in the second panel).

/in 'start.in' and, for example,

```
/run_pars
cvsid='$Id$'
    nt=150000, it1=10, cdt=0.9, isave=50, itorder=3
dsnap=100. dvid=5., ialive=1
/
&magnetic_run_pars
    eta=0.
/
&shear_run_pars
    Sshear=-1.
/
```

in ‘run.in’. The output includes, among other things

```
arms(f10.7)
```
The result is shown in Figure 25, where we show the wind-up of the magnetic field, which leads to a linear increase in the rms magnetic field strength until Ohmic diffusion begins to become important (top panel). During this time the magnetic helicity is conserved. With Ohmic diffusion, the decay of \( \langle A \cdot B \rangle \) is well described by integrating \(-2\eta\langle J \cdot B \rangle\) (indicated by “from rhs” in the second panel).
H Numerical methods

H.1 Sixth-order spatial derivatives

Spectral methods are commonly used in almost all studies of ordinary (usually incompressible) turbulence. The use of this method is justified mainly by the high numerical accuracy of spectral schemes. Alternatively, one may use high order finite differences that are faster to compute and that can possess almost spectral accuracy. Nordlund & Stein [26] and Brandenburg et al. [12] use high order finite difference methods, for example fourth and sixth order compact schemes [24].

The sixth order first and second derivative schemes are given by

\[ f'_i = \frac{(-f_{i-3} + 9f_{i-2} - 45f_{i-1} + 45f_{i+1} - 9f_{i+2} + f_{i+3})}{(60 \delta x)}, \]

\[ f''_i = \frac{(2f_{i-3} - 27f_{i-2} + 270f_{i-1} - 490f_i + 270f_{i+1} - 27f_{i+2} + 2f_{i+3})}{(180 \delta x^2)}, \]

In Fig. 26 we plot effective wavenumbers for different schemes. Apart from the different explicit finite difference schemes given above, we also consider a compact scheme of 6th order, which can be written in the form

\[ \frac{1}{3} f'_{i-1} + f'_i + \frac{1}{3} f'_{i+1} = \frac{(f_{i-2} - 28f_{i-1} + 28f_{i+1} - f_{i+2})}{(36 \delta x)}, \]

for the first derivative, and

\[ \frac{2}{11} f''_{i-1} + f''_i + \frac{2}{11} f''_{i+1} = \frac{(3f_{i-2} + 48f_{i-1} - 102f_i + 48f_{i+1} + 3f_{i+2})}{(44 \delta x^2)}. \]

In the second panel of Fig. 26 we have plotted effective wavenumbers for second derivatives, which were calculated as

\[ (\cos kx)''_{\text{num}} = -k^2_{\text{eff}} \cos kx. \]

Of particular interest is the behavior of the second derivative at the Nyquist frequency, because that is relevant for damping zig-zag modes. For a second-order finite difference scheme \( k^2_{\text{eff}} \) is only 4, which is less than half the theoretical value of \( \pi^2 = 9.87 \). For fourth, sixth, and tenth order schemes this value is respectively 5.33, 6.04, 6.83. The last value is almost the same as for the 6th order compact scheme, which is 6.86. Significantly stronger damping at the Nyquist frequency can be obtained by using hyperviscosity, which Nordlund & Galsgaard (1995) treat as a quenching factor that diminishes the value of the second derivative for wavenumbers that are small compared with the Nyquist frequency. Accurate high order second derivatives (with no quenching factors) are important when calculating the current \( J \) in the Lorentz force \( J \times B \) from a vector potential \( A \) using \(-\mu_0 J = \nabla^2 A - \nabla \nabla \cdot A\). This will be important in the MHD calculations presented below.

---

18The fourth order compact scheme is really identical to calculating derivatives from a cubic spline, as was done in Ref. [26]. In the book by Collatz [13] the compact methods are also referred to as Hermitian methods or as Mehrstellen-Verfahren, because the derivative in one point is calculated using the derivatives in neighboring points.
Figure 26: Effective wave numbers for first and second derivatives using different schemes. Note that for the second derivatives the sixth order compact scheme is almost equivalent to the tenth order explicit scheme. For the first derivative the sixth order compact scheme is still superior to the tenth order explicit scheme.

H.2 Upwind derivatives to avoid ‘wiggles’

High-order centered-difference convection simulations often show “wiggles” (Nyquist zigzag pattern) in $\ln \rho$, which are apparently caused by a velocity profile where the velocity approaches zero on the boundary or inside the box. This causes the density profile to be squeezed into a boundary layer where eventually numerical resolution is insufficient and, for centered differences, a spurious Nyquist signal is generated that almost instantaneously propagates into much of the whole box.

Even if the stagnation point is on the boundary (and enforced by the boundary conditions), this behavior is hardly influenced by the boundary conditions on $\ln \rho$ at all. A solution, however, is to apply upwinded derivative operators. The simplest upwind derivative is a finite-difference derivative operator where the point furthest downwind is excluded from the stencil. For $u > 0$, that means that instead of

$$f_0' = \frac{-f_{-3} + 9f_{-2} - 45f_{-1}}{60 \delta x} + \frac{45f_1 - 9f_2 + f_3}{60} - \frac{\delta x^6 f^{(7)}}{140} = D^{(\text{cent,6})} + O (\delta x^6) ,$$

one takes

$$f_0' = \frac{-2f_{-3} + 15f_{-2} - 60f_{-1} + 20f_0 + 30f_1 - 3f_2}{60 \delta x} + \frac{\delta x^5 f^{(6)}}{60} = D^{(\text{up,5})} + O (\delta x^5) .$$

A fourth-order upwind scheme (excluding two downwind points) would be

$$f_0' = \frac{-f_{-3} + 6f_{-2} - 18f_{-1} + 10f_0 + 3f_1}{12 \delta x} - \frac{\delta x^4 f^{(5)}}{20} = D^{(\text{up,4})} + O (\delta x^4) .$$

$^{19}$A simple one-dimensional test profile would be $u(x) = 1 - x^2$ on $x \in [-1, 1]$, which will accumulate more and more mass near the right boundary $x = 1$.

In two- or three-dimensional settings, the presence of stagnation points of X-type leads to the same configuration, this time with the possibility of a steady state (i.e. without accumulation of mass). Such stagnation points occur e.g. at the top of an upwelling, or at the bottom of a downdraft in convection simulations, where locally $u_z \propto z_X - z$. 

The effect of upwinding is mostly to stop the Nyquist perturbations from spreading away from the boundary or stagnation point. With the fourth-order formula they actually hardly ever develop.

The difference between central and fifth-order upwind derivative is

\[
[D^{(up,5)} - D^{(cent,6)}]f_0 = \frac{-f_{-3} + 6f_{-2} - 15f_{-1} + 20f_0 - 15f_1 + 6f_2 - f_3}{60\delta x} = -\frac{\delta x^5}{60} f_0^{(6)}. \tag{224}
\]

In other words, 5th-order upwinding can be represented for any sign of \(u\) as hyperdiffusion (Dobler et al. 2006):

\[
-uf'(up,5th) = -uf'(centr,6th) + |u| \frac{\delta x^5}{60} f^{(6)}. \tag{225}
\]

The advantage over adopting constant hyperdiffusion is that in the upwinding scheme hyperdiffusion is only applied where it is needed (i.e. where advection is happening, hence the factor \(|u|\)).

The form (225) also suggests an easy way to get ‘stronger’ upwinding: Rather than excluding more points in the downwind direction, we can simply treat the weight of the hyperdiffusion term as a free parameter \(\alpha\):

\[
-uf'(up,5th,\alpha) = -uf'(centr,6th) + \alpha |u| \delta x^5 f^{(6)}. \tag{226}
\]

If \(\alpha\) is large, this may affect the time step, but for \(\alpha = 1/60\), the stability requirement for the hyperdiffusive term should always be weaker than the advective Courant criterion.

**H.3 The bidiagonal scheme for cross-derivatives**

The old default scheme used for cross-derivatives of type \(\partial^2 / (\partial x \partial y)\) used to read as follows:

\[
df=\text{fac}*( \& \\
270.* ( \ f(11+1:12+1,m+1,n,k) - f(11-1:12-1,m+1,n,k) \ & \\
+ f(11-1:12-1,m-1,n,k) - f(11+1:12+1,m-1,n,k)) \ & \\
- 27.* ( \ f(11+2:12+2,m+2,n,k) - f(11-2:12-2,m+2,n,k) \ & \\
+ f(11-2:12-2,m-2,n,k) - f(11+2:12+2,m-2,n,k)) \ & \\
+ 2.* ( \ f(11+3:12+3,m+3,n,k) - f(11-3:12-3,m+3,n,k) \ & \\
+ f(11-3:12-3,m-3,n,k) - f(11+3:12+3,m-3,n,k)) \ & 
\)
\]

and is “visualized” in the left part of Fig. 27. It is way more efficient than the straightforward approach of first taking the \(x\) and the \(y\) derivative consecutively. (shown in the right part of Fig. 27).

Off-diagonal terms enter not only the diffusion terms through \(\nabla \nabla \cdot u\) and \(\nabla \nabla \cdot A\) terms, but also through the \(J = \nabla \times \nabla \times A\) operator. The general formula is \(J_i = A_{j,ij} - A_{i,jj}\), so in 2-D in the \(xy\)-plane we have

\[
J_x = A_{x,xx} + A_{y,xy} - A_{x,xx} - A_{x,yy} = A_{y,xy} - A_{x,yy}, \tag{227}
\]
\[
J_y = A_{x,yy} + A_{y,yy} - A_{y,xx} - A_{y,yy} = A_{x,xy} - A_{y,xx}. \tag{228}
\]

Figure 28 shows how the two schemes perform for the propagation of Alfvén waves,
\[
\begin{array}{cccccccc}
-2 & 0 & 0 & 0 & 0 & 0 & +2 \\
0 & +27 & 0 & 0 & 0 & -27 & 0 \\
0 & 0 & -270 & 0 & +270 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & +270 & 0 & -270 & 0 & 0 \\
0 & -27 & 0 & 0 & 0 & +27 & 0 \\
+2 & 0 & 0 & 0 & 0 & 0 & -2 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
9 & -27 & 135 & 0 & -135 & 27 & -9 \\
-27 & 81 & -405 & 0 & 405 & -81 & 27 \\
135 & -405 & 2025 & 0 & -2025 & 405 & -135 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-135 & 405 & -2025 & 0 & 2025 & -405 & 135 \\
27 & -81 & 405 & 0 & -405 & 81 & -27 \\
-9 & 27 & -135 & 0 & 135 & -27 & 9 \\
\end{array}
\]

Figure 27: Weights of bidiagonal scheme (left) and consecutive scheme (right) for mixed derivatives $\partial^2/\partial x \partial y$. The numbers shown need to be divided by $720 \delta x \delta y$ for the bidiagonal and by $3600 \delta x \delta y$ for the consecutive scheme.

Figure 28: Alfvén wave for $B_0 = (1, 2, 0)$ and $k = (1, 2, 0)$ after $t = 2\pi$. The wave travels in the direction of $k$. Red symbols are for the bidiagonal scheme, black symbols show results obtained using the consecutive scheme. Already for $16^2$ mesh points there are no clear differences. For $8^2$ mesh points both schemes are equally imprecise regarding the phase error, but the amplitude error is still quite small (but this is mainly a property of the time stepping scheme).

\[\dot{u}_z = J_x B_{0y} - J_y B_{0x}, \]
\[\dot{A}_x = -u_z B_{0y}, \]
\[\dot{A}_y = +u_z B_{0x}. \]

The initial condition (as implemented in subroutine alfven_xy) is

\[u_z \sim \cos(k_x x + k_y y - \omega t), \]
\[A_x \sim +B_{0y} \sin(k_x x + k_y y - \omega t)/\omega, \]
\[A_y \sim -B_{0x} \sin(k_x x + k_y y - \omega t)/\omega, \]

where $\omega = k \cdot B_0$. The figure shows that there is no clear advantage of either scheme, so the code uses the more efficient bidiagonal one.

**H.4 The 2N-scheme for time-stepping**

For time stepping, higher-order schemes are necessary in order to reduce the amplitude and phase errors of the scheme and, to some extent, to allow longer time steps. Usually
such schemes require large amounts of memory. However, we here use the memory-effective $2N$-schemes that require only two sets of variables to be held in memory. Such schemes work for arbitrarily high order, although not all Runge-Kutta schemes can be written as $2N$-schemes \cite{29,28}. Consider the ordinary differential equation (ODE)

$$\dot{u} = F(u,t), \quad (235)$$

which can also be used as a prototype for a system of ODEs to be solved, like the ones obtained by spatial discretization of PDEs. The $2N$-schemes construct an approximation to the solution

$$u^{(n)} = u(t_n) \quad (236)$$

according to the iterative procedure

$$w_i = \alpha_i w_{i-1} + \delta t F(u_{i-1}, t_{i-1}), \quad (237)$$

$$u_i = u_{i-1} + \beta_i w_i. \quad (238)$$

For a three-step (RK3-2N) scheme we have $i = 1, ..., 3$. In order to advance the variable $u$ from $u^{(n)}$ at time $t^{(n)}$ to $u^{(n+1)}$ at time $t^{(n+1)} = t^{(n)} + \delta t$ we set in Eq. (238)

$$u_0 = u^{(n)} \quad \text{and, after the last step,} \quad u^{(n+1)} = u_3, \quad (239)$$

with $u_1$ and $u_2$ being intermediate steps. In order to be able to calculate the first step, $i = 1$, for which no $w_{i-1} \equiv w_0$ exists, we have to require $\alpha_1 = 0$. Thus, we are left with 5 unknowns, $\alpha_2$, $\alpha_3$, $\beta_1$, $\beta_2$, and $\beta_3$. Three conditions follow from the fact that the scheme be third order for linear equations, so we have to have two more conditions. One possibility is to choose the fractional times at which the right hand side is evaluated, for example $(0, 1/3, 2/3)$ or even $(0, 1/2, 1)$. Yet another possibility is to require that inhomogeneous equations of the form $\dot{u} = t^n$ with $n = 1$ and $2$ are solved exactly. The corresponding coefficients are listed in Table 17 and compared with those given by Williamson \cite{29}. In practice all of them are about equally good when it comes to real applications, although we found the first one in Table 17 (‘symmetric’) marginally better in some simple test problems where an analytic solution was known. In Ref. \cite{3} the accuracy of some nonlinear equations is tested.

Table 17: Coefficients for different $2N$-type third-order Runge-Kutta schemes. The coefficients $c_i$ (which are determined by the $\alpha_i$, $\beta_i$) give the time for each substep, $t_i = t_0 + c_i \delta t$

<table>
<thead>
<tr>
<th>scheme</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$\alpha_2$</th>
<th>$\alpha_3$</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
<th>$\beta_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>symmetric [predictor/corrector]</td>
<td>0</td>
<td>1/3</td>
<td>2/3</td>
<td>-2/3</td>
<td>-1</td>
<td>1/3</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>inhomogeneous</td>
<td>0</td>
<td>1/2</td>
<td>1</td>
<td>-1/4</td>
<td>-4/3</td>
<td>1/2</td>
<td>2/3</td>
<td>1/2</td>
</tr>
</tbody>
</table>

H.5 Diffusive error from the time-stepping

In many cases we use centered first derivatives for the advection operator, so the resulting discretization errors are only of dispersive nature (proportional to odd derivatives). A diffusive error can be obtained from the discretization error of the time-stepping scheme. For the RK3-2N scheme we have

$$\left(\frac{df}{dt}\right)_{\text{nth order}} = \left(\frac{df}{dt}\right)_{\text{exact}} + a_n \delta t^n \left(\frac{df^{n+1}}{dt^{n+1}}\right) + \ldots, \quad (240)$$
where \( a_n = 1/(n+1)! = 0.5 \). In particular, for \( n = 1 \) we have \( a_1 = 1/2 = 0.2 \) and for \( n = 3 \) we have \( a_3 = 1/24 \approx 0.04 \). The advection operator leads to a diffusive error equal to \( a_1 \delta t (u \cdot \nabla)^2 \) for \( n = 1 \) and a hyperdiffusive error equal to \( a_3 \delta t^3 (u \cdot \nabla)^4 \) for \( n = 3 \). Substituting \( \delta t = c_{CFL} \delta x / |u| \), where \( c_{CFL} \) is Courant-Friedrich-Lewy constant, we have a diffusive error \( \nu \nabla^2 \) with negative \( \nu = -a_1 c_{CFL} |u| \delta x \) for \( n = 1 \), and a hyperdiffusive error \( -\nu_{hyp} \nabla^4 \) with positive \( \nu_{hyp} = a_3 c_{CFL}^3 |u| \delta x^3 \) for \( n = 3 \). The fact that the hyperdiffusive error has a positive effective hyperdiffusivity is an important factor in the choice of this scheme.

To decide whether the effective hyperdiffusivity from the diffusive error is significant, we can compare with the error that would occur had we used a third-order upwinding scheme (Sect. [H.2]). In that case we would have an effective hyperdiffusive coefficient \( |u| \delta x/12 \) that is \( 1/(12a_3 c_{CFL}^3) \approx 5.8 \) times larger than that from the time stepping scheme. In this sense, the hyperdiffusive error can be regarded as small.

Since the hyperdiffusive error is proportional to \( -\nabla^4 \), we cannot directly compare with the physical diffusion which is proportional to \( \nabla^2 \). Therefore we define an effective viscosity as \( \nu_{eff} = \nu_{hyp} k_{Ny}^2 \) with \( k_{Ny} = \pi / \delta x \) being the Nyquist wavenumber of the mesh of the domain covered by \( N \) mesh points. We define Reynolds number based on the Nyquist wavenumber as \( Re_{Ny} = |u| / \nu_{eff} k_{Ny} \), and find \( Re = -24/(\pi c_{CFL})^3 \approx 2.3 \) for our favorite choice \( c_{CFL} = 0.7 \). Thus, at the scale of the mesh, the effective Reynolds number is comparable to the value often obtained in simulations. However, in turbulence simulations the viscous cutoff wavenumber is usually 5–10 times smaller than \( k_{Ny} \), so the relevant Reynolds number at the viscous scale is then another 2–3 orders of magnitude larger and does therefore not impose a constraint in view of the physical viscosity that is applied in such calculations.

### H.6 Ionization

The specific entropy of each particle species (neutral hydrogen, electrons, protons and neutral helium) may be written as

\[
\frac{s_i}{s_0} = x_i \left( \ln \left( \frac{1}{x_{tot}} \rho_i \left( \frac{T}{T_0} \right)^{3/2} \right) + \frac{5}{2} \right),
\]

where

\[
x_H = 1 - y, \quad x_e = x_p = y, \quad x_{tot} = 1 + y + x_{He}
\]

\[
s_0 = \frac{k_B}{\mu m_H}, \quad T_0 = \frac{\chi_H}{k_B},
\]

and

\[
\rho_i = \mu m_H \left( \frac{m_i \chi_H}{2 \pi \hbar^2} \right)^{3/2}.
\]

The specific entropy of mixing is

\[
\frac{s_{mix}}{s_0} = - \sum_i x_i \ln \frac{x_i}{x_{tot}}.
\]
Summing up everything, we get the total specific entropy

\[
\frac{s}{s_0} = \frac{1}{\sum_i \frac{s_i}{s_0}} + \frac{s_{\text{mix}}}{s_0} = \sum_i x_i \left( \ln \left[ \frac{1}{x_i \rho_i \left( \frac{T}{T_0} \right)^{3/2}} \right] + \frac{5}{2} \right) \tag{246}
\]

\[
= \sum_i x_i \ln \frac{\rho_i}{x_i} + x_{\text{tot}} \left( \ln \left[ \frac{1}{\rho \left( \frac{T}{T_0} \right)^{3/2}} \right] + \frac{5}{2} \right). \tag{247}
\]

Solving for \( T \) gives

\[
\frac{3}{2} \ln \frac{T}{T_0} = \frac{s}{s_0} + \sum_i x_i \ln \frac{x_i}{\rho_i} + \ln \rho - \frac{5}{2}. \tag{248}
\]

Using this expression and the constants defined above, we may obtain the ionization fraction \( y \) for given \( \ln \rho \) and \( s \) by finding the root of

\[
F = \ln \left[ \frac{\rho_e \left( \frac{T}{T_0} \right)^{3/2} \left( 1 - y \right) \rho_e}{y^2} \right] - \frac{T_0}{T}. \tag{249}
\]

The derivative with respect to \( y \) for Newton-Raphson is

\[
\frac{\partial F}{\partial y} = \left( \frac{3}{2} + \frac{T_0}{T} \right) \frac{\partial \ln T/T_0}{\partial y} - \frac{1}{1 - y} - \frac{2}{y}, \tag{250}
\]

where

\[
\frac{\partial \ln T/T_0}{\partial y} = \frac{\frac{2}{3} \left( \ln \rho_n/\rho_p - F - T_0/T \right) - 1}{1 + y + x_{\text{He}}}. \tag{251}
\]

In order to compute the pressure gradient in the momentum equation, the derivative of \( y \) with respect to \( \ln \rho \) and \( s \) needs to be known:

\[
\frac{\partial \ln P}{\partial \ln \rho} = \frac{1}{1 + y + x_{\text{He}}} \frac{\partial y}{\partial \ln \rho} + \frac{\partial \ln T}{\partial \ln \rho} + \frac{\partial \ln T}{\partial y} \frac{\partial y}{\partial \ln \rho} + 1, \tag{252}
\]

\[
\frac{\partial \ln P}{\partial s} = \frac{1}{1 + y + x_{\text{He}}} \frac{\partial y}{\partial s} + \frac{\partial \ln T}{\partial s} + \frac{\partial \ln T}{\partial y} \frac{\partial y}{\partial s}. \tag{253}
\]

Since \( F = 0 \) for all desired solutions \((y, \ln \rho, s)\) we also have

\[
dF = \frac{\partial F}{\partial \ln \rho} d \ln \rho + \frac{\partial F}{\partial s} ds + \frac{\partial F}{\partial y} dy = 0, \tag{254}
\]

and thus

\[
\frac{\partial y}{\partial \ln \rho} = \left( \frac{dy}{d \ln \rho} \right)_{ds=0} = \frac{-\partial F/\partial \ln \rho}{\partial F/\partial y} \tag{255}
\]

and

\[
\frac{\partial y}{\partial s} = \left( \frac{dy}{ds} \right)_{d \ln \rho=0} = \frac{-\partial F/\partial s}{\partial F/\partial y}. \tag{256}
\]

### H.7 Radiative transfer

#### H.7.1 Solving the radiative transfer equation

A formal solution of Eq. (76) is given by

\[
I(\tau) = I(\tau_0) e^{-(\tau - \tau_0)} + \int_{\tau_0}^{\tau} e^{-(\tau - \tau')} S(\tau') d\tau'. \tag{257}
\]
Using a generalization of the trapezoidal rule,
\[
\int_{\tau_0}^{\tau} e^{-(\tau-\tau')} f(\tau') d\tau' \approx \int_{\tau_0}^{\tau} e^{-(\tau-\tau')} \left[ f(\tau_0) + \frac{f(\tau) - f(\tau_0)}{\tau - \tau_0} (\tau - \tau_0) \right] d\tau'
\]
(258)
\[
\approx \left[ 1 - e^{-(\tau - \tau_0)} \right] f(\tau) - \frac{1 - e^{-(\tau - \tau_0)} (1 + \tau - \tau_0)}{\tau - \tau_0} [f(\tau) - f(\tau_0)] ,
\]
(259)
which is exact for linear functions \( S(\tau) \), we discretize this as
\[
I_{k+1} = I_k e^{-\delta\tau} + (1 - e^{-\delta\tau}) S_{k+1} - \frac{1 - e^{-\delta\tau} (1 + \delta\tau)}{\delta\tau} (S_{k+1} - S_k) ,
\]
(260)
\[
= I_k e^{-\delta\tau} + (1 - e^{-\delta\tau}) S_k + \frac{e^{-\delta\tau} - 1 + \delta\tau}{\delta\tau} (S_{k+1} - S_k) .
\]
(261)
Here the simplest way to calculate \( \delta\tau \) is
\[
\delta\tau = \frac{\chi_{k+1} + \chi_k}{2} \delta x ;
\]
(262)
more accurate alternatives are
\[
\delta\tau = \sqrt{\chi_k \chi_{k+1}} \delta x
\]
(263)
or
\[
\delta\tau = \frac{\chi_{k+1} - \chi_k}{\ln \frac{\chi_{k+1}}{\chi_k}} \delta x
\]
(264)

**H.7.2 Angular integration**

<table>
<thead>
<tr>
<th>Directions</th>
<th>( Y_{0}^{0} )</th>
<th>( Y_{0}^{2} )</th>
<th>( Y_{0}^{4} )</th>
<th>( Y_{0}^{\pm4} )</th>
<th>( Y_{6}^{0} )</th>
<th>( Y_{6}^{\pm4} )</th>
<th>( Y_{8}^{0} )</th>
<th>( Y_{8}^{\pm4} )</th>
<th>( Y_{8}^{\pm8} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coord.</td>
<td>6</td>
<td>0</td>
<td>21/2</td>
<td>3/4 ( \sqrt{70} )</td>
<td>3/4 ( \sqrt{13} )</td>
<td>3/8 ( \sqrt{182} )</td>
<td>99/32 ( \sqrt{17} )</td>
<td>3/32 ( \sqrt{2618} )</td>
<td>3 ( \sqrt{64} ) ( \sqrt{24310} )</td>
</tr>
<tr>
<td>Face diag.</td>
<td>12</td>
<td>0</td>
<td>-21/4</td>
<td>-3/8 ( \sqrt{70} )</td>
<td>-39/16 ( \sqrt{13} )</td>
<td>3/8 ( \sqrt{182} )</td>
<td>891/256 ( \sqrt{17} )</td>
<td>27/256 ( \sqrt{2618} )</td>
<td>27 ( \sqrt{512} ) ( \sqrt{24310} )</td>
</tr>
<tr>
<td>Space diag.</td>
<td>8</td>
<td>0</td>
<td>-28/3</td>
<td>-2/3 ( \sqrt{70} )</td>
<td>16/9 ( \sqrt{13} )</td>
<td>-8/9 ( \sqrt{182} )</td>
<td>11/9 ( \sqrt{17} )</td>
<td>1/27 ( \sqrt{2618} )</td>
<td>1 ( \sqrt{54} ) ( \sqrt{24310} )</td>
</tr>
<tr>
<td>Non-h. f.d.</td>
<td>8</td>
<td>2( \sqrt{5} )</td>
<td>-39/4</td>
<td>3/8 ( \sqrt{70} )</td>
<td>-19/16 ( \sqrt{13} )</td>
<td>27/32 ( \sqrt{182} )</td>
<td>611/256 ( \sqrt{17} )</td>
<td>51/256 ( \sqrt{2618} )</td>
<td>3 ( \sqrt{512} ) ( \sqrt{24310} )</td>
</tr>
<tr>
<td>Coord. ( x, y )</td>
<td>4</td>
<td>-2( \sqrt{5} )</td>
<td>9/2</td>
<td>-4/3 ( \sqrt{70} )</td>
<td>-5/4 ( \sqrt{13} )</td>
<td>-3/8 ( \sqrt{182} )</td>
<td>35/32 ( \sqrt{17} )</td>
<td>3/32 ( \sqrt{2618} )</td>
<td>3 ( \sqrt{64} ) ( \sqrt{24310} )</td>
</tr>
<tr>
<td>Coord. ( z )</td>
<td>2</td>
<td>2( \sqrt{5} )</td>
<td>6</td>
<td>0</td>
<td>2( \sqrt{13} )</td>
<td>0</td>
<td>2( \sqrt{17} )</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 18: Sums \( \sqrt{4\pi} Y_{l}^{m}(\theta, \phi) \) for special sets of directions. For all degrees and orders up to \( l = 8 \) not mentioned in this table, the sums are 0. The label 'Non-h. f-d.' stands for 'non-horizontal face-diagonals', i.e. the eight face diagonals that are not in the horizontal plane.

For angular integration over the full solid angle, we make the ansatz
\[
\int \frac{f(\theta, \phi) d\omega}{4\pi} = \sum_{i=1}^{N} w_i f(\theta_i, \phi_i) + R_N .
\]
(265)
Table [18] shows the sums $\sqrt{4\pi}Y^m_l(\theta_i, \phi_i)$ over special sets of directions $(\theta_i, \phi_i)$. Using these numbers and requiring that angular integration is exact for $l \leq l_{max}$, we find the following weights $w_i$ for different sets of directions (see also [1], §25.4.65):

1. Axes
   Coordinate axes: $1/6$
   $l_{max} = 3$

2. Face diagonals
   Face diagonals: $1/12$
   $l_{max} = 3$

3. Space diagonals
   Space diagonals: $1/8$
   $l_{max} = 3$

4. Axes + face diagonals
   Coordinate axes: $1/30$
   Face diagonals: $1/15$
   $l_{max} = 5$

5. Axes + space diagonals
   Coordinate axes: $1/15$
   Space diagonals: $3/40$
   $l_{max} = 5$

6. Face + space diagonals
   Face diagonals: $2/15$
   Space diagonals: $-3/40$
   $l_{max} = 5$

7. Axes, face + space diagonals
   Coordinate axes: $1/21$
   Face diagonals: $4/105$
   Space diagonals: $9/280$
   $l_{max} = 7$

8. Axes, non-horizontal face diagonals
   Coordinate axes $x, y$: $1/10$
   Coordinate axes $z$: $1/30$
   Non-hor. face diagonals: $1/15$
   $l_{max} = 3$

9. Axes, non-horizontal face diagonals, space diagonals
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate axes $x, y$</td>
<td>12/215</td>
</tr>
<tr>
<td>Coordinate axes $z$</td>
<td>10/129</td>
</tr>
<tr>
<td>Non-hor. face diagonals</td>
<td>-14/645</td>
</tr>
<tr>
<td>Space diagonals</td>
<td>171/1720</td>
</tr>
</tbody>
</table>

$l_{\text{max}} = 5$
I Switchable modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hydro.f90</td>
<td>This module takes care of most of the things related to velocity. Pressure, for example, is added in the energy (entropy) module.</td>
</tr>
<tr>
<td>gpu_astaroth.f90</td>
<td>This module contains GPU related types and functions to be used with ASTAROTH nucleus.</td>
</tr>
<tr>
<td>noentropy.f90</td>
<td>Calculates pressure gradient term for polytropic equation of state $p = \text{const} \rho^\Gamma$.</td>
</tr>
<tr>
<td>nogpu.f90</td>
<td>This module contains GPU related dummy types and functions.</td>
</tr>
<tr>
<td>nohydro.f90</td>
<td>no variable $u$: useful for kinematic dynamo runs.</td>
</tr>
<tr>
<td>nopower_spectrum.f90</td>
<td>reads in full snapshot and calculates power spectrum of $u$</td>
</tr>
<tr>
<td>noyinyang.f90</td>
<td>This module contains Yin-Yang related dummy types and functions.</td>
</tr>
<tr>
<td>noyinyang_mpi.f90</td>
<td>This module contains Yin-Yang related dummy types and functions.</td>
</tr>
<tr>
<td>power_spectrum.f90</td>
<td>reads in full snapshot and calculates power spectrum of $u$</td>
</tr>
<tr>
<td>timestep.f90</td>
<td>Runge-Kutta time advance, accurate to order $i\text{torder}$. At the moment, $i\text{torder}$ can be 1, 2, or 3.</td>
</tr>
<tr>
<td>timestep_strang.f90</td>
<td>Runge-Kutta time advance, accurate to order $i\text{torder}$. At the moment, $i\text{torder}$ can be 1, 2, or 3.</td>
</tr>
<tr>
<td>timestep_subcycle.f90</td>
<td>This is a highly specified timestep module currently only working together with the special module coronae.f90.</td>
</tr>
<tr>
<td>yinyang.f90</td>
<td>This module contains Yin-Yang related types and functions which are incompatible with FORTRAN 95.</td>
</tr>
<tr>
<td>yinyang_mpi.f90</td>
<td>This module contains Yin-Yang related types and functions which are incompatible with FORTRAN 95.</td>
</tr>
</tbody>
</table>

J Startup and run-time parameters

J.1 List of startup parameters for 'start.in'

The following table lists all (at the time of writing, September 2002) namelists used in 'start.in', with the corresponding parameters and their default values (in square brackets). Any variable referred to as a flag can be set to any nonzero value to switch the corresponding feature on. Not all parameters are used for a given scenario. This list is not necessarily up to date; also, in many cases it can only give an idea of the corresponding initial state; to get more insight and the latest set of parameters, you need to look at the code.

The value $\varepsilon$ corresponds to 5 times the smallest number larger than zero. For single precision, this is typically about $\varepsilon \approx 5 \times 1.2 \times 10^{-7} = 6 \times 10^{-7}$; for double precision, $\varepsilon \approx 10^{-15}$.

<table>
<thead>
<tr>
<th>Variable [default value]</th>
<th>Meaning</th>
</tr>
</thead>
</table>
Namelist *init_pars*

- **cvsid ["""]**: the svn identification string, which allows you to keep track of the version of `start.in`.
- **ip [14]**: (anti-)verbosity level: ip=1 produces lots of diagnostic output, ip=14 virtually none.
- **xyz0 [(-π, -π, -π)], Lxyz [(2π, 2π, 2π)], lperi [(T,T,T)]**: determine the geometry of the box. All three are vectors of the form (x-comp., y-comp., z-comp.); `xyz0` describes the left (lower) corner of the box, `Lxyz` the box size. `lperi` specifies whether a direction is considered periodic (in which case the last point is omitted) or not. In all cases, three ghost zones will be added.
- **lprocz_slowest [T]**: if set to F, the ordering of processor numbers is changed, so the z processors are now in the inner loop. Since nprocy=4 is optimal (see Sect. 5.20.2), you may want to put `lprocz_slowest=T` when `nygrid > nzgrid`.
- **lwrite_ic [F]**: if set T, the initial data are written into the file ‘VAR0’. This is generally useful, but doing this all the time uses up plenty of disk space.
- **lnowrite [F]**: if set T, all initialization files are written, including the `param.nml` file, except ‘`var.dat`’. This option allows you to use old filevar.dat files, but updates all other initialization files. This could be useful after having changed the code and, in particular, when the ‘`var.dat`’ files will be overwritten by ‘remesh.csh’.
- **lwrite_aux [F]**: if set T, auxiliary variables (those calculated at each step, but not evolved mathematically) to ‘var.dat’ after the evolved quantities.
- **lwrite_2d [F]**: if set T, only 2D-snapshots are written into VAR files in the case of 2D-runs with `nygrid = 1` or `nzgrid = 1`.
- **lread_oldsnap [F]**: if set T, the old snapshot will be read in before producing (overwriting) initial conditions. For example, if you just want to add a perturbation to the magnetic field, you’d give no initial condition for density and velocity (so you keep the data from a hopefully relaxed run), and just add whatever you need for the magnetic field. In this connection you may want to touch NOERASE, so as not to erase the previous data.
**J.1 Startup parameters for 'start.in'**

*lread_oldsnap_nomag [F]*

If set T, the old snapshot from a non-magnetic run will be read in before producing (overwriting) initial conditions. This allows one to let a hydrodynamic run relax before adding a magnetic field. However, for this to work one has to modify manually `data/param.nml` by adding an entry for `MAGNETIC_INIT_PARS` or `PSCALAR_INIT_PARS`. In addition, for idl to read correctly after the first restarted run, you must adjust the value of `mvar` in `data/dim.dat`.

*lread_oldsnap_nopscalar [F]*

If set T, the old snapshot from a run without passive scalar will be read in before producing (overwriting) initial conditions. This allows one to let a hydrodynamic run relax before adding a passive scalar.

*lshift_origin [F,F,F]*

If set T for any or some of the three directions, the mesh is shifted by 1/2 meshpoint in that or those directions so that the mesh goes through the origin.

*unit_system ['cgs']*

You can set this character string to `SI`, which means that you can give physical dimensions in SI units. The default is cgs units.

*unit_length [1]*

Allows you to set the unit length. Suppose you want the unit length to be 1 kpc, then you would say `unit_length='3e21'`. (Of course, politically correct would be to say `unit_system='SI'` in which case you say `unit_length='3e19'`.)

*unit_velocity [1]*

Example: if you want km/s you say `unit_length='1e5'`.

*unit_density [1]*

Example: if you want your unit density to be $10^{-24}$ g/cm$^3$ you say `unit_density='1e-24'`.

*unit_temperature [1]*

Example: `unit_temperature='1e6'` if you want mega-Kelvin.

*random_gen [system]*

Choose random number generator; currently valid choices are

- `system` (your compiler’s generator),
- `min_std` (the ‘minimal standard’ generator ran0() from ‘Numerical Recipes’),
- `nr_f90` (the Parker-Miller-Marsaglia generator ran() from ‘Numerical Recipes for F90’).

*bcx [(’p’, ’p’,...)],*

*bcy [(’p’, ’p’,...)],*

*bcz [(’p’, ’p’,...)]*

Boundary conditions. See Sect. 5.16 for a discussion of where and how to set these.

*pretend_InTT [F]*

Selects ln $T$ as fundamental thermodynamic variable in the entropy module.

---

Namelist `hydro_init_pars`
**inituu [‘zero’]**

Initialization of velocity. Currently valid choices are

- ‘zero’ ($u = 0$),
- ‘gaussian-noise’ (random, normally-distributed $u_x, u_z$),
- ‘gaussian-noise-x’ (random, normally-distributed $u_x$),
- ‘sound-wave’ (sound wave in $x$ direction),
- ‘shock-tube’ (polytropic standing shock),
- ‘bullets’ (blob-like velocity perturbations),
- ‘Alfven-circ-x’ (circularly polarized Alfven wave in $x$ direction),
- ‘const-ux’ (constant $x$-velocity),
- ‘tang-discont-z’ (tangential discontinuity: velocity is directed along $x$, jump is at $z = 0$),
- ‘Fourier-trunc’ (truncated Fourier series),
- ‘up-down’ (flow upward in one spot, downward in another; not solenoidal).

**ampluu [0.]**

Amplitude for some types of initial velocities.

**widthuu [0.1]**

Width for some types of initial velocities.

**urand [0.]**

Additional random perturbation of $u$. If $urand>0$, the perturbation is additive, $u_i \mapsto u_i + urand U_{[0.5,0.5]}$; if $urand<0$, it is multiplicative, $u_i \mapsto u_i \times urand U_{[0.5,0.5]}$; in both cases, $U_{[0.5,0.5]}$ is a uniformly distributed random variable on the interval $[-0.5,0.5]$.

**uu_left [0.], uu_right [0.]**

Needed for `inituu=’shock-tube’`.

---

**Namelist density_init_pars**
J.1 Startup parameters for 'start.in'

初始化密度参数。目前可用的选择包括
- ‘zero’ (\(\ln \rho = 0\)),
- ‘isothermal’ (等温层状传播),
- ‘polytropic_simple’ (等温层状传播),
- ‘hydrostatic-z-2’ (等温层状垂直层状传播,用于等温大气),
- ‘xjump’ (密度在x方向的宽度widthlnrho的跳),
- ‘rho-jump-z’ (密度在z方向的宽度widthlnrho的跳),
- ‘piecew-poly’ (分段等温层状垂直层状传播,用于太阳对流),
- ‘polytropic’ (等温层状传播),
- ‘sound-wave’ (声波),
- ‘shock-tube’ (等温层状驻波),
- ‘gaussian-noise’ (高斯分布,未相关噪声),
- ‘gaussian-noise’ (高斯分布,未相关噪声在x方向,但在y和z方向统一),
- ‘hydrostatic-r’ (等温层状径向密度层状传播,用于等温或等温大气),
- ‘sin-xy’ (x和y方向的正弦分布),
- ‘sin-xy-rho’ (x和y方向的正弦分布,但在\(\ln \rho\),不\(\ln \rho\)),
- ‘linear’ (线性分布),
- ‘planet’ (行星解;见[C.7])。

\[\gamma = \frac{5}{3} \quad \text{adiabatic index} \gamma = c_p/c_v.\]
\[cs0 [1] \quad \text{can be used to set the dimension of velocity; larger values can be used to decrease stratification.} \]
\[rho0 [1] \quad \text{reference values of sound speed and density, i.e. values at height zref.} \]
\[ampllnrho [0.], \quad \text{amplitude and width for some types of initial densities.} \]
\[widthlnrho [0.1] \quad \text{needed for initlnrho='shock-tube'.} \]
\[z1 [0.] \quad \text{sound speed at bottom and top. Needed for some types of stratification.} \]

**Namelist grav_init_pars**

\[zref [0.] \quad \text{reference height where in the initial stratification} \]
\[c_s^2 = c_s^2 \text{ and } \ln \rho = \ln \rho_0. \]
\[gravz [-1.] \quad \text{vertical gravity component } g_z. \]
\[grav_profile ['const'] \quad \text{constant gravity } g_z = gravz (grav_profile='const') \]
\[\text{gravity or linear profile } g_z = gravz \cdot z \text{ (grav_profile='linear', for accretion discs and similar).} \]
**z2 [1.]** specific to the solar convection case initlnrho='piecew-poly'. The stable layer is \( z_0 < z < z_1 \), the unstable layer \( z_1 < z < z_2 \), and the top (isothermal) layer is \( z_2 < z < z_{\text{top}} \).

**nu_epicycle [1.]** vertical epicyclic frequency; for accretion discs it should be equal to Omega, but not for galactic discs; see Eq. (125) in Sect. C.5.

**grav_amp [0.], grav_tilt [0.]** specific to the tilted gravity case (amplitude and angle wrt the vertical direction).

---

**Namelist entropy_init_pars**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>initss ['nothing']</strong></td>
<td>initialization of entropy. Currently valid choices are 'nothing' (leaves the initialization done in the density module unchanged), 'zero' (put ( s = 0 ) explicitly; this may overwrite the initialization done in the density module), 'isothermal' (isothermal stratification, ( T = \text{const} )), 'isobaric' (isobaric, ( p = \text{const} )), 'isentropic' (isentropic with superimposed hot [or cool] bubble), 'linprof' (linear entropy profile in ( z )), 'piecew-poly' (piecewise polytropic stratification for convection), 'polytropic' (polytropic stratification, polytropic exponent is ( m_{\text{poly}0} )), 'blob' (puts a gaussian blob in entropy for buoyancy experiments; see Ref. [5] for details) 'xjump' (jump in ( x ) direction), 'hor-tube' (horizontal flux tube in entropy, oriented in the ( y )-direction).</td>
</tr>
<tr>
<td><strong>pertss ['zero']</strong></td>
<td>additional perturbation to entropy. Currently valid choices are 'zero' (no perturbation) 'hexagonal' (hexagonal perturbation for convection).</td>
</tr>
<tr>
<td><strong>ampl_ss [0.], widthss [2( \varepsilon )]</strong></td>
<td>amplitude and width for some types of initial entropy.</td>
</tr>
<tr>
<td><strong>grads0 [0.]</strong></td>
<td>initial entropy gradient for initss=linprof.</td>
</tr>
<tr>
<td><strong>radius_ss [0.1]</strong></td>
<td>radius of bubble for initss=isentropic.</td>
</tr>
<tr>
<td><strong>mpoly0 [1.5], mpoly1 [1.5],</strong></td>
<td></td>
</tr>
</tbody>
</table>
Startup parameters for `start.in` 169

*mpoly2 [1.5]* specific to the solar convection case

`initss=piecew-poly`: polytropic indices of unstable (`mpoly0`), stable (`mpoly1`) and top layer (`mpoly2`). If the flag `isothtop` is set, the top layer is initialized to be isothermal, otherwise thermal (plus hydrostatic) equilibrium is assumed for all three layers, which results in a piecewise polytropic stratification.

*isothtop [0]* flag for isothermal top layer for `initss=piecew-poly`.

*khor_ss [1.]* horizontal wave number for `pertss=hexagonal`.

### Namelist `magnetic_init_pars`

`initaa ['zero']` initialization of magnetic field (vector potential). Currently valid choices are

- `'Alfven-x'` (Alfvén wave traveling in the $x$-direction; this also sets the velocity),
- `'Alfven-z'` (Alfvén wave traveling in the $z$-direction; this also sets the velocity),
- `'Alfvenz-rot'` (same as `'Alfven-z'`, but with rotation),
- `'Alfven-circ-x'` (circularly polarized Alfvén wave in $x$ direction),
- `'Beltrami-x'` ($x$-dependent Beltrami wave),
- `'Beltrami-y'` ($y$-dependent Beltrami wave),
- `'Beltrami-z'` ($z$-dependent Beltrami wave),
- `'Bz(x)'` ($B_z \propto \cos(kx)$),
- `'crazy'` (for testing purposes),
- `'diffrot'` ([needs to be documented]),
- `'fluxrings'` (two interlocked magnetic fluxrings; see §[C.4]),
- `'gaussian-noise'` (white noise),
- `'halfcos-Bx'` ([needs to be documented]),
- `'hor-tube'` (horizontal flux tube in $B$, oriented in the $y$-direction).
- `'hor-fluxlayer'` (horizontal flux layer),
- `'mag-support'` ([needs to be documented]),
- `'mode'` ([needs to be documented]),
- `'modeb'` ([needs to be documented]),
- `'propto-ux'` ([needs to be documented]),
- `'propto-uy'` ([needs to be documented]),
- `'propto-uz'` ([needs to be documented]),
- `'sinxsinz'` ($\sin x \sin z$),
- `'uniform-Bx'` (uniform field in $x$ direction),
- `'uniform-By'` (uniform field in $y$ direction),
- `'uniform-Bz'` (uniform field in $z$ direction),
- `'zero'` (zero field),
**initaa2 ['zero']**

additional perturbation of magnetic field. Currently valid choices are
- ‘zero’ (zero perturbation),
- ‘Beltrami-x’ (x-dependent Beltrami wave),
- ‘Beltrami-y’ (y-dependent Beltrami wave),
- ‘Beltrami-z’ (z-dependent Beltrami wave).

**amplaa [0.]**

amplitude for some types of initial magnetic fields.

**amplaa2 [0.]**

amplitude for some types of magnetic field perturbation.

**fring{1,2} [0.], Iring{1,2} [0.], Rring{1,2} [1.], wr{1,2} [0.3]**

flux, current, outer and inner radius of flux ring 1/2; see Sect. [C.4]

**radius [0.1]**

used by some initial fields.

**epsilonaa [10^{−2}]**

used by some initial fields.

**widthaa [0.5]**

used by some initial fields.

**z0aa [0.]**

used by some initial fields.

**kx_aa [1.], ky_aa [1.], kz_aa [1.]**

wavenumbers used by some initial fields.

**lpres equl [F]**

flag for pressure equilibrium (can be used in connection with all initial fields)

---

**Namelist pscalar_init_pars**

**initlncc ['zero']**

initialization of passive scalar (concentration per unit mass, c). Currently valid choices (for ln c) are
- ‘zero’ (ln c = 0.),
- ‘gaussian-noise’ (white noise),
- ‘wave-x’ (wave in x direction),
- ‘wave-y’ (wave in y direction),
- ‘wave-z’ (wave in z direction),
- ‘tang-discont-z’ (Kelvin-Helmholtz instability),
- ‘hor-tube’ (horizontal tube in concentration; used as a marker for magnetic flux tubes).

**initlncc2 ['zero']**

additional perturbation of passive scalar concentration c. Currently valid choices are
- ‘zero’ (\( \delta \ln c = 0. \)),
- ‘wave-x’ (add x-directed wave to ln c).

**ampllncc [0.1]**

amplitude for some types of initial concentration.

**ampllncc2 [0.]**

amplitude for some types of concentration perturbation.

**kx_lncc [1.], ky_lncc [1.], kz_lncc [1.]**

wave numbers for some types of initial concentration.

---

**Namelist shear_init_pars**
J.2 List of runtime parameters for ‘run.in’

The following table lists all (at the time of writing, September 2002) namelists used in file ‘run.in’, with the corresponding parameters and their default values (in square brackets). Default values marked as [start] are taken from ‘start.in’. Any variable referred to as a flag can be set to any nonzero value to switch the corresponding feature on. Not all parameters are used for a given scenario. This list is not necessarily up to date; also, in many cases it can only give an idea of the corresponding setup; to get more insight and the latest set of parameters, you need to look at the code.

Once you have changed any of the ‘*.in’ files, you may want to first execute the command pc_configtest in order to test the correctness of these configuration files, before you apply them in an active simulation run.

<table>
<thead>
<tr>
<th>Variable [default value]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cvsid ['']</code></td>
<td>svn identification string, which allows you to keep track of the version of ‘run.in’.</td>
</tr>
<tr>
<td><code>nt [0]</code></td>
<td>number of time steps to run. This number can be increased or decreased during the run by touch RELOAD.</td>
</tr>
</tbody>
</table>
**it1**[10]  
write diagnostic output every **it1** time steps (see Sect. 5.5).

**it1d**[**it1**]  
write averages every **it1d** time steps (see Sect. 5.8.1).  
**it1d** has to be greater than or equal to **it1**.

**cdt**[0.4]  
Courant coefficient for advective time step; see § 5.15.

**cdtv**[0.08]  
Courant coefficient for diffusive time step; see § 5.15.

**dt**[0.]  
time step; if ≠ 0., this overwrites the Courant time step. See § 5.15 for a discussion of the latter.

**dtmin**[10⁻⁶]  
abort if time step δt < δt_{min}.

**tmax**[10⁻³³]  
don’t run time steps beyond this time. Useful if you want to run for a given amount of time, but don’t know the necessary number of time steps.

**isave**[100]  
update current snapshot ‘var.dat’ every **isave** time steps.

**itorder**[3]  
order of time step (1 for Euler; 2 for 3nd-order, 3 for 3rd-order Runge–Kutta).

**dsnap**[100.]  
save permanent snapshot every **dsnap** time units to files ‘VARN’, where N counts from N = 1 upward. (This information is stored in the file ‘data/tsnap.dat’; see the module wsnaps.f90, which in turn uses the subroutines out1 and out2).

**dvid**[100.]  
write two-dimensional sections for generation of videos every **dvid** time units (not timesteps; see the subroutines out1 and out2 in the code).

**iwig**[0]  
if ≠ 0, apply a Nyquist filter (a filter eliminating any signal at the Nyquist frequency, but affecting large scales as little as possible) every **iwig** time steps to logarithmic density (sometimes necessary with convection simulations).

**ix**[−1], **iy**[−1], **iz**[−1], **iz2**[−1]  
position of slice planes for video files. Any negative value of some of these variables will be overwritten according to the value of **slice_position**. See § 5.7 for details.

**slice_position**[‘p’]  
symbolic specification of slice position. Currently valid choices are  
‘p’ (periphery of the box)  
‘m’ (middle of the box)  
‘e’ (equator for half-sphere calculations, i.e. x, y centered, z bottom)

These settings are overridden by explicitly setting **ix**, **iy**, **iz** or **iz2**. See § 5.7 for details.

**zbot_slice**[value]  
z position of slice xy-plane. The value can be any float number inside the z domain. These settings are overridden by explicitly setting **ix**, **iy**, **iz** or **iz2**. Saved as slice with the suffix xy. See § 5.7 for details.

**ztop_slice**[value]  
z position of slice xy-plane. The value can be any float number inside the z domain. These settings are overridden by explicitly setting **ix**, **iy**, **iz** or **iz2**. Saved as slice with the suffix xy2. See § 5.7 for details.
tavg [0] averaging time $\tau_{\text{avg}}$ for time averages (if $\neq 0$); at the same time, time interval for writing time averages. See §5.8.4 for details.

idx_tavg [(0, 0, ..., 0)] indices of variables to time-average. See §5.8.4 for details.

d2davg [100.] time interval for azimuthal and $z$-averages, i.e. the averages that produce 2d data. See §5.8.3 for details.

ialive [0] if $\neq 0$, each processor writes the current time step to ‘alive.info’ every ialive time steps. This provides the best test that the job is still alive. (This can be used to find out which node has crashed if there is a problem and the run is hanging.)

bcx [('p', 'p', ...)],
bcy [('p', 'p', ...)],
bcz [('p', 'p', ...)] boundary conditions. See Sect. 5.16 for a discussion of where and how to set these.

random_gen [start] see start parameters, p. 165

lwrite_aux [start] if set T, auxiliary variables (those calculated at each step, but not evolved mathematically) to ‘var.dat’ and ‘VAR’ files after the evolved quantities.

---

Namelist hydro_run_pars

Omega [0.] magnitude of angular velocity for Coriolis force (note: the centrifugal force is turned off by default, unless lcentrifugal_force=T is set).  

theta [0.] direction of angular velocity in degrees ($\theta = 0$ for $z$-direction, $\theta = 90$ for the negative $x$-direction, corresponding to a box located at the equator of a rotating sphere. Thus, e.g., $\theta = 60$ corresponds to $30^\circ$ latitude. (Note: prior to April 29, 2007, there was a minus sign in the definition of $\theta$.)

transient [0.] initial time span for which to do something special (transient). Currently just used to smoothly switch on heating [Should be in run_pars, rather than here].

dampu [0.],
tdamp [0.],
ldamp_fade [F] damp motions during the initial time interval $0 < t < t_{\text{damp}}$ with a damping term $-\text{dampu}(u)$. If ldamp_fade is set, smoothly reduce damping to zero over the second half of the time interval tdamp. Initial velocity damping is useful for situations where initial conditions are far from equilibrium.

dampuint [0.],
dampuext [0.] weighting of damping external to spherical region (see wdamp, damp below).

dampuext [0.],
rampuext [impossible] weighting of damping in internal spherical region (see wdamp, damp below).

rdampint [0.],
rdampext [impossible] radius of internal damping region

radius of external damping region, used in place of former variable rdamp
wdamp [0.2], \( \text{permanent damping of motions in } |x| < r_{\text{dampint}} \) with damping term \(-damp_{\text{int}} u \chi(r-r_{\text{dampint}})\) or \(|x| > r_{\text{dampext}} \) with damping term \(-damp_{\text{ext}} u \chi(r-r_{\text{dampext}})\), where \(\chi(\cdot)\) is a smooth profile of width \(wdamp\).

ampl_forc [0.], amplitude of the u\(x\)-forcing or u\(y\)-forcing on the vertical boundaries that is of the form \(u_x(t) = ampl_{\text{forc}} \sin(k_{\text{forc}} x) \cos(w_{\text{forc}} t)\) [must be used in connection with bcx='g' or bcz='g' and force_lower_bound='vel_time' or force_upper_bound='vel_time']

k_forc [0.], corresponding horizontal wavenumber
w_forc [0.], corresponding frequency

<table>
<thead>
<tr>
<th>Namelist density_run_pars</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs0 [start],</td>
</tr>
<tr>
<td>rho0 [start],</td>
</tr>
<tr>
<td>gamma [start]</td>
</tr>
<tr>
<td>cdiffrho [0.]</td>
</tr>
<tr>
<td>Coefficient for mass diffusion (diffusion term will be (C_{\text{diffrho}} \delta x c_{s0}))</td>
</tr>
<tr>
<td>cs2bot [start],</td>
</tr>
<tr>
<td>cs2top [start]</td>
</tr>
<tr>
<td>squared sound speed at bottom and top for boundary condition 'c2'.</td>
</tr>
<tr>
<td>lupw_lnrho [.false.]</td>
</tr>
<tr>
<td>use 5th-order upwind derivative operator for the advection term (u \cdot \nabla \ln \rho) to avoid spurious Nyquist signal ('wiggles'); see §H.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Namelist entropy_run_pars</th>
</tr>
</thead>
<tbody>
<tr>
<td>hcond0 [0.],</td>
</tr>
<tr>
<td>hcond1 [start],</td>
</tr>
<tr>
<td>hcond2 [start]</td>
</tr>
<tr>
<td>specific to the solar convection case initss=piecew-poly: heat conductivities (K) in the individual layers. (hcond0) is the value (K_{\text{unst}}) in the unstable layer, (hcond1) is the ratio (K_{\text{stab}}/K_{\text{unst}}) for the stable layer, and (hcond2) is the ratio (K_{\text{top}}/K_{\text{unst}}) for the top layer. The function (K(z)) is not discontinuous, as the transition between the different values is smoothed over the width (widthss). If (hcond1) or (hcond2) are not set, they are calculated according to the polytropic indices of the initial profile, (K \propto m+1).</td>
</tr>
<tr>
<td>iheatcond ['K-const']</td>
</tr>
<tr>
<td>select type of heat conduction. Currently valid choices are 'K-const' (constant heat conductivity), 'K-profile' (vertical or radial profile), 'chi-const' (constant thermal diffusivity), 'magnetic' (heat conduction by electrons in magnetic field – currently still experimental).</td>
</tr>
</tbody>
</table>
lcalc_heatcond_constchi [F] flag for assuming thermal diffusivity \(\chi = K/(c_p \rho) = \text{const}\), rather than \(K = \text{const}\) (which is the default). This is currently only correct with 'noionization.f90'. Superseded by iheatcond.
chi [0.]
widthss [start]
isothtop [start]
luminosity [0.],
wheat [0.1],
cooltype ['Temp']
cool [0.],
wcool [0.1]
rcool [1.]
Fbot [start]
chi_t [0.]
lupw_ss [.false.]
tauheat_buffer [0.]
zheat_buffer [0.]
dheat_buffer1 [0.]
TTheat_buffer [0.]

value of $\chi$ when lcalc_heatcond_constchi=T.
width of transition region between layers. See start parameters, p. [169]
flag for isothermal top layer for solar convection case. See start parameters, p. [169]
strength and width of heating region.
type of cooling; currently only implemented for spherical geometry. Currently valid choices are
‘Temp’, ‘cs2’ (cool temperature toward $c_s^2 = cs2cool$) with a cooling term

$$-C = -c_{cool} \frac{c_s^2 - c_{cool}^2}{c_{cool}^2}$$

‘Temp-rho’, ‘cs2-rho’ (cool temperature toward $c_s^2 = cs2cool$) with a cooling term

$$-C = -c_{cool} \rho \frac{c_s^2 - c_{cool}^2}{c_{cool}^2}$$

— this avoids numerical instabilities in low-density regions [currently, the cooling coefficient $c_{cool} = cool$ is not taken into account when the time step is calculated]

‘entropy’ (cool entropy toward 0.).

strength $c_{cool}$ and smoothing width of cooling region.
radius of cooling region: cool for $|x| \geq rcool$.
heat flux for bottom boundary condition ‘c1’. For polytropic atmospheres, if Fbot is not set, it will be calculated from the value of hcond0 in ‘start.x’, provided the entropy boundary condition is set to ‘c1’.
entropy diffusion coefficient for diffusive term $\partial s/\partial t = \ldots + \chi_t \nabla^2 s$ in the entropy equation, that can represent some kind of turbulent (sub-grid) mixing. It is probably a bad idea to combine this with heat conduction $hcond0 \neq 0$.
use 5th-order upwind derivative operator for the advection term $u \cdot \nabla s$ to avoid spurious Nyquist signal (‘wiggles’); see §H.2.
time scale for heating to target temperature ($=TTheat_buffer$); zero disables the buffer zone.
z coordinate of the thermal buffer zone. Buffering is active in $|z| > TTheat_buffer$.
Inverse thickness of transition to buffered layer.
target temperature in thermal buffer zone (z direction only).
lhcond_global [F] flag for calculating the heat conductivity $K$ (and also $\nabla \log K$) globally using the global arrays facility. Only valid when iheatcond='K-profile'.

Namelist magnetic_run_pars

B_ext [(0., 0., 0.)] uniform background magnetic field (for fully periodic boundary conditions, uniform fields need to be explicitly added, since otherwise the vector potential $A$ has a linear $x$-dependence which is incompatible with periodicity).

lignore_Bext_in_b2 [F] add uniform background magnetic field when computing $b^2$ pencils

or luse_Bext_in_b2 [T] magnetic diffusivity $\eta = 1/(\mu_0 \sigma)$, where $\sigma$ is the electric conductivity.

eta [0.] used to add extra diffusivity in a halo region.

eta_out [0.] used to add extra diffusivity inside sphere of radius $r_{int}$.

eta_ext [0.] used to add extra diffusivity outside sphere of radius $r_{ext}$.

kinflow [''] set type of flow fixed with 'nohydro'. Currently the only recognized value is 'ABC' for an $ABC$ flow; all other values lead to $u = 0$.

kx [1.,]
ky [1.,]
ksz [1.] wave numbers for $ABC$ flow.

ABC_A [1.,]
ABC_B [1.,]
ABC_C [1.] amplitudes $A$, $B$ and $C$ for $ABC$ flow.

Namelist pscalar_run_pars

pscalar_diff [0.] diffusion for passive scalar concentration $c$.

tensor_pscalar_diff [0.] coefficient for non-isotropic diffusion of passive scalar.

reinitialize_Lncc [F] reinitialize the passive scalar to the value of $cc_{const}$ in start.in at next run

Namelist forcing_run_pars

iforce [2] select form of forcing in the equation of motion; currently valid choices are

'zero' (no forcing),

'irrotational' (irrotational forcing),

'helical' (helical forcing),

'fountain' (forcing of "fountain flow"; see Ref. [11]),

'horizontal-shear' (forcing localized horizontal sinusoidal shear).

'variable_gravz' (time-dependent vertical gravity for forcing internal waves),

iforce2 [0] select form of additional forcing in the equation of motion; valid choices are as for iforce.
**Runtime parameters for ‘run.in’**

- **force [0.]**  
  Amplitude of forcing.

- **relhel [1.]**  
  Helicity of forcing. The parameter `relhel` corresponds to $\sigma$ introduced in Sect. G.2 ($\sigma = \pm 1$ corresponds to maximum helicity of either sign).

- **height_ff [0.]**  
  Multiply forcing by $z$-dependent profile of width `height_ff` (if $\neq 0$).

- **r_ff [0.]**  
  If $\neq 0$, multiply forcing by spherical cutoff profile (of radius `r_ff`) and flip signs of helicity at equatorial plane.

- **width_ff [0.5]**  
  Width of vertical and radial profiles for modifying forcing.

- **kfountain [5]**  
  Horizontal wavenumber of the fountain flow.

- **fountain [1.]**  
  Amplitude of the fountain flow.

- **omega_ff [1.]**  
  Frequency of the cos or sin forcing [e.g. $\cos(omega_ff*t)$].

- **ampl_ff [1.]**  
  Amplitude of forcing in front of cos or sin [e.g. $\text{ampl}_ff\cos(omega_ff*t)$].

---

### Namelist **grav_run_pars**

- **zref [start], gravz [start], grav_profile [start]**  
  See p. 167.

- **nu_epicycle [start]**  
  See Eq. (125) in Sect. C.5.

---

### Namelist **viscosity_run_pars**

- **nu [0.]**  
  Kinematic viscosity.

- **nu_hyper2 [0.]**  
  Kinematic hyperviscosity (with $\nabla^4 u$).

- **nu_hyper3 [0.]**  
  Kinematic hyperviscosity (with $\nabla^6 u$).

- **zeta [0.]**  
  Bulk viscosity.

- **ivisc [’nu-const’]**  
  Select form of viscous term (see §6.2); currently valid choices are
  - **’nu-const’**  
    - Viscous force for $\nu = \text{const}$, $F_{visc} = \nu(\nabla^2 u + \frac{1}{3} \nabla \nabla \cdot u + 25 \cdot \nabla \ln \rho)$
  - **’rho_nu-const’**  
    - Viscous force for $\mu = \rho \nu = \text{const}$, $F_{visc} = (\mu/\rho)(\nabla^2 u + \frac{1}{3} \nabla \nabla \cdot u)$. With this option, the input parameter `nu` actually sets the value of $\mu/\rho_0$ ($\rho_0$ is another input parameter, see pp. 167 and 174)
  - **’simplified’**  
    - Simplified viscous force $F_{visc} = \nu \nabla^2 u$

---

### Namelist **shear_run_pars**

- **qshear [start]**  
  See p. 171.

---

### Namelist **particles_run_pars**

- **ldragforce_dust_par [F]**  
  Drag force on particles

- **ldragforce_gas_par [F]**  
  Particle-gas friction force

- **ldraglaw steadystate [F]**  
  Particle forces only with $\frac{1}{2} \Delta v$

- **lpscalar_sink [F]**  
  Particles consume passive scalar

- **pscalar_sink_rate [0]**  
  Volumetric pscalar consumption rate
The following table lists all possible inputs to the file `print.in’ that are documented.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>it</code></td>
<td>number of time step (since beginning of job only)</td>
</tr>
<tr>
<td><code>t</code></td>
<td>time ( t ) (since start.csh)</td>
</tr>
<tr>
<td><code>dt</code></td>
<td>time step ( \delta t )</td>
</tr>
<tr>
<td><code>walltime</code></td>
<td>wall clock time since start of run.x, in seconds</td>
</tr>
<tr>
<td><code>Rmesh</code></td>
<td>( R_{\text{mesh}} )</td>
</tr>
<tr>
<td><code>Rmesh3</code></td>
<td>( R_{\text{mesh}}^{(3)} )</td>
</tr>
<tr>
<td><code>maxadvec</code></td>
<td>maxadvec</td>
</tr>
<tr>
<td><code>u2tm</code></td>
<td>( \left\langle u(t) \cdot \int_0^t u(t')dt' \right\rangle )</td>
</tr>
<tr>
<td><code>uotm</code></td>
<td>( \left\langle u(t) \cdot \int_0^t \omega(t')dt' \right\rangle )</td>
</tr>
<tr>
<td><code>outm</code></td>
<td>( \left\langle \omega(t) \cdot \int_0^t u(t')dt' \right\rangle )</td>
</tr>
<tr>
<td><code>fkinzm</code></td>
<td>( \left\langle \frac{1}{2} u^2 u_z \right\rangle )</td>
</tr>
<tr>
<td><code>u2m</code></td>
<td>( \left\langle u^2 \right\rangle )</td>
</tr>
<tr>
<td><code>uxpt</code></td>
<td>( u_x(x_1, y_1, z_1, t) )</td>
</tr>
<tr>
<td><code>uypt</code></td>
<td>( u_y(x_1, y_1, z_1, t) )</td>
</tr>
<tr>
<td><code>uzpt</code></td>
<td>( u_z(x_1, y_1, z_1, t) )</td>
</tr>
<tr>
<td><code>uxp2</code></td>
<td>( u_x(x_2, y_2, z_2, t) )</td>
</tr>
<tr>
<td><code>uyp2</code></td>
<td>( u_y(x_2, y_2, z_2, t) )</td>
</tr>
<tr>
<td><code>uzp2</code></td>
<td>( u_z(x_2, y_2, z_2, t) )</td>
</tr>
<tr>
<td><code>urms</code></td>
<td>( \left\langle u^2 \right\rangle^{1/2} )</td>
</tr>
<tr>
<td><code>urmsx</code></td>
<td>( \left\langle u_x^2 \right\rangle^{1/2} ) for the hydro_xaver_range</td>
</tr>
<tr>
<td><code>urmsz</code></td>
<td>( \left\langle u_z^2 \right\rangle^{1/2} ) for the hydro_zaver_range</td>
</tr>
<tr>
<td><code>durms</code></td>
<td>( \left\langle \delta u^2 \right\rangle^{1/2} )</td>
</tr>
<tr>
<td><code>umax</code></td>
<td>( \max(</td>
</tr>
<tr>
<td><code>umin</code></td>
<td>( \min(</td>
</tr>
<tr>
<td><code>uxrms</code></td>
<td>( \left\langle u_x^2 \right\rangle^{1/2} )</td>
</tr>
<tr>
<td><code>uyrms</code></td>
<td>( \left\langle u_y^2 \right\rangle^{1/2} )</td>
</tr>
</tbody>
</table>
uzrms \( \langle u_z^2 \rangle^{1/2} \)
uxmin \( \min(|u_x|) \)
uymin \( \min(|u_y|) \)
uzmin \( \min(|u_z|) \)
uxmax \( \max(|u_x|) \)
uymax \( \max(|u_y|) \)
uzmax \( \max(|u_z|) \)
uxm \( \langle u_x \rangle \)
uym \( \langle u_y \rangle \)
uzm \( \langle u_z \rangle \)
ux2m \( \langle u_x^2 \rangle \)
uy2m \( \langle u_y^2 \rangle \)
uz2m \( \langle u_z^2 \rangle \)
ux2ccm \( \langle u_x^2 \cos^2 k_z \rangle \)
ux2ssm \( \langle u_x^2 \sin^2 k_z \rangle \)
uy2ccm \( \langle u_y^2 \cos^2 k_z \rangle \)
uy2ssm \( \langle u_y^2 \sin^2 k_z \rangle \)
uxuyycsm \( \langle u_x u_y \cos k_z \sin k_z \rangle \)
uxuym \( \langle u_x u_y \rangle \)
uxuzm \( \langle u_x u_z \rangle \)
uyuzm \( \langle u_y u_z \rangle \)
uxm \( \langle u_x \rangle \)
uym \( \langle u_y \rangle \)
uzm \( \langle u_z \rangle \)
omumz \( \langle W \rangle_{xy} \cdot \langle U \rangle_{xy} \) (xy-averaged mean cross helicity production)
umamz \( \langle u \rangle_{xy} \cdot \langle A \rangle_{xy} \)
umbmz \( \langle U \rangle_{xy} \cdot \langle B \rangle_{xy} \) (xy-averaged mean cross helicity production)
umxbmz \( \langle U \rangle_{xy} \times \langle B \rangle_{xy} \rangle_z \) (xy-averaged mean emf)
rux2m \( \langle p u_x^2 \rangle \)
ruy2m \( \langle p u_y^2 \rangle \)
ruz2m \( \langle p u_z^2 \rangle \)
divum \( \langle \text{div}u \rangle \)
rdivum \( \langle \text{div}u \rangle \)
divu2m \( \langle (\text{div}u)^2 \rangle \)
gdivu2m \( \langle (\text{grad} \text{div}u)^2 \rangle \)
u3u21m \( \langle u_3 u_{2,1} \rangle \)
u1u32m \( \langle u_1 u_{3,2} \rangle \)
u1u32m \( \langle u_2 u_{1,3} \rangle \)
u2u31m \( \langle u_2 u_{3,1} \rangle \)
u3u12m \( \langle u_3 u_{1,2} \rangle \)
u1u23m \( \langle u_1 u_{2,3} \rangle \)
ruxm \( \langle p u_x \rangle \) (mean x-momentum density)
ruym \( \langle p u_y \rangle \) (mean y-momentum density)
ruzm \( \langle p u_z \rangle \) (mean z-momentum density)
ruxtot \( \langle \rho |u| \rangle \) (mean absolute x-momentum density)
rumax \( \max(\rho |u|) \) (maximum modulus of momentum)
\[ \langle \rho u_x u_y \rangle \] (mean Reynolds stress)
\[ \langle \rho u_x u_z \rangle \] (mean Reynolds stress)
\[ \langle \rho u_y u_z \rangle \] (mean Reynolds stress)

\[ \text{divrhourms} \] \[ \nabla \cdot (\rho \vec{u}) \] \[ \text{rms} \]
\[ \text{divrhoumax} \] \[ \nabla \cdot (\rho \vec{u}) \] \[ \text{max} \]
\[ \text{rlx} \] \[ \langle \rho u_z - z u_y \rangle \]
\[ \text{rly} \] \[ \langle \rho z u_x - x u_z \rangle \]
\[ \text{rlz} \] \[ \langle \rho x u_y - y u_z \rangle \]
\[ \text{rlx2m} \] \[ \langle (\rho u_z - z u_y)^2 \rangle \]
\[ \text{rly2m} \] \[ \langle (\rho z u_x - x u_z)^2 \rangle \]
\[ \text{rlz2m} \] \[ \langle (\rho x u_y - y u_z)^2 \rangle \]

\[ \text{tot_ang_mom} \] Total angular momentum in spherical coordinates about the axis.

\[ \text{dtu} \] \[ \delta t / [c_s \delta x / \text{max } |\vec{u}|] \] (time step relative to advective time step; see §5.11)
\[ \text{oum} \] \[ \langle \vec{\omega} \cdot \vec{u} \rangle \]
\[ \text{ou_int} \] \[ \int_V \vec{\omega} \cdot \vec{u} \, dV \]
\[ \text{fum} \] \[ \langle \vec{f} \cdot \vec{u} \rangle \]
\[ \text{odel2um} \] \[ \langle \vec{\omega} \nabla^2 \vec{u} \rangle \]
\[ \text{o2m} \] \[ \langle \vec{\omega}^2 \rangle \equiv \langle (\nabla \times \vec{u})^2 \rangle \]
\[ \text{orms} \] \[ \langle \vec{\omega}^2 \rangle^{1/2} \]
\[ \text{omax} \] \[ \text{max } |\vec{\omega}| \]
\[ \text{ox2m} \] \[ \langle \omega_x^2 \rangle \]
\[ \text{oy2m} \] \[ \langle \omega_y^2 \rangle \]
\[ \text{oz2m} \] \[ \langle \omega_z^2 \rangle \]
\[ \text{oxuzxm} \] \[ \langle \omega_x u_{z,x} \rangle \]
\[ \text{oyuzym} \] \[ \langle \omega_y u_{z,y} \rangle \]
\[ \text{oxoym} \] \[ \langle \omega_x \omega_y \rangle \]
\[ \text{oxyzm} \] \[ \langle \omega_x \omega_z \rangle \]
\[ \text{ozyzm} \] \[ \langle \omega_y \omega_z \rangle \]
\[ \text{qfm} \] \[ \langle \vec{q} \cdot \vec{f} \rangle \]
\[ \text{q2m} \] \[ \langle \vec{q}^2 \rangle \]
\[ \text{qrms} \] \[ \langle \vec{q}^2 \rangle^{1/2} \]
\[ \text{qmax} \] \[ \text{max } |\vec{q}| \]
\[ \text{qom} \] \[ \langle \vec{q} \cdot \vec{\omega} \rangle \]
\[ \text{quxom} \] \[ \langle \vec{q} \cdot (\vec{u} \times \vec{\omega}) \rangle \]
\[ \text{pvzvm} \] \[ \langle \omega_z + 2\Omega / \rho \rangle \] (z component of potential vorticity)
\[ \text{oumphi} \] \[ \langle \vec{\omega} \cdot \vec{u} \rangle \]
\[ \text{ugurmsx} \] \[ \langle (\vec{u} \nabla \vec{u})^2 \rangle^{1/2} \] for the hydro_xaver_range
\[ \text{ugu2m} \] \[ \langle \vec{u} \nabla \vec{u} \rangle^2 \]
\[ \text{dudx} \] \[ \langle \delta \vec{u} / \delta x \rangle \]
\[ \text{Marms} \] \[ \langle \vec{u}^2 / c_s^2 \rangle \] (rms Mach number)
\[ \text{Mamax} \] \[ \text{max } |\vec{u}| / c_s \] (maximum Mach number)
\[ \text{ekin} \] \[ \int_V \frac{1}{2} \rho \vec{u}^2 \, dV \]
\[ \text{ekintot} \] \[ \int_V \frac{1}{2} \rho \vec{u}^2 \, dV \]
\[ \text{uxglnrym} \] \[ \langle u_x \partial_y \ln \rho \rangle \]
\[ \text{uyglnrnxm} \] \[ \langle u_y \partial_x \ln \rho \rangle \]
\[ \text{uzdivum} \] \[ \langle u_z \nabla \cdot \vec{u} \rangle \]
\[ \text{uxuydivum} \] \[ \langle u_x u_y \nabla \cdot \vec{u} \rangle \]
### J.3 Parameters for 'print.in' 181

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>divuHrms</td>
<td>((\nabla_H \cdot \mathbf{u}_H)^{\text{rms}})</td>
</tr>
<tr>
<td>uxxrms</td>
<td>(u_{x,x}^{\text{rms}})</td>
</tr>
<tr>
<td>uyyrms</td>
<td>(u_{y,y}^{\text{rms}})</td>
</tr>
<tr>
<td>uzzrms</td>
<td>(u_{z,z}^{\text{rms}})</td>
</tr>
<tr>
<td>uzyrms</td>
<td>(u_{z,y}^{\text{rms}})</td>
</tr>
<tr>
<td>dtF</td>
<td>(\delta t / \left[ c_{\delta t} \delta x / \max</td>
</tr>
<tr>
<td>udpxxm</td>
<td>components of symmetric tensor (\langle u_i \partial_j p + u_j \partial_i p \rangle)</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>rhom</td>
<td>(\langle \rho \rangle) (mean density)</td>
</tr>
<tr>
<td>rhomxmask</td>
<td>(\langle \rho \rangle) for the density_xaver_range</td>
</tr>
<tr>
<td>rhomzmask</td>
<td>(\langle \rho \rangle) for the density_zaver_range</td>
</tr>
<tr>
<td>drho2m</td>
<td>(&lt; (\rho - \rho_0)^2 &gt;)</td>
</tr>
<tr>
<td>drhom</td>
<td>(&lt; \rho - \rho_0 &gt;)</td>
</tr>
<tr>
<td>rhomin</td>
<td>(\min(\rho))</td>
</tr>
<tr>
<td>rhomax</td>
<td>(\max(\rho))</td>
</tr>
<tr>
<td>ugrhom</td>
<td>(\langle \mathbf{u} \cdot \nabla \rho \rangle)</td>
</tr>
<tr>
<td>uglrhom</td>
<td>(\langle \mathbf{u} \cdot \nabla \ln \rho \rangle)</td>
</tr>
<tr>
<td>totmass</td>
<td>(\int \rho , dV)</td>
</tr>
<tr>
<td>mass</td>
<td>(\int \rho , dV)</td>
</tr>
<tr>
<td>vol</td>
<td>(\int dV) (volume)</td>
</tr>
<tr>
<td>grhromax</td>
<td>(\max(</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtc</td>
<td>(\delta t / \left[ c_{\delta t} \delta x / \max c_s \right] ) (time step relative to acoustic time step; see § 5.15)</td>
</tr>
<tr>
<td>ethm</td>
<td>(\langle \rho e \rangle) (mean thermal [=internal] energy)</td>
</tr>
<tr>
<td>ssruzm</td>
<td>(\langle s \rho u_z / c_p \rangle)</td>
</tr>
<tr>
<td>ssuzm</td>
<td>(\langle s \rho u_z / c_p \rangle)</td>
</tr>
<tr>
<td>ssm</td>
<td>(\langle s / c_p \rangle) (mean entropy)</td>
</tr>
<tr>
<td>ss2m</td>
<td>(\langle (s / c_p)^2 \rangle) (mean squared entropy)</td>
</tr>
<tr>
<td>eem</td>
<td>(\langle \epsilon \rangle)</td>
</tr>
<tr>
<td>ppm</td>
<td>(\langle p \rangle)</td>
</tr>
<tr>
<td>csm</td>
<td>(\langle c_s \rangle)</td>
</tr>
<tr>
<td>pdivum</td>
<td>(\langle p \nabla \cdot \mathbf{u} \rangle)</td>
</tr>
<tr>
<td>fradbot</td>
<td>(\int F_{\text{bot}} \cdot dS)</td>
</tr>
<tr>
<td>fradtop</td>
<td>(\int F_{\text{top}} \cdot dS)</td>
</tr>
<tr>
<td>TTtop</td>
<td>(\int T_{\text{top}} dS)</td>
</tr>
<tr>
<td>ethtot</td>
<td>(\int V \rho e , dV) (total thermal [=internal] energy)</td>
</tr>
<tr>
<td>dtchi</td>
<td>(\delta t / \left[ c_{\delta t,v} \delta x^2 / \chi_{\text{max}} \right] ) (time step relative to time step based on heat conductivity; see § 5.15)</td>
</tr>
<tr>
<td>Hmax</td>
<td>(H_{\text{max}}) (net heat sources summed see § 5.15)</td>
</tr>
<tr>
<td>tauhmin</td>
<td>(H_{\text{max}}) (net heat sources summed see § 5.15)</td>
</tr>
<tr>
<td>dtH</td>
<td>(\delta t / \left[ c_{\delta t,s} c_v T / H_{\text{max}} \right] ) (time step relative to time step based on heat sources; see § 5.15)</td>
</tr>
<tr>
<td>yHm</td>
<td>mean hydrogen ionization</td>
</tr>
<tr>
<td>yHmax</td>
<td>max of hydrogen ionization</td>
</tr>
<tr>
<td>TTm</td>
<td>(\langle T \rangle)</td>
</tr>
</tbody>
</table>
\[ TT_{\text{max}} \quad T_{\text{max}} \]
\[ TT_{\text{min}} \quad T_{\text{min}} \]
\[ gT_{\text{max}} \quad \max(|\nabla T|) \]
\[ ss_{\text{max}} \quad s_{\text{max}} \]
\[ ss_{\text{min}} \quad s_{\text{min}} \]
\[ gT_{\text{rms}} \quad (\nabla T)_{\text{rms}} \]
\[ g_{\text{rms}} \quad (\nabla s)_{\text{rms}} \]
\[ gT_{\times s_{\text{rms}}} \quad (\nabla T \times \nabla s)_{\text{rms}} \]

\[ f_{\text{convm}} \quad \langle c_{p}u_{z}T \rangle \]
\[ u_{\text{presm}} \quad \langle -u/\rho \nabla p \rangle \]
\[ K_{\text{kramers}} \quad \langle K_{\text{kramers}} \rangle \]

---

**Module 'magnetic.f90'**

\[ \eta_{\text{tdep}} \quad i\text{-dependent } \eta \]
\[ ab_{\text{int}} \quad \int A \cdot B \ dV \]
\[ jb_{\text{int}} \quad \int j \cdot B \ dV \]
\[ b2tm \quad \langle b(t) \cdot \int_{0}^{t} b(t')dt' \rangle \]
\[ hjtm \quad \langle b(t) \cdot \int_{0}^{t} j(t')dt' \rangle \]
\[ jbtm \quad \langle j(t) \cdot \int_{0}^{t} b(t')dt' \rangle \]
\[ b2ruzm \quad \langle B^{2}u_{z} \rangle \]
\[ b2uzm \quad \langle B^{2}u_{z} \rangle \]
\[ ubbzxm \quad \langle u \cdot B \rangle B_{z} \]
\[ b1m \quad \langle B \rangle \]
\[ b2m \quad \langle B^{2} \rangle \]
\[ b4m \quad \langle B^{4} \rangle \]
\[ bm2 \quad \max(B^{2}) \]
\[ j2m \quad \langle j^{2} \rangle \]
\[ jm2 \quad \max(j^{2}) \]
\[ abm \quad \langle A \cdot B \rangle \]
\[ abumx \quad \langle u_{x}A \cdot B \rangle \]
\[ abumy \quad \langle u_{y}A \cdot B \rangle \]
\[ abumz \quad \langle u_{z}A \cdot B \rangle \]
\[ abmh \quad \langle A \cdot B \rangle \text{ (temp)} \]
\[ abmn \quad \langle A \cdot B \rangle \text{ (north)} \]
\[ abms \quad \langle A \cdot B \rangle \text{ (south)} \]
\[ abrms \quad \langle (A \cdot B)^{2} \rangle^{1/2} \]
\[ jbrms \quad \langle (j \cdot B)^{2} \rangle^{1/2} \]
\[ ajm \quad \langle j \cdot A \rangle \]
\[ jbm \quad \langle j \cdot B \rangle \]
\[ jbmh \quad \langle J \cdot B \rangle \text{ (temp)} \]
\[ jbmn \quad \langle J \cdot B \rangle \text{ (north)} \]
\[ jbsm \quad \langle J \cdot B \rangle \text{ (south)} \]
\[ ubm \quad \langle u \cdot B \rangle \]
\[ dubrms \quad \langle (u - B)^{2} \rangle^{1/2} \]
\[ dobbrms \quad \langle (\omega - B)^{2} \rangle^{1/2} \]
\[ uxbxm \quad \langle u_{x}B_{x} \rangle \]
\[ uybxm \quad \langle u_{y}B_{x} \rangle \]
\[ uzbxm \quad \langle u_{z}B_{x} \rangle \]
uxbym $\langle u_x B_y \rangle$
uybym $\langle u_y B_y \rangle$
uzbym $\langle u_z B_y \rangle$
uxbzm $\langle u_x B_z \rangle$
uybzm $\langle u_y B_z \rangle$
uzbzm $\langle u_z B_z \rangle$

$\cosubm$ $\langle U \cdot B / (|U| |B|) \rangle$

jxbxm $\langle j_x B_x \rangle$
jybxm $\langle j_y B_x \rangle$
jzbxm $\langle j_z B_x \rangle$
jxbym $\langle j_x B_y \rangle$
jybym $\langle j_y B_y \rangle$
jzbym $\langle j_z B_y \rangle$
jxbzm $\langle j_x B_z \rangle$
jybzm $\langle j_y B_z \rangle$
jzbzm $\langle j_z B_z \rangle$

uam $\langle u \cdot A \rangle$
ujm $\langle u \cdot J \rangle$

$fbm$ $\langle f \cdot B \rangle$
fxbxm $\langle f_x B_x \rangle$

$\epsM$ $\langle \eta \mu_0 j_2 \rangle$

$\epsAD$ $\langle \rho^{-1} t_{AD} (J \times B)^2 \rangle$ (heating by ion-neutrals friction)

bxpt $B_x(x_1, y_1, z_1, t)$
bypt $B_y(x_1, y_1, z_1, t)$
bzpt $B_z(x_1, y_1, z_1, t)$
jxpt $J_x(x_1, y_1, z_1, t)$
jypt $J_y(x_1, y_1, z_1, t)$
jzpt $J_z(x_1, y_1, z_1, t)$

Expt $\mathcal{E}_x(x_1, y_1, z_1, t)$
Eypt $\mathcal{E}_y(x_1, y_1, z_1, t)$
Ezpt $\mathcal{E}_z(x_1, y_1, z_1, t)$

axpt $A_x(x_1, y_1, z_1, t)$
aypt $A_y(x_1, y_1, z_1, t)$
azpt $A_z(x_1, y_1, z_1, t)$

bxp2 $B_x(x_2, y_2, z_2, t)$
byp2 $B_y(x_2, y_2, z_2, t)$

Expt $\mathcal{E}_x(x_2, y_2, z_2, t)$
Eypt $\mathcal{E}_y(x_2, y_2, z_2, t)$

axp2 $A_x(x_2, y_2, z_2, t)$

exabot $\int E \times A dS |_{\text{bot}}$
exatop $\int E \times A dS |_{\text{top}}$
emag $\int V \frac{1}{\mu_0} B^2 dV$
brms $\langle B^2 \rangle^{1/2}$
bf rms \langle B^2 \rangle^{1/2}

bf 2m \langle B^2 \rangle

bf 4m \langle B^4 \rangle

b max \max(|B|)

b x min \min(|B_x|)

b y min \min(|B_y|)

b x min \min(|B_z|)

b y max \max(|B_y|)

b z max \max(|B_z|)

b bx max \max(|B_x|)_{\text{excluding } B_{\text{v ext}}}

b by max \max(|B_y|)_{\text{excluding } B_{\text{v ext}}}

b bz max \max(|B_z|)_{\text{excluding } B_{\text{v ext}}}

j x max \max(|j_x|)

j y max \max(|j_y|)

j z max \max(|j_z|)

j rms \langle j^2 \rangle^{1/2}

h j rms \langle j^2 \rangle^{3/2}

j max \max(|j|)

v Ar m s \langle B^2 / \rho \rangle^{1/2}

v A m x \max(B^2 / \rho)^{1/2}

d t b \delta t / [c_{\text{St}} \delta x / v_{\text{A, max}}] \quad \text{(time step relative to Alfvén time step; see § 5.15)}

d t e t a \delta t / [c_{\text{St}} \delta x / \eta_{\text{max}}] \quad \text{(time step relative to resistive time step; see § 5.15)}

a 2 m \langle A^2 \rangle

a r m s \langle A^2 \rangle^{1/2}

a m x \max(|A|)

d i v a r m s \langle (\nabla \cdot A)^2 \rangle^{1/2}

b et a 1 m \langle B^2 / (2 \mu_0 p) \rangle \quad \text{(mean inverse plasma beta)}

b et a 1 m a x \max(B^2 / (2 \mu_0 p)) \quad \text{(maximum inverse plasma beta)}

b et a m \langle \beta \rangle

b et a m a x \max \beta

b et a m i n \min \beta

b x m \langle B_x \rangle

b y m \langle B_y \rangle

b z m \langle B_z \rangle

b x y m \langle B_x B_y \rangle

b m x \langle B^2_{yz} \rangle^{1/2} \quad \text{(energy of } yz\text{-averaged mean field)}

b m y \langle B^2_{xz} \rangle^{1/2} \quad \text{(energy of } xz\text{-averaged mean field)}

b m z \langle B^2_{xy} \rangle^{1/2} \quad \text{(energy of } xy\text{-averaged mean field)}

b m z S 2 \langle B^2_{S, xy} \rangle

b m z A 2 \langle B^2_{A, xy} \rangle

j m x \langle J^2_{yz} \rangle^{1/2} \quad \text{(energy of } yz\text{-averaged mean current density)}
J.3 Parameters for 'print.in'

\[ j_{my} \langle J_{xz}^2 \rangle^{1/2} \text{ (energy of } xz\text{-averaged mean current density)} \]

\[ j_{mz} \langle J_{xy}^2 \rangle^{1/2} \text{ (energy of } xy\text{-averaged mean current density)} \]

\[ b_{mz\phi h} \text{ Phase of a Beltrami field} \]

\[ b_{mz\phi e} \text{ Error of phase of a Beltrami field} \]

\[ b_{s\sin\phi z} \text{ sine of phase of a Beltrami field} \]

\[ b_{c\cos\phi z} \text{ cosine of phase of a Beltrami field} \]

\[ e_{mxamz3} \langle E_{xy} \times \langle A \rangle_{xy} \rangle \text{ (} xy\text{-averaged mean field helicity flux)} \]

\[ e_{mbmz} \langle E_{xy} \cdot \langle B \rangle_{xy} \rangle \text{ (} xy\text{-averaged mean field helicity production)} \]

\[ a_{mbmz} \langle A_{xy} \cdot \langle B \rangle_{xy} \rangle \text{ (magnetic helicity of } xy\text{-averaged mean field)} \]

\[ a_{mbmzh} \langle A_{xy} \cdot \langle B \rangle_{xy} \rangle \text{ (magnetic helicity of } xy\text{-averaged mean field, temp)} \]

\[ a_{mbmzn} \langle A_{xy} \cdot \langle B \rangle_{xy} \rangle \text{ (magnetic helicity of } xy\text{-averaged mean field, north)} \]

\[ a_{mbmzs} \langle A_{xy} \cdot \langle B \rangle_{xy} \rangle \text{ (magnetic helicity of } xy\text{-averaged mean field, south)} \]

\[ j_{mbmz} \langle J_{xy} \cdot \langle B \rangle_{xy} \rangle \text{ (current helicity of } xy\text{-averaged mean field)} \]

\[ R_{mmz} \langle u \times B \rangle_0 \text{ (magnetic helicity of } xy\text{-averaged mean field)} \]

\[ k_{x_\Delta a} \langle u \times B \rangle_0 / \langle B \rangle_0^2 \]

\[ b_{x2m} \langle B_x^2 \rangle \]

\[ b_{y2m} \langle B_y^2 \rangle \]

\[ b_{z2m} \langle B_z^2 \rangle \]

\[ u_{xbm} \langle u \times B \rangle \cdot B_0 / B_0^2 \]

\[ j_{xbm} \langle j \times B \rangle \cdot B_0 / B_0^2 \]

\[ m_{agfricmax} \langle j \times B \rangle \cdot B^2 \]

\[ b_{3b21m} \langle B_3 B_{2,1} \rangle \]

\[ b_{3b12m} \langle B_3 B_{1,2} \rangle \]

\[ b_{1b32m} \langle B_1 B_{3,2} \rangle \]

\[ b_{1b23m} \langle B_1 B_{2,3} \rangle \]

\[ b_{2b13m} \langle B_2 B_{1,3} \rangle \]

\[ b_{2b31m} \langle B_2 B_{3,1} \rangle \]

\[ u_{xbm} \langle u \times B \rangle_x \]

\[ u_{xbm} \langle u \times B \rangle_y \]

\[ u_{xbm} \langle u \times B \rangle_z \]

\[ j_{xbm} \langle j \times B \rangle_x \]

\[ j_{xbm} \langle j \times B \rangle_y \]

\[ j_{xbm} \langle j \times B \rangle_z \]

\[ e_{axm} \langle E \times A \rangle_x \]

\[ e_{axm} \langle E \times A \rangle_y \]

\[ e_{axm} \langle E \times A \rangle_z \]

\[ e_{exjmx} \langle E \times J \rangle_x \]

\[ e_{exjmy} \langle E \times J \rangle_y \]

\[ e_{exjmx} \langle E \times J \rangle_z \]
\[
dexbmx \langle \nabla \times E \times B \rangle |_x \\
dexbmy \langle \nabla \times E \times B \rangle |_y \\
dexbmz \langle \nabla \times E \times B \rangle |_z \\
phibmx \langle \phi B \rangle |_x \\
phibmy \langle \phi B \rangle |_y \\
phibmz \langle \phi B \rangle |_z \\
h2divum \langle B^2 \nabla \cdot u \rangle \\
jdel2am \langle J \cdot \nabla^2 A \rangle \\
ujxbm \langle u \cdot (J \times B) \rangle \\
jxbbrmax \max(\langle |J \times B| / \rho \rangle) \\
jxbbr2m \langle (J \times B) / \rho \rangle^2 \\
bmxy_rms \sqrt{\langle (b_x)_x^2 (x, y) \rangle^2 + \langle (b_y)_y^2 (x, y) \rangle^2 + \langle (b_z)_z^2 (x, y) \rangle^2} \\
etasmagm \text{Mean of Smagorinsky resistivity} \\
etasmagmin \text{Min of Smagorinsky resistivity} \\
etasmagmax \text{Max of Smagorinsky resistivity} \\
etavamax \text{Max of artificial resistivity } \eta \sim v_A \\
etajmax \text{Max of artificial resistivity } \eta \sim J / \sqrt{\rho} \\
etaj2max \text{Max of artificial resistivity } \eta \sim J^2 / \rho \\
etajrhomax \text{Max of artificial resistivity } \eta \sim J / \rho \\
cosjbm \langle J \cdot B / (|J| |B|) \rangle \\
jparallelm \text{Mean value of the component of } J \text{ parallel to } B \\
jperpm \text{Mean value of the component of } J \text{ perpendicular to } B \\
jparallelpm \text{Mean value of the component of } J_{\text{hyper}} \text{ parallel to } B \\
jperpm \text{Mean value of the component of } J_{\text{hyper}} \text{ perpendicular to } B \\
brmsx \langle B^2 \rangle_1^{1/2} \text{ for the magnetic_xaver_range} \\
brmsz \langle B^2 \rangle_1^{1/2} \text{ for the magnetic_zaver_range} \\
Exmxy \langle E_x \rangle_z \\
Eymxy \langle E_y \rangle_z \\
Ezmxy \langle E_z \rangle_z \\
\text{Module 'pscalar.f90'}
\]
\[
rhoccm \langle \rho c \rangle \\
ccmax \max(c) \\
ccglnrm \langle c \nabla_z g \rangle \\
\text{Module '1D_loop.f90'}
\]
\[
dtchi2 \text{ heatconduction} \\
dtrad \text{ radiative loss from RTV} \\
dtrespitzer \text{ Spitzer heat conduction time step} \\
qmax \text{ max of heat flux vector} \\
qrms \text{ rms of heat flux vector} \\
\text{Module 'advective_gauge.f90'}
\]
\[
\text{Lamm} \langle \Lambda \rangle \\
\text{Lampt} \Lambda(x_1, y_1, z_1) \\
\text{Lamp2} \Lambda(x_2, y_2, z_2) \\
\text{Lamrms} \langle \Lambda \rangle^{1/2} \\
\text{Lambzm} \langle \Lambda B_z \rangle \\
\text{Lambzmz} \langle \Lambda B_z \rangle_{xy} \\
\text{gLambm} \langle \Lambda B \rangle \\
\]
### J.3 Parameters for ‘print.in’

- **apbrms**: \((\langle A'B \rangle^2)^{1/2}\)
- **jxarms**: \((\langle J \times A \rangle^2)^{1/2}\)
- **jxaprms**: \((\langle J \times A' \rangle^2)^{1/2}\)
- **jxgLamrms**: \((\langle J \times \nabla \Lambda \rangle^2)^{1/2}\)
- **divabrms**: \((\langle (\nabla \cdot A)B \rangle^2)^{1/2}\)
- **divapbrms**: \((\langle (\nabla \cdot A')B \rangle^2)^{1/2}\)
- **d2Lambrms**: \((\langle (\nabla^2\Lambda)B \rangle^2)^{1/2}\)
- **d2Lamrms**: \((\langle (\nabla^2\Lambda)^2 \rangle)^{1/2}\)

#### Module ‘anelastic.f90’

- **rhom**: \(\langle \bar{\rho} \rangle\) (mean density)
- **ugrhom**: \(\langle u \cdot \nabla \bar{\rho} \rangle\)
- **mass**: \(\int \bar{\rho} dV\)
- **divrhoun**: \(\langle \nabla \cdot (\bar{\rho}u) \rangle\)
- **divrhourms**: \(\langle \nabla \cdot (\bar{\rho}u) \rangle_{\text{rms}}\)
- **divrhoumax**: \(\langle \nabla \cdot (\bar{\rho}u) \rangle_{\text{max}}\)

#### Module ‘bfield.f90’

- **bmax**: \(\text{max } B\)
- **bmin**: \(\text{min } B\)
- **brms**: \(\langle B^2 \rangle^{1/2}\)
- **bm**: \(\langle B \rangle\)
- **b2m**: \(\langle B^2 \rangle\)
- **bxmax**: \(\text{max } |B_x|\)
- **bymax**: \(\text{max } |B_y|\)
- **bzmax**: \(\text{max } |B_z|\)
- **bxm**: \(\langle B_x \rangle\)
- **bym**: \(\langle B_y \rangle\)
- **bzm**: \(\langle B_z \rangle\)
- **bx2m**: \(\langle B^2_x \rangle\)
- **by2m**: \(\langle B^2_y \rangle\)
- **bz2m**: \(\langle B^2_z \rangle\)
- **bxym**: \(\langle B_x B_y \rangle\)
- **bxzm**: \(\langle B_x B_z \rangle\)
- **byzm**: \(\langle B_y B_z \rangle\)
- **dbxmax**: \(\text{max } |B_x - B_{\text{ext},x}|\)
- **dbymax**: \(\text{max } |B_y - B_{\text{ext},y}|\)
- **dbzmax**: \(\text{max } |B_z - B_{\text{ext},z}|\)
- **dbxm**: \(\langle B_x - B_{\text{ext},x} \rangle\)
- **dbym**: \(\langle B_y - B_{\text{ext},y} \rangle\)
- **dbzm**: \(\langle B_z - B_{\text{ext},z} \rangle\)
- **dbx2m**: \(\langle (B_x - B_{\text{ext},x})^2 \rangle\)
- **dby2m**: \(\langle (B_y - B_{\text{ext},y})^2 \rangle\)
- **dbz2m**: \(\langle (B_z - B_{\text{ext},z})^2 \rangle\)
- **jmax**: \(\text{max } J\)
- **jmin**: \(\text{min } J\)
- **jrms**: \(\langle J^2 \rangle^{1/2}\)
- **jm**: \(\langle J \rangle\)
\[ j2m \quad \langle J^2 \rangle \]
\[ jx_{\text{max}} \quad \max |J_x| \]
\[ jy_{\text{max}} \quad \max |J_y| \]
\[ jz_{\text{max}} \quad \max |J_z| \]
\[ jx_m \quad \langle J_x \rangle \]
\[ jy_m \quad \langle J_y \rangle \]
\[ jz_m \quad \langle J_z \rangle \]
\[ jx_{2m} \quad \langle J_x^2 \rangle \]
\[ jy_{2m} \quad \langle J_y^2 \rangle \]
\[ jz_{2m} \quad \langle J_z^2 \rangle \]
\[ \text{divb}_{\text{max}} \quad \max |\nabla \cdot B| \]
\[ \text{divb}_{\text{rms}} \quad \langle |\nabla \cdot B|^2 \rangle^{1/2} \]
\[ \text{beta}_{\text{max}} \quad \max \beta \]
\[ \text{beta}_{\text{min}} \quad \min \beta \]
\[ vA_{\text{max}} \quad \max v_A \]
\[ vA_{\text{min}} \quad \min v_A \]

---

**Module 'chemistry.f90'**

\[ dt_{\text{chem}} \quad dt_{\text{chem}} \]

**Module 'chemistry_simple.f90'**

\[ dt_{\text{chem}} \quad dt_{\text{chem}} \]

**Module 'chiral_fluids.f90'**

\[ \mu_{5m} \quad \langle \mu_5 \rangle \]
\[ \mu_{5\text{rms}} \quad \langle \mu_5^2 \rangle^{1/2} \]
\[ \text{bg}_{\mu_{5\text{rms}}} \quad \langle (B \cdot \nabla \mu_5)^2 \rangle^{1/2} \]
\[ \text{bg}_{\theta_{5\text{rms}}} \quad \langle (B \cdot \nabla \theta_5)^2 \rangle^{1/2} \]
\[ \theta_{5m} \quad \langle \theta_5 \rangle \]
\[ \theta_{5\text{rms}} \quad \langle \theta_5^2 \rangle^{1/2} \]

**Module 'chiral_fluids_gradtheta.f90'**

\[ \mu_{5m} \quad \langle \mu_5 \rangle \]
\[ \mu_{5\text{rms}} \quad \langle \mu_5^2 \rangle^{1/2} \]
\[ \text{bg}_{\mu_{5\text{rms}}} \quad \langle (B \cdot \nabla \mu_5)^2 \rangle^{1/2} \]
\[ \text{bg}_{\theta_{5\text{rms}}} \quad \langle (B \cdot \nabla \theta_5)^2 \rangle^{1/2} \]
\[ \theta_{5\text{rms}} \quad \langle \theta_5^2 \rangle^{1/2} \]
\[ \theta_{5\text{mx}} \quad \langle \theta_{5x} \rangle \]
\[ \theta_{5\text{my}} \quad \langle \theta_{5y} \rangle \]
\[ \theta_{5\text{mz}} \quad \langle \theta_{5z} \rangle \]

**Module 'chiral_mhd.f90'**

\[ \mu_{5m} \quad \langle \mu_5 \rangle \]
\[ \mu_{5\text{rms}} \quad \langle \mu_5^2 \rangle^{1/2} \]
\[ \text{g}_{\mu_{5\text{rms}}} \quad \langle (\nabla \mu_5)^2 \rangle^{1/2} \]
\[ \text{g}_{\mu_{5\text{mx}}} \quad \langle \nabla \mu_5 \rangle^2 \]
\[ \text{g}_{\mu_{5\text{my}}} \quad \langle \nabla \mu_5 \rangle \]
\[ \text{g}_{\mu_{5\text{mz}}} \quad \langle \nabla \mu_5 \rangle \]
J.3 Parameters for 'print.in'

- $gmu5my$: $\langle \nabla \mu_5 \rangle_y$
- $gmu5mz$: $\langle \nabla \mu_5 \rangle_z$
- $bgmu5rms$: $\langle (B \cdot \nabla \mu_5)^2 \rangle^{1/2}$
- $dt\_mu5\_1$: $\min(\mu_5/B^2)\delta x/(\lambda \eta)$
- $dt\_mu5\_2$: $(\lambda \eta \min(B^2))^{-1}$
- $dt\_mu5\_3$: $\delta x^2/D_5$
- $dt\_bb\_1$: $\delta x/(\eta \max(\mu_5))$
- $dt\_chiral$: total time-step contribution from chiral MHD

Module 'coronae.f90'

- $dtchi2$: $\delta t/[c_{s,t,v} \delta x^2/\chi_{\max}]$ (time step relative to time step based on heat conductivity; see §5.15)
- $dtspitzer$: Spitzer heat conduction time step
- $dtrad$: radiative loss from RTV

Module 'cosmicray_current.f90'

- $ekincr$: $\langle \frac{1}{2} \rho u_{cr}^2 \rangle$
- $ethmcr$: $\langle \rho e_{cr} \rangle$

Module 'density_stratified.f90'

- $mass$: $\int \rho \, d^3x$
- $rhomin$: $\min |\rho|$
- $rhomax$: $\max |\rho|$
- $drhom$: $\langle \Delta \rho/\rho_0 \rangle$
- $drho2m$: $\langle (\Delta \rho/\rho_0)^2 \rangle$
- $drhorms$: $\langle \Delta \rho/\rho_0 \rangle_{rms}$
- $drhormax$: $\max |\Delta \rho/\rho_0|$

Module 'detonate.f90'

- $detn$: Number of detonated sites (summed over time steps between adjacent outputs)
- $dettot$: Total energy input (summed over time steps between adjacent outputs)

Module 'dustdensity.f90'

- $KKm$: $\sum T_k^{\text{coag}}$
- $KK2m$: $\sum T_k^{\text{coag}}$
- $MMxm$: $\sum M_x^{\text{coag}}$
- $MMym$: $\sum M_y^{\text{coag}}$
- $MMzm$: $\sum M_z^{\text{coag}}$

Module 'entropy_anelastic.f90'

- $dtc$: $\delta t/[c_{s,t} \delta x/\max c_s]$ (time step relative to acoustic time step; see §5.15)
- $ethm$: $\langle e \rangle$ (mean thermal [=internal] energy)
- $ssm$: $\langle s/c_p \rangle$ (mean entropy)
- $ss2m$: $\langle (s/c_p)^2 \rangle$ (mean squared entropy)
- $eem$: $\langle e \rangle$
- $ppm$: $\langle p \rangle$
- $cs$: $\langle c_s \rangle$
- $pdium$: $\langle p \nabla u \rangle$
fradbot $\int F_{bot} \cdot dS$
fradtop $\int F_{top} \cdot dS$
TTtop $\int T_{top} dS$
ethtot $\int_V \rho e dV$ (total thermal [=internal] energy)
dtchi $\Delta t / \left[ \chi_{St} v \delta x / \chi_{max} \right]$ (time step relative to time step based on heat conductivity; see § 5.15)

Module ‘gravitational_waves.f90’

<table>
<thead>
<tr>
<th>Variable</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>hhT2m</td>
<td>$\langle h_T^2 \rangle$</td>
</tr>
<tr>
<td>hhX2m</td>
<td>$\langle h_X^2 \rangle$</td>
</tr>
<tr>
<td>hhThhXm</td>
<td>$\langle h_T h_X \rangle$</td>
</tr>
<tr>
<td>ggTpt</td>
<td>$g_T(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>strTpt</td>
<td>$S_T(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>strXpt</td>
<td>$S_X(x_1, y_1, z_1, t)$</td>
</tr>
</tbody>
</table>

Module ‘gravitational_waves_hij6.f90’

<table>
<thead>
<tr>
<th>Variable</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>h22rms</td>
<td>$h_{22}^{rms}$</td>
</tr>
<tr>
<td>h33rms</td>
<td>$h_{33}^{rms}$</td>
</tr>
<tr>
<td>h23rms</td>
<td>$h_{23}^{rms}$</td>
</tr>
<tr>
<td>g11pt</td>
<td>$g_{11}(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>g22pt</td>
<td>$g_{22}(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>g33pt</td>
<td>$g_{33}(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>g12pt</td>
<td>$g_{12}(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>g23pt</td>
<td>$g_{23}(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>g31pt</td>
<td>$g_{31}(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>hhTpt</td>
<td>$h_T(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>hhXpt</td>
<td>$h_X(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>ggTpt</td>
<td>$g_T(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>ggXpt</td>
<td>$g_X(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>hhTp2</td>
<td>$h_T(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>hhXp2</td>
<td>$h_X(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>ggTp2</td>
<td>$g_T(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>ggXp2</td>
<td>$g_X(x_1, y_1, z_1, t)$</td>
</tr>
<tr>
<td>hrms</td>
<td>$\langle h_T^2 + h_X^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>gg2m</td>
<td>$\langle g_T^2 + g_X^2 \rangle$</td>
</tr>
<tr>
<td>hhT2m</td>
<td>$\langle h_T^2 \rangle$</td>
</tr>
<tr>
<td>hhX2m</td>
<td>$\langle h_X^2 \rangle$</td>
</tr>
<tr>
<td>hhTXm</td>
<td>$\langle h_T h_X \rangle$</td>
</tr>
<tr>
<td>ggT2m</td>
<td>$\langle g_T^2 \rangle$</td>
</tr>
<tr>
<td>ggX2m</td>
<td>$\langle g_X^2 \rangle$</td>
</tr>
<tr>
<td>ggTXm</td>
<td>$\langle g_T g_X \rangle$</td>
</tr>
<tr>
<td>ggTm</td>
<td>$\langle g_T \rangle$</td>
</tr>
<tr>
<td>ggXm</td>
<td>$\langle g_X \rangle$</td>
</tr>
</tbody>
</table>

Module ‘gravity_simple.f90’

<table>
<thead>
<tr>
<th>Variable</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>epot</td>
<td>$\langle \rho \Phi_{grav} \rangle$ (mean potential energy)</td>
</tr>
<tr>
<td>epottot</td>
<td>$\int_V \rho \Phi_{grav} dV$ (total potential energy)</td>
</tr>
<tr>
<td>ugm</td>
<td>$\langle u \cdot g \rangle$</td>
</tr>
</tbody>
</table>
## Module 'heatflux.f90'

- **dtspitzer**: Spitzer heat conduction time step
- **dtq**: heatflux time step
- **dtq2**: heatflux time step due to tau
- **qmax**: $\max(|q|)$
- **qxmin**: $\min(|q_x|)$
- **qymin**: $\min(|q_y|)$
- **qzmin**: $\min(|q_z|)$
- **qxmax**: $\max(|q_x|)$
- **qymin**: $\max(|q_y|)$
- **qzmax**: $\max(|q_z|)$
- **qrms**: rms of heat flux vector
- **qsatmin**: minimum of qsat/qabs
- **qsatrms**: rms of qsat/abs

## Module 'lorenz_gauge.f90'

- **phim**: $\langle \phi \rangle$
- **phipt**: $\phi(x_1,y_1,z_1)$
- **phip2**: $\phi(x_2,y_2,z_2)$
- **phibzm**: $\langle \phi B_z \rangle$
- **phibzmz**: $\langle \phi B_z \rangle_{xy}$

## Module 'magnetic_shearboxJ.f90'

- **ab_int**: $\int (A \cdot B) \, dV$
- **jb_int**: $\int (j \cdot B) \, dV$
- **b2tm**: $\langle b(t) \cdot \int_0^t b(t') \, dt' \rangle$
- **bjtm**: $\langle b(t) \cdot \int_0^t j(t') \, dt' \rangle$
- **jbtm**: $\langle j(t) \cdot \int_0^t b(t') \, dt' \rangle$
- **b2ruzm**: $\langle B^2 \rho u_z \rangle$
- **b2uzm**: $\langle B^2 u_z \rangle$
- **ubbzmn**: $\langle (u \cdot B) B_z \rangle$
- **b1m**: $\langle |B| \rangle$
- **b2m**: $\langle B^2 \rangle$
- **bm2**: $\max(B^2)$
- **j2m**: $\langle j^2 \rangle$
- **jm2**: $\max(j^2)$
- **abm**: $\langle A \cdot B \rangle$
- **abumx**: $\langle u_x A \cdot B \rangle$
- **abumy**: $\langle u_y A \cdot B \rangle$
- **abumz**: $\langle u_z A \cdot B \rangle$
- **abmh**: $\langle A \cdot B \rangle$ (temp)
- **abmn**: $\langle A \cdot B \rangle$ (north)
- **abms**: $\langle A \cdot B \rangle$ (south)
- **abrms**: $\langle (A \cdot B)^2 \rangle^{1/2}$
- **jbrms**: $\langle (j \cdot B)^2 \rangle^{1/2}$
- **ajm**: $\langle j \cdot A \rangle$
- **jbm**: $\langle j \cdot B \rangle$
- **jbmh**: $\langle J \cdot B \rangle$ (temp)
The Pencil Code

\[ \langle J \cdot B \rangle \text{ (north)} \]
\[ \langle J \cdot B \rangle \text{ (south)} \]
\[ \langle u \cdot B \rangle \]
\[ \langle (u - B)^2 \rangle^{1/2} \]
\[ \langle (\omega - B)^2 \rangle^{1/2} \]
\[ \langle u_x B_x \rangle \]
\[ \langle u_y B_x \rangle \]
\[ \langle u_z B_x \rangle \]
\[ \langle u_x B_y \rangle \]
\[ \langle u_y B_y \rangle \]
\[ \langle u_z B_y \rangle \]
\[ \langle u_x B_z \rangle \]
\[ \langle u_y B_z \rangle \]
\[ \langle u_z B_z \rangle \]
\[ \cos \langle U \cdot B / (|U| |B|) \rangle \]
\[ \langle j_x B_x \rangle \]
\[ \langle j_y B_x \rangle \]
\[ \langle j_z B_x \rangle \]
\[ \langle j_x B_y \rangle \]
\[ \langle j_y B_y \rangle \]
\[ \langle j_z B_y \rangle \]
\[ \langle j_x B_z \rangle \]
\[ \langle j_y B_z \rangle \]
\[ \langle j_z B_z \rangle \]
\[ \langle u \cdot A \rangle \]
\[ \langle u \cdot J \rangle \]
\[ \langle f \cdot B \rangle \]
\[ \langle f_x B_x \rangle \]
\[ \langle \eta \mu_0 J^2 \rangle \]
\[ \langle \rho^{-1} t_{AD} (J \times B)^2 \rangle \text{ (heating by ion-neutrals friction)} \]
\[ B_x (x_1, y_1, z_1, t) \]
\[ B_y (x_1, y_1, z_1, t) \]
\[ B_z (x_1, y_1, z_1, t) \]
\[ J_x (x_1, y_1, z_1, t) \]
\[ J_y (x_1, y_1, z_1, t) \]
\[ J_z (x_1, y_1, z_1, t) \]
\[ E_x (x_1, y_1, z_1, t) \]
\[ E_y (x_1, y_1, z_1, t) \]
\[ E_z (x_1, y_1, z_1, t) \]
\[ A_x (x_1, y_1, z_1, t) \]
\[ A_y (x_1, y_1, z_1, t) \]
\[ A_z (x_1, y_1, z_1, t) \]
\[ B_x (x_2, y_2, z_2, t) \]
\[ B_y (x_2, y_2, z_2, t) \]
\[ B_z (x_2, y_2, z_2, t) \]
\[ J_x (x_2, y_2, z_2, t) \]
\[ J_y (x_2, y_2, z_2, t) \]
\[ J_z (x_2, y_2, z_2, t) \]
\[ E_x (x_2, y_2, z_2, t) \]
\[ E_y (x_2, y_2, z_2, t) \]
\begin{align*}
Ezp2 & \mathcal{E}_z(x_2, y_2, z_2, t) \\
axp2 & A_x(x_2, y_2, z_2, t) \\
ayp2 & A_y(x_2, y_2, z_2, t) \\
azzp2 & A_z(x_2, y_2, z_2, t) \\
exabot & \int E \times A dS|_{\text{bot}} \\
exatop & \int E \times A dS|_{\text{top}} \\
emag & \int_V \frac{1}{2 \mu_0} B^2 dV \\
brms & \left\langle B^2 \right\rangle^{1/2} \\
bfrms & \left\langle B'^2 \right\rangle^{1/2} \\
bmax & \max(|B|) \\
bxmin & \min(|B_x|) \\
bym & \min(|B_y|) \\
bzmin & \min(|B_z|) \\
bxmax & \max(|B_x|) \\
bymax & \max(|B_y|) \\
bzmax & \max(|B_z|) \\
bbxmax & \max(|B_x|) \text{ excluding } B_{v_{\text{ext}}} \\
bbymax & \max(|B_y|) \text{ excluding } B_{v_{\text{ext}}} \\
bbzmax & \max(|B_z|) \text{ excluding } B_{v_{\text{ext}}} \\
jxmax & \max(|j_{v_{\text{ext}}}|) \\
jymax & \max(|j_{v_{\text{ext}}}|) \\
jzmax & \max(|j_{v_{\text{ext}}}|) \\
jrms & \left\langle J^2 \right\rangle^{1/2} \\
hjrms & \left\langle J'^2 \right\rangle^{1/2} \\
jmax & \max(|j|) \\
vArms & \left\langle B^2 / \theta \right\rangle^{1/2} \\
vAmax & \max(B^2 / \theta)^{1/2} \\
dtb & \delta t / [c_{\text{Alfvén}} \delta x / v_{A\text{max}}] \quad \text{(time step relative to Alfvén time step; see §5.15)} \\
dteta & \delta t / [c_{\text{res}, \delta t} \delta x^2 / \eta_{\text{max}}] \quad \text{(time step relative to resistive time step; see §5.15)} \\
a2m & \left\langle A^2 \right\rangle \\
arms & \left\langle A^2 \right\rangle^{1/2} \\
amax & \max(|A|) \\
divarms & \left\langle (\nabla \cdot A)^2 \right\rangle^{1/2} \\
beta1m & \left\langle B^2 / (2\mu_0 p) \right\rangle \quad \text{(mean inverse plasma beta)} \\
beta1max & \max[B^2 / (2\mu_0 p)] \quad \text{(maximum inverse plasma beta)} \\
bet & \langle \beta \rangle \\
betamax & \max \beta \\
betamin & \min \beta \\
btmx & \langle B_x \rangle \\
btmy & \langle B_y \rangle \\
btmz & \langle B_z \rangle \\
btxmy & \langle B_x B_y \rangle \\
bmx & \left\langle B^2_{y_{\text{yz}}} \right\rangle^{1/2} \quad \text{(energy of } yz\text{-averaged mean field)} \\
bmy & \left\langle B^2_{x_{\text{xz}}} \right\rangle^{1/2} \quad \text{(energy of } xz\text{-averaged mean field)} \\
bnz & \left\langle B^2_{x_{\text{xy}}} \right\rangle^{1/2} \quad \text{(energy of } xy\text{-averaged mean field)}
\end{align*}
\[ \langle B_s \rangle_{xy}^2 \]

\[ \langle B_A \rangle_{xy}^2 \]

\[ \langle J \rangle_{yz}^2 \] (energy of \(yz\)-averaged mean current density)

\[ \langle J \rangle_{xz}^2 \] (energy of \(xz\)-averaged mean current density)

\[ \langle J \rangle_{xy}^2 \] (energy of \(xy\)-averaged mean current density)

\( bmz \) Phase of a Beltrami field

\( bsinphz \) sine of phase of a Beltrami field

\( bcosphz \) cosine of phase of a Beltrami field

\[ \langle E \rangle_{xy} \times \langle A \rangle_{xy} \] (\(xy\)-averaged mean field helicity flux)

\[ \langle E \rangle_{xy} \cdot \langle B \rangle_{xy} \] (\(xy\)-averaged mean field helicity production)

\[ \langle A \rangle_{xy} \cdot \langle B \rangle_{xy} \] (magnetic helicity of \(xy\)-averaged mean field)

\[ \langle A \rangle_{xy} \cdot \langle B \rangle_{xy} \] (magnetic helicity of \(xy\)-averaged mean field, temp)

\[ \langle A \rangle_{xy} \cdot \langle B \rangle_{xy} \] (magnetic helicity of \(xy\)-averaged mean field, north)

\[ \langle A \rangle_{xy} \cdot \langle B \rangle_{xy} \] (magnetic helicity of \(xy\)-averaged mean field, south)

\[ \langle J \rangle_{xy} \cdot \langle B \rangle_{xy} \] (current helicity of \(xy\)-averaged mean field)

\[ \langle J \rangle_{xy} \cdot \langle B \rangle_{xy} \] (current helicity of \(xy\)-averaged mean field)

\[ \langle u \times B \rangle \cdot B_0 / B_0^2 \]

\[ \langle j \times B \rangle \cdot B_0 / B_0^2 \]

\( magfricmax \) Magneto-Frictional velocity \( \langle j \times B \rangle \cdot B^2 \)

\[ \langle B_3 \rangle_{B_{2,1}} \]

\[ \langle B_3 \rangle_{B_{1,2}} \]

\[ \langle B_3 \rangle_{B_{3,2}} \]

\[ \langle B_1 \rangle_{B_{2,3}} \]

\[ \langle B_2 \rangle_{B_{1,5}} \]

\[ \langle B_3 \rangle_{B_{2,3}} \]

\[ \langle u \times B \rangle_z \]

\[ \langle u \times B \rangle_y \]

\[ \langle u \times B \rangle_z \]

\[ \langle j \times B \rangle_z \]

\[ \langle j \times B \rangle_y \]

\[ \langle j \times B \rangle_z \]

\[ \langle E \times A \rangle_x \]

\[ \langle E \times A \rangle_y \]

\[ \langle E \times A \rangle_z \]
J.3 Parameters for 'print.in'

\[ \langle E \times J \rangle \big|_x \]
\[ \langle E \times J \rangle \big|_y \]
\[ \langle E \times J \rangle \big|_z \]
\[ \langle \nabla \times E \times B \rangle \big|_x \]
\[ \langle \nabla \times E \times B \rangle \big|_y \]
\[ \langle \nabla \times E \times B \rangle \big|_z \]
\[ \langle \phi B \rangle \big|_x \]
\[ \langle \phi B \rangle \big|_y \]
\[ \langle \phi B \rangle \big|_z \]
\[ \langle \nabla \times E \times B \rangle \big| \times \nabla \cdot u \]
\[ \langle u \cdot (J \times B) \rangle \]
\[ \max(|J \times B|) \]
\[ \langle |J \times B|/\rho \rangle^2 \]
\[ \sqrt{\langle b_x \rangle_z(x,y)^2 + \langle b_y \rangle_z(x,y)^2 + \langle b_z \rangle_z(x,y)^2} \]
\[ \text{Mean of Smagorinsky resistivity} \]
\[ \text{Min of Smagorinsky resistivity} \]
\[ \text{Max of Smagorinsky resistivity} \]
\[ \text{Max of artificial resistivity } \eta \sim v_A \]
\[ \text{Max of artificial resistivity } \eta \sim J/\sqrt{\rho} \]
\[ \text{Max of artificial resistivity } \eta \sim J^2/\rho \]
\[ \langle J \cdot B/(|J| |B|) \rangle \]
\[ \text{Mean value of the component of } J \text{ parallel to } B \]
\[ \text{Mean value of the component of } J \text{ perpendicular to } B \]
\[ \text{Mean value of the component of } J_{\text{hyper}} \text{ parallel to } B \]
\[ \langle B^2 \rangle^{1/2} \text{ for the } \text{magnetic_xaver_range} \]
\[ \langle E \rangle_z \]
\[ \langle \phi \rangle \]
\[ \langle \eta_t \rangle \]
\[ \text{EMF}_{\text{mz1}}\big|_x \]
\[ \text{EMF}_{\text{mz2}}\big|_y \]
\[ \text{EMF}_{\text{mz3}}\big|_z \]
\[ \text{EMFdotBm} \big|_{x,y,z} \]
\[ \text{EMFdotB_int} \int \mathbf{E} \cdot \mathbf{B} dV \]

Module 'meanfield.f90'

Module 'meanfield_demfdt.f90'
Module ‘noentropy.f90’

\[ dtc = \frac{\delta t}{[c_{gs} \delta x / \max c_a]} \]  
(time step relative to acoustic time step; see §5.15)

\[ ethm = \langle \rho c \rangle \]  
(mean thermal [=internal] energy)

\[ pdivum = \langle p \nabla u \rangle \]

Module ‘particles_chemistry.f90’

\[ Shcm = \text{meanparticleSherwoodnumber} \]

Module ‘particles_dust.f90’

\[ \begin{align*}
\text{vrelpabsm} &= \text{Absolutevalueofmeanrelativevelocity} \\
\text{vp cumulative} &= \langle u \rangle \\
\text{vp momentum} &= \langle u^2 \rangle \\
\text{ekinp} &= E_{\text{kin,part}} \\
\text{vpx max} &= \text{MAX}(u) \\
\text{vpx min} &= \text{MIN}(u) \\
\text{npm} &= \text{meanparticlenumberdensity}
\end{align*} \]

Module ‘particles_dust_brdeplete.f90’

\[ \begin{align*}
\text{vrelpabsm} &= \text{Absolutevalueofmeanrelativevelocity} \\
\text{vp cumulative} &= \langle u \rangle \\
\text{vp momentum} &= \langle u^2 \rangle \\
\text{ekinp} &= E_{\text{kin,part}} \\
\text{vpx max} &= \text{MAX}(u) \\
\text{vpx min} &= \text{MIN}(u) \\
\text{npm} &= \text{meanparticlenumberdensity}
\end{align*} \]

Module ‘particles_lagrangian.f90’

\[ \begin{align*}
\text{vrelpabsm} &= \text{Absolutevalueofmeanrelativevelocity} \\
\text{vp cumulative} &= \langle u \rangle \\
\text{vp momentum} &= \langle u^2 \rangle \\
\text{ekinp} &= E_{\text{kin,part}} \\
\text{vpx max} &= \text{MAX}(u) \\
\text{vpx min} &= \text{MIN}(u) \\
\text{npm} &= \text{meanparticlenumberdensity}
\end{align*} \]

Module ‘particles_mass_swarm.f90’

\[ \begin{align*}
\text{m cumulative} &= \langle m \rangle \\
\text{m min} &= \min_j m_{p,j} \\
\text{m max} &= \max_j m_{p,j}
\end{align*} \]

Module ‘particles_surfspec.f90’

\[ dtpchem = dt_{\text{particle,chemistry}} \]
### Module ‘polymer.f90’

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>polytrm</td>
<td>$\langle Tr[C_{ij}] \rangle$</td>
</tr>
<tr>
<td>frmax</td>
<td>$\max(f(r))$</td>
</tr>
</tbody>
</table>

### Module ‘shear.f90’

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtshear</td>
<td>advec_shear/cdt</td>
</tr>
<tr>
<td>deltax</td>
<td>deltax</td>
</tr>
</tbody>
</table>

### Module ‘shock.f90’

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shockmax</td>
<td>Max shock number</td>
</tr>
</tbody>
</table>

### Module ‘shock_highorder.f90’

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>gshockmax</td>
<td>$\max</td>
</tr>
</tbody>
</table>

### Module ‘solar_corona.f90’

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtvel</td>
<td>Velocity driver time step</td>
</tr>
<tr>
<td>dtnewt</td>
<td>Radiative cooling time step</td>
</tr>
<tr>
<td>dtradloss</td>
<td>Radiative losses time step</td>
</tr>
<tr>
<td>dtchi2</td>
<td>$\delta t/[c_{\delta t,v} \delta x^2/\chi_{\text{max}}]$ (time step relative to time step based on heat conductivity; see §5.15)</td>
</tr>
<tr>
<td>dtspitzer</td>
<td>Spitzer heat conduction time step</td>
</tr>
<tr>
<td>mag_flux</td>
<td>Total vertical magnetic flux at</td>
</tr>
</tbody>
</table>

### Module ‘solid_cells_CGEO.f90’

### Module ‘solid_cells_reactive.f90’

### Module ‘temperature_idealgas.f90’

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTmax</td>
<td>$\max(T)$</td>
</tr>
<tr>
<td>gTmax</td>
<td>$\max(</td>
</tr>
<tr>
<td>TTmin</td>
<td>$\min(T)$</td>
</tr>
<tr>
<td>TTm</td>
<td>$\langle T \rangle$</td>
</tr>
<tr>
<td>TTzmask</td>
<td>$\langle T \rangle$ for the temp_zaver_range</td>
</tr>
<tr>
<td>TT2m</td>
<td>$\langle T^2 \rangle$</td>
</tr>
<tr>
<td>TugTm</td>
<td>$\langle T \cdot \nabla T \rangle$</td>
</tr>
<tr>
<td>Trms</td>
<td>$\sqrt{\langle T^2 \rangle}$</td>
</tr>
<tr>
<td>uXTm</td>
<td>$\langle u_x T \rangle$</td>
</tr>
<tr>
<td>uyTm</td>
<td>$\langle u_y T \rangle$</td>
</tr>
<tr>
<td>uzTm</td>
<td>$\langle u_z T \rangle$</td>
</tr>
<tr>
<td>gT2m</td>
<td>$\langle (\nabla T)^2 \rangle$</td>
</tr>
<tr>
<td>guxgTm</td>
<td>$\langle \nabla u_x \cdot \nabla T \rangle$</td>
</tr>
<tr>
<td>guygTm</td>
<td>$\langle \nabla u_y \cdot \nabla T \rangle$</td>
</tr>
<tr>
<td>guzgTm</td>
<td>$\langle \nabla u_z \cdot \nabla T \rangle$</td>
</tr>
<tr>
<td>Tugux_uXugTm</td>
<td>$\langle T \cdot \nabla u_x + u_x T \cdot \nabla T \rangle = \langle u \cdot \nabla (u_x T) \rangle$</td>
</tr>
<tr>
<td>Tuguy_uYugTm</td>
<td>$\langle T \cdot \nabla u_y + u_y T \cdot \nabla T \rangle = \langle u \cdot \nabla (u_y T) \rangle$</td>
</tr>
<tr>
<td>Tuguz_uZugTm</td>
<td>$\langle T \cdot \nabla u_z + u_z T \cdot \nabla T \rangle = \langle u \cdot \nabla (u_z T) \rangle$</td>
</tr>
<tr>
<td>Tdpxm</td>
<td>$\langle T \cdot dp/\delta x \rangle$</td>
</tr>
<tr>
<td>Tdpygm</td>
<td>$\langle T \cdot dp/\delta y \rangle$</td>
</tr>
<tr>
<td>Tdzpm</td>
<td>$\langle T \cdot dp/\delta z \rangle$</td>
</tr>
<tr>
<td>fradtop</td>
<td>$&lt; -K \frac{dT}{dz} &gt;_{\text{top}}$ (top radiative flux)</td>
</tr>
</tbody>
</table>
The Pencil Code

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>fradbot</td>
<td>$&lt; -K \frac{dT}{dz} &gt;_{bot}$ (bottom radiative flux)</td>
</tr>
<tr>
<td>yHmax</td>
<td>DOCUMENT ME</td>
</tr>
<tr>
<td>yHmin</td>
<td>DOCUMENT ME</td>
</tr>
<tr>
<td>yHm</td>
<td>DOCUMENT ME</td>
</tr>
<tr>
<td>ethm</td>
<td>$\langle e_{th} \rangle = \langle c_p \rho T \rangle$ (mean thermal energy)</td>
</tr>
<tr>
<td>eem</td>
<td>$\langle e \rangle = \langle c_p T \rangle$ (mean internal energy)</td>
</tr>
<tr>
<td>ethtot</td>
<td>$\int_V \rho e , dV$ (total thermal energy)</td>
</tr>
<tr>
<td>ssm</td>
<td>$\bar{S}$</td>
</tr>
<tr>
<td>thcool</td>
<td>$\tau_{cool}$</td>
</tr>
<tr>
<td>ppm</td>
<td>$\bar{P}$</td>
</tr>
<tr>
<td>csm</td>
<td>$\bar{c}_s$</td>
</tr>
<tr>
<td>dtc</td>
<td>$\delta t / [c_s \delta x / \max c_s]$ (time step relative to acoustic time step; see §5.15)</td>
</tr>
<tr>
<td>dtchi</td>
<td>$\delta t / [c_s \delta x^2 / \chi_{max}]$ (time step relative to time step based on heat conductivity; see §5.15)</td>
</tr>
<tr>
<td>Emzmask</td>
<td>$\langle n^2 \exp - (\log T - \log T_0)^2 / (\delta \log T)^2 \rangle$ the emiss.zaver.range</td>
</tr>
</tbody>
</table>

**Module 'temperature_ionization.f90'**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTmax</td>
<td>$\max(T)$</td>
</tr>
<tr>
<td>TTmin</td>
<td>$\min(T)$</td>
</tr>
<tr>
<td>TTm</td>
<td>$\langle T \rangle$</td>
</tr>
<tr>
<td>ethm</td>
<td>$\langle e_{th} \rangle = \langle c_p \rho T \rangle$ (mean thermal energy)</td>
</tr>
<tr>
<td>eem</td>
<td>$\langle e \rangle$ (mean internal energy)</td>
</tr>
<tr>
<td>ppm</td>
<td>$\langle p \rangle$</td>
</tr>
</tbody>
</table>

**Module 'testfield_axisym.f90'**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpPERP</td>
<td>$\alpha_{\bot}$</td>
</tr>
<tr>
<td>alpPARA</td>
<td>$\alpha_{\bot}$</td>
</tr>
<tr>
<td>gam</td>
<td>$\gamma$</td>
</tr>
<tr>
<td>betPERP</td>
<td>$\beta_{\bot}$</td>
</tr>
<tr>
<td>betPARA</td>
<td>$\beta_{\bot}$</td>
</tr>
<tr>
<td>del</td>
<td>$\delta$</td>
</tr>
<tr>
<td>kapPERP</td>
<td>$\kappa_{\bot}$</td>
</tr>
<tr>
<td>kapPARA</td>
<td>$\kappa_{\bot}$</td>
</tr>
<tr>
<td>mu</td>
<td>$\mu$</td>
</tr>
<tr>
<td>alpPERPz</td>
<td>$\alpha_{\bot}(z)$</td>
</tr>
<tr>
<td>alpPARAz</td>
<td>$\alpha_{\bot}(z)$</td>
</tr>
<tr>
<td>gamz</td>
<td>$\gamma(z)$</td>
</tr>
<tr>
<td>betPERPz</td>
<td>$\beta_{\bot}(z)$</td>
</tr>
<tr>
<td>betPARAz</td>
<td>$\beta_{\bot}(z)$</td>
</tr>
<tr>
<td>delz</td>
<td>$\delta(z)$</td>
</tr>
<tr>
<td>kapPERPz</td>
<td>$\kappa_{\bot}(z)$</td>
</tr>
<tr>
<td>kapPARAz</td>
<td>$\kappa_{\bot}(z)$</td>
</tr>
<tr>
<td>muz</td>
<td>$\mu(z)$</td>
</tr>
<tr>
<td>bx1pt</td>
<td>$b_{x1}^1$</td>
</tr>
<tr>
<td>bx2pt</td>
<td>$b_{x2}^1$</td>
</tr>
<tr>
<td>bx3pt</td>
<td>$b_{x3}^1$</td>
</tr>
<tr>
<td>b1rms</td>
<td>$\langle b_1^1 \rangle^{1/2}$</td>
</tr>
<tr>
<td>b2rms</td>
<td>$\langle b_2^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>b3rms</td>
<td>$\langle b_3^3 \rangle^{1/2}$</td>
</tr>
</tbody>
</table>
### Module 'testfield_axisym2.f90'

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpPERP</td>
<td>$\alpha_\perp$</td>
</tr>
<tr>
<td>alpPARA</td>
<td>$\alpha_\perp$</td>
</tr>
<tr>
<td>gam</td>
<td>$\gamma$</td>
</tr>
<tr>
<td>betPERP</td>
<td>$\beta_\perp$</td>
</tr>
<tr>
<td>betPARA</td>
<td>$\beta_\perp$</td>
</tr>
<tr>
<td>del</td>
<td>$\delta$</td>
</tr>
<tr>
<td>kapPERP</td>
<td>$\kappa_\perp$</td>
</tr>
<tr>
<td>kapPARA</td>
<td>$\kappa_\perp$</td>
</tr>
<tr>
<td>mu</td>
<td>$\mu$</td>
</tr>
<tr>
<td>bx1pt</td>
<td>$b_1^x$</td>
</tr>
<tr>
<td>bx2pt</td>
<td>$b_2^x$</td>
</tr>
<tr>
<td>bx3pt</td>
<td>$b_3^x$</td>
</tr>
<tr>
<td>b1rms</td>
<td>$\langle b_1^x \rangle^{1/2}$</td>
</tr>
<tr>
<td>b2rms</td>
<td>$\langle b_2^x \rangle^{1/2}$</td>
</tr>
<tr>
<td>b3rms</td>
<td>$\langle b_3^x \rangle^{1/2}$</td>
</tr>
</tbody>
</table>

### Module 'testfield_axisym4.f90'

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpPERP</td>
<td>$\alpha_\perp$</td>
</tr>
<tr>
<td>alpPARA</td>
<td>$\alpha_\perp$</td>
</tr>
<tr>
<td>gam</td>
<td>$\gamma$</td>
</tr>
<tr>
<td>betPERP</td>
<td>$\beta_\perp$</td>
</tr>
<tr>
<td>betPERP2</td>
<td>$\beta_\perp^{(2)}$</td>
</tr>
<tr>
<td>betPARA</td>
<td>$\beta_\perp$</td>
</tr>
<tr>
<td>del</td>
<td>$\delta$</td>
</tr>
<tr>
<td>del2</td>
<td>$\delta^{(2)}$</td>
</tr>
<tr>
<td>kapPERP</td>
<td>$\kappa_\perp$</td>
</tr>
<tr>
<td>kapPERP2</td>
<td>$\kappa_\perp^{(2)}$</td>
</tr>
<tr>
<td>kapPARA</td>
<td>$\kappa_\perp$</td>
</tr>
<tr>
<td>mu</td>
<td>$\mu$</td>
</tr>
<tr>
<td>mu2</td>
<td>$\mu^{(2)}$</td>
</tr>
<tr>
<td>alpPERPz</td>
<td>$\alpha_\perp(z)$</td>
</tr>
<tr>
<td>alpPARAz</td>
<td>$\alpha_\perp(z)$</td>
</tr>
<tr>
<td>gamz</td>
<td>$\gamma(z)$</td>
</tr>
<tr>
<td>betPERPz</td>
<td>$\beta_\perp(z)$</td>
</tr>
<tr>
<td>betPARAz</td>
<td>$\beta_\perp(z)$</td>
</tr>
<tr>
<td>delz</td>
<td>$\delta(z)$</td>
</tr>
<tr>
<td>kapPERPz</td>
<td>$\kappa_\perp(z)$</td>
</tr>
<tr>
<td>kapPARAz</td>
<td>$\kappa_\perp(z)$</td>
</tr>
<tr>
<td>muz</td>
<td>$\mu(z)$</td>
</tr>
<tr>
<td>bx1pt</td>
<td>$b_1^x$</td>
</tr>
<tr>
<td>bx2pt</td>
<td>$b_2^x$</td>
</tr>
<tr>
<td>bx3pt</td>
<td>$b_3^x$</td>
</tr>
<tr>
<td>b1rms</td>
<td>$\langle b_1^x \rangle^{1/2}$</td>
</tr>
<tr>
<td>b2rms</td>
<td>$\langle b_2^x \rangle^{1/2}$</td>
</tr>
<tr>
<td>b3rms</td>
<td>$\langle b_3^x \rangle^{1/2}$</td>
</tr>
</tbody>
</table>

### Module 'testfield_compress_z.f90'


\[ \begin{align*} 
alp11 & = \alpha_{11} \\
alp21 & = \alpha_{21} \\
alp31 & = \alpha_{31} \\
alp12 & = \alpha_{12} \\
alp22 & = \alpha_{22} \\
alp32 & = \alpha_{32} \\
eta11 & = \eta_{11} \\
eta21 & = \eta_{21} \\
eta12 & = \eta_{12} \\
eta22 & = \eta_{22} \\
alpK & = \alpha_{K} \\
alpM & = \alpha_{M} \\
alpMK & = \alpha_{MK} \\
phi11 & = \phi_{11} \\
phi21 & = \phi_{21} \\
phi12 & = \phi_{12} \\
phi22 & = \phi_{22} \\
phi32 & = \phi_{32} \\
psi11 & = \psi_{11} \\
psi21 & = \psi_{21} \\
psi12 & = \psi_{12} \\
psi22 & = \psi_{22} \\
phiK & = \phi_{K} \\
phiM & = \phi_{M} \\
phiMK & = \phi_{MK} \\
alp11cc & = \alpha_{11} \cos^2 k z \\
alp21sc & = \alpha_{21} \sin k z \cos k z \\
alp12cs & = \alpha_{12} \cos k z \sin k z \\
alp22ss & = \alpha_{22} \sin^2 k z \\
eta11cc & = \eta_{11} \cos^2 k z \\
eta21sc & = \eta_{21} \sin k z \cos k z \\
eta12cs & = \eta_{12} \cos k z \sin k z \\
eta22ss & = \eta_{22} \sin^2 k z \\
s2kzDFm & = \langle \sin 2 k z \nabla \cdot F \rangle \\
M11 & = M_{11} \\
M22 & = M_{22} \\
M33 & = M_{33} \\
M11cc & = M_{11} \cos^2 k z \\
M11ss & = M_{11} \sin^2 k z \\
M22cc & = M_{22} \cos^2 k z \\
M22ss & = M_{22} \sin^2 k z \\
M12cs & = M_{12} \cos k z \sin k z \\
bx11pt & = b_{x}^{11} \\
bx21pt & = b_{x}^{21} \\
bx12pt & = b_{x}^{12} \\
bx22pt & = b_{x}^{22} \\
bx0pt & = b_{x}^{0} \\
by11pt & = b_{y}^{11} \\
by21pt & = b_{y}^{21} \\
by12pt & = b_{y}^{12} \\
by12pt & = b_{y}^{12} \end{align*} \]
### J.3 Parameters for 'print.in'

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$by22pt$</td>
<td>$b^{22}$</td>
</tr>
<tr>
<td>$by0pt$</td>
<td>$b^0_y$</td>
</tr>
<tr>
<td>$u11rms$</td>
<td>$\langle u_{11}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$u21rms$</td>
<td>$\langle u_{21}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$u12rms$</td>
<td>$\langle u_{12}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$u22rms$</td>
<td>$\langle u_{22}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$j11rms$</td>
<td>$\langle j_{11}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$b11rms$</td>
<td>$\langle b_{11}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$b21rms$</td>
<td>$\langle b_{21}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$b12rms$</td>
<td>$\langle b_{12}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$b22rms$</td>
<td>$\langle b_{22}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$ux0m$</td>
<td>$\langle u_{0x} \rangle$</td>
</tr>
<tr>
<td>$uy0m$</td>
<td>$\langle u_{0y} \rangle$</td>
</tr>
<tr>
<td>$ux11m$</td>
<td>$\langle u_{11x} \rangle$</td>
</tr>
<tr>
<td>$uy11m$</td>
<td>$\langle u_{11y} \rangle$</td>
</tr>
<tr>
<td>$u0rms$</td>
<td>$\langle u_{0}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$b0rms$</td>
<td>$\langle b_{0}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$jb0m$</td>
<td>$\langle j_{b0} \rangle$</td>
</tr>
<tr>
<td>$E11rms$</td>
<td>$\langle E_{11}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$E21rms$</td>
<td>$\langle E_{21}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$E12rms$</td>
<td>$\langle E_{12}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$E22rms$</td>
<td>$\langle E_{22}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$E0rms$</td>
<td>$\langle E_{0}^2 \rangle^{1/2}$</td>
</tr>
<tr>
<td>$Ex11pt$</td>
<td>$E_{11}^x$</td>
</tr>
<tr>
<td>$Ex21pt$</td>
<td>$E_{21}^x$</td>
</tr>
<tr>
<td>$Ex12pt$</td>
<td>$E_{12}^x$</td>
</tr>
<tr>
<td>$Ex22pt$</td>
<td>$E_{22}^x$</td>
</tr>
<tr>
<td>$Ex0pt$</td>
<td>$E_{0}^x$</td>
</tr>
<tr>
<td>$Ey11pt$</td>
<td>$E_{11}^y$</td>
</tr>
<tr>
<td>$Ey21pt$</td>
<td>$E_{21}^y$</td>
</tr>
<tr>
<td>$Ey12pt$</td>
<td>$E_{12}^y$</td>
</tr>
<tr>
<td>$Ey22pt$</td>
<td>$E_{22}^y$</td>
</tr>
<tr>
<td>$Ey0pt$</td>
<td>$E_{0}^y$</td>
</tr>
<tr>
<td>$bamp$</td>
<td>$bamp$</td>
</tr>
<tr>
<td>$E111z$</td>
<td>$E_{11}^z$</td>
</tr>
<tr>
<td>$E211z$</td>
<td>$E_{21}^z$</td>
</tr>
<tr>
<td>$E311z$</td>
<td>$E_{31}^z$</td>
</tr>
<tr>
<td>$E121z$</td>
<td>$E_{12}^z$</td>
</tr>
<tr>
<td>$E221z$</td>
<td>$E_{22}^z$</td>
</tr>
<tr>
<td>$E321z$</td>
<td>$E_{32}^z$</td>
</tr>
<tr>
<td>$E112z$</td>
<td>$E_{11}^2$</td>
</tr>
<tr>
<td>$E212z$</td>
<td>$E_{21}^2$</td>
</tr>
<tr>
<td>$E312z$</td>
<td>$E_{31}^2$</td>
</tr>
<tr>
<td>$E122z$</td>
<td>$E_{12}^2$</td>
</tr>
<tr>
<td>$E222z$</td>
<td>$E_{22}^2$</td>
</tr>
<tr>
<td>$E322z$</td>
<td>$E_{32}^2$</td>
</tr>
<tr>
<td>$E10z$</td>
<td>$E_{10}^z$</td>
</tr>
<tr>
<td>$E20z$</td>
<td>$E_{20}^z$</td>
</tr>
</tbody>
</table>
\( E30z \quad \varepsilon^0 \)
\( EBpq \quad \varepsilon \cdot B^{pq} \)
\( E0Um \quad \varepsilon^0 \cdot U \)
\( E0Wm \quad \varepsilon^0 \cdot W \)
\( bx0mz \quad \langle b_x \rangle_{xy} \)
\( by0mz \quad \langle b_y \rangle_{xy} \)
\( bz0mz \quad \langle b_z \rangle_{xy} \)
\( M11z \quad \langle M_{11} \rangle_{xy} \)
\( M22z \quad \langle M_{22} \rangle_{xy} \)
\( M33z \quad \langle M_{33} \rangle_{xy} \)

Module ‘testfield_meri.f90’

\( E11xy \quad E_{11xy} \)
\( E12xy \quad E_{12xy} \)
\( E13xy \quad E_{13xy} \)
\( E21xy \quad E_{21xy} \)
\( E22xy \quad E_{22xy} \)
\( E23xy \quad E_{23xy} \)
\( E31xy \quad E_{31xy} \)
\( E32xy \quad E_{32xy} \)
\( E33xy \quad E_{33xy} \)
\( E41xy \quad E_{41xy} \)
\( E42xy \quad E_{42xy} \)
\( E43xy \quad E_{43xy} \)
\( E51xy \quad E_{51xy} \)
\( E52xy \quad E_{52xy} \)
\( E53xy \quad E_{53xy} \)
\( E61xy \quad E_{61xy} \)
\( E62xy \quad E_{62xy} \)
\( E63xy \quad E_{63xy} \)
\( E71xy \quad E_{71xy} \)
\( E72xy \quad E_{72xy} \)
\( E73xy \quad E_{73xy} \)
\( E81xy \quad E_{81} \)
\( E82xy \quad E_{82} \)
\( E83xy \quad E_{83} \)
\( E91xy \quad E_{91} \)
\( E92xy \quad E_{92} \)
\( E93xy \quad E_{93} \)
\( a11xy \quad \alpha_{11} \)
\( a12xy \quad \alpha_{12} \)
\( a13xy \quad \alpha_{13} \)
\( a21xy \quad \alpha_{21} \)
\( a22xy \quad \alpha_{22} \)
\( a23xy \quad \alpha_{23} \)
\( a31xy \quad \alpha_{31} \)
\( a32xy \quad \alpha_{32} \)
\( a33xy \quad \alpha_{33} \)
\( b111xy \quad z_{111} \)
\( b121xy \quad z_{121} \)
### Parameters for 'print.in'

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_{131xy}$</td>
<td>$z_{131}$</td>
</tr>
<tr>
<td>$b_{211xy}$</td>
<td>$z_{211}$</td>
</tr>
<tr>
<td>$b_{221xy}$</td>
<td>$z_{221}$</td>
</tr>
<tr>
<td>$b_{231xy}$</td>
<td>$z_{231}$</td>
</tr>
<tr>
<td>$b_{311xy}$</td>
<td>$z_{311}$</td>
</tr>
<tr>
<td>$b_{321xy}$</td>
<td>$z_{321}$</td>
</tr>
<tr>
<td>$b_{331xy}$</td>
<td>$z_{331}$</td>
</tr>
<tr>
<td>$b_{112xy}$</td>
<td>$z_{112}$</td>
</tr>
<tr>
<td>$b_{122xy}$</td>
<td>$z_{122}$</td>
</tr>
<tr>
<td>$b_{132xy}$</td>
<td>$z_{132}$</td>
</tr>
<tr>
<td>$b_{212xy}$</td>
<td>$z_{212}$</td>
</tr>
<tr>
<td>$b_{222xy}$</td>
<td>$z_{222}$</td>
</tr>
<tr>
<td>$b_{232xy}$</td>
<td>$z_{232}$</td>
</tr>
<tr>
<td>$b_{312xy}$</td>
<td>$z_{312}$</td>
</tr>
<tr>
<td>$b_{322xy}$</td>
<td>$z_{322}$</td>
</tr>
<tr>
<td>$b_{332xy}$</td>
<td>$z_{332}$</td>
</tr>
</tbody>
</table>

**Module 'testfield_nonlin_z.f90'**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alp11</td>
<td>$\alpha_{11}$</td>
</tr>
<tr>
<td>alp21</td>
<td>$\alpha_{21}$</td>
</tr>
<tr>
<td>alp31</td>
<td>$\alpha_{31}$</td>
</tr>
<tr>
<td>alp12</td>
<td>$\alpha_{12}$</td>
</tr>
<tr>
<td>alp22</td>
<td>$\alpha_{22}$</td>
</tr>
<tr>
<td>alp32</td>
<td>$\alpha_{32}$</td>
</tr>
<tr>
<td>eta11</td>
<td>$\eta_{11k}$</td>
</tr>
<tr>
<td>eta21</td>
<td>$\eta_{21k}$</td>
</tr>
<tr>
<td>eta12</td>
<td>$\eta_{12k}$</td>
</tr>
<tr>
<td>eta22</td>
<td>$\eta_{22k}$</td>
</tr>
<tr>
<td>alpK</td>
<td>$\alpha^K$</td>
</tr>
<tr>
<td>alpM</td>
<td>$\alpha^M$</td>
</tr>
<tr>
<td>alpMK</td>
<td>$\alpha^{MK}$</td>
</tr>
<tr>
<td>phi11</td>
<td>$\phi_{11}$</td>
</tr>
<tr>
<td>phi21</td>
<td>$\phi_{21}$</td>
</tr>
<tr>
<td>phi12</td>
<td>$\phi_{12}$</td>
</tr>
<tr>
<td>phi22</td>
<td>$\phi_{22}$</td>
</tr>
<tr>
<td>phi32</td>
<td>$\phi_{32}$</td>
</tr>
<tr>
<td>psi11</td>
<td>$\psi_{11k}$</td>
</tr>
<tr>
<td>psi21</td>
<td>$\psi_{21k}$</td>
</tr>
<tr>
<td>psi12</td>
<td>$\psi_{12k}$</td>
</tr>
<tr>
<td>psi22</td>
<td>$\psi_{22k}$</td>
</tr>
<tr>
<td>phiK</td>
<td>$\phi^K$</td>
</tr>
<tr>
<td>phiM</td>
<td>$\phi^M$</td>
</tr>
<tr>
<td>phiMK</td>
<td>$\phi^{MK}$</td>
</tr>
<tr>
<td>alp11cc</td>
<td>$\alpha_{11} \cos^2 k_z$</td>
</tr>
<tr>
<td>alp21sc</td>
<td>$\alpha_{21} \sin k_z \cos k_z$</td>
</tr>
<tr>
<td>alp12cs</td>
<td>$\alpha_{12} \cos k_z \sin k_z$</td>
</tr>
<tr>
<td>alp22ss</td>
<td>$\alpha_{22} \sin^2 k_z$</td>
</tr>
<tr>
<td>eta11cc</td>
<td>$\eta_{11} \cos^2 k_z$</td>
</tr>
<tr>
<td>eta21sc</td>
<td>$\eta_{21} \sin k_z \cos k_z$</td>
</tr>
<tr>
<td>eta12cs</td>
<td>$\eta_{12} \cos k_z \sin k_z$</td>
</tr>
</tbody>
</table>
$\eta_{22} \sin^2 k z$

$\langle \sin 2 k z \nabla \cdot F \rangle$

$M_{11}$

$M_{22}$

$M_{33}$

$M_{11cc}$

$M_{11} \cos^2 k z$

$M_{11ss}$

$M_{11} \sin^2 k z$

$M_{22cc}$

$M_{22} \cos^2 k z$

$M_{22ss}$

$M_{22} \sin^2 k z$

$M_{12cs}$

$M_{12} \cos k z \sin k z$

$b_{x11}$

$b_{x21}$

$b_{x12}$

$b_{x22}$

$b_{x0}$

$b_{y11}$

$b_{y21}$

$b_{y12}$

$b_{y22}$

$b_{0y}$

$\langle u_{11}^2 \rangle^{1/2}$

$\langle u_{21}^2 \rangle^{1/2}$

$\langle u_{12}^2 \rangle^{1/2}$

$\langle u_{22}^2 \rangle^{1/2}$

$\langle j_{11}^2 \rangle^{1/2}$

$\langle j_{11}^2 \rangle^{1/2}$

$\langle b_{11}^2 \rangle^{1/2}$

$\langle b_{12}^2 \rangle^{1/2}$

$\langle b_{22}^2 \rangle^{1/2}$

$\langle u_{0x} \rangle$

$\langle u_{0y} \rangle$

$\langle u_{11x} \rangle$

$\langle u_{11y} \rangle$

$\langle u_{0}^2 \rangle^{1/2}$

$\langle b_{0y}^2 \rangle^{1/2}$

$\langle j_{0} \rangle$

$\langle \varepsilon_{11}^2 \rangle^{1/2}$

$\langle \varepsilon_{21}^2 \rangle^{1/2}$

$\langle \varepsilon_{12}^2 \rangle^{1/2}$

$\langle \varepsilon_{22}^2 \rangle^{1/2}$

$\langle \varepsilon_{0}^2 \rangle^{1/2}$

$\varepsilon_{x1}$

$\varepsilon_{x2}$

$\varepsilon_{x12}$

$\varepsilon_{x22}$

$\varepsilon_{x0}$

$\varepsilon_{y1}$

$\varepsilon_{y2}$

$\varepsilon_{y12}$

$\varepsilon_{y22}$

$\varepsilon_{y0}$
J.3 Parameters for 'print.in'

\( Ey_{12pt} \)
\( \mathcal{E}_y^{12} \)
\( Ey_{22pt} \)
\( \mathcal{E}_y^{22} \)
\( Ey_{0pt} \)
\( \mathcal{E}_y^0 \)
\( bamp \)
\( bamp \)
\( E_{111z} \)
\( \mathcal{E}_{11}^1 \)
\( E_{211z} \)
\( \mathcal{E}_{21}^1 \)
\( E_{311z} \)
\( \mathcal{E}_{31}^1 \)
\( E_{121z} \)
\( \mathcal{E}_{21}^{21} \)
\( E_{221z} \)
\( \mathcal{E}_{31}^{21} \)
\( E_{321z} \)
\( \mathcal{E}_{31}^{21} \)
\( E_{112z} \)
\( \mathcal{E}_{12}^1 \)
\( E_{212z} \)
\( \mathcal{E}_{12}^2 \)
\( E_{312z} \)
\( \mathcal{E}_{12}^{12} \)
\( E_{122z} \)
\( \mathcal{E}_{22}^1 \)
\( E_{222z} \)
\( \mathcal{E}_{22}^2 \)
\( E_{322z} \)
\( \mathcal{E}_{22}^{22} \)
\( E_{10z} \)
\( \mathcal{E}^0_1 \)
\( E_{20z} \)
\( \mathcal{E}^0_2 \)
\( E_{30z} \)
\( \mathcal{E}^0_3 \)
\( EBpq \)
\( \mathcal{E} \cdot \mathbf{B}^{pq} \)
\( E0Um \)
\( \mathcal{E}^0 \cdot \mathbf{U} \)
\( E0Wm \)
\( \mathcal{E}^0 \cdot \mathbf{W} \)
\( bx0mz \)
\( \langle b_x \rangle_{xy} \)
\( by0mz \)
\( \langle b_y \rangle_{xy} \)
\( bz0mz \)
\( \langle b_z \rangle_{xy} \)
\( M11z \)
\( \langle M_{11} \rangle_{xy} \)
\( M22z \)
\( \langle M_{22} \rangle_{xy} \)
\( M33z \)
\( \langle M_{33} \rangle_{xy} \)

Module 'testfield_x.f90'

\( alp_{11} \)
\( \alpha_{11} \)
\( alp_{21} \)
\( \alpha_{21} \)
\( alp_{31} \)
\( \alpha_{31} \)
\( alp_{12} \)
\( \alpha_{12} \)
\( alp_{22} \)
\( \alpha_{22} \)
\( alp_{32} \)
\( \alpha_{32} \)
\( eta_{11} \)
\( \eta_{11} k \)
\( eta_{21} \)
\( \eta_{21} k \)
\( eta_{12} \)
\( \eta_{12} k \)
\( eta_{22} \)
\( \eta_{22} k \)
\( alp_{11cc} \)
\( \alpha_{11} \cos^2 kx \)
\( alp_{21sc} \)
\( \alpha_{21} \sin kx \cos kx \)
\( alp_{12cs} \)
\( \alpha_{12} \cos kx \sin kx \)
\( alp_{22ss} \)
\( \alpha_{22} \sin^2 kx \)
\( eta_{11cc} \)
\( \eta_{11} \cos^2 kx \)
\( eta_{21sc} \)
\( \eta_{21} \sin kx \cos kx \)
\( eta_{12cs} \)
\( \eta_{12} \cos kx \sin kx \)
\( eta_{22ss} \)
\( \eta_{22} \sin^2 kx \)
\( alp_{11_x} \)
\( \alpha_{11} x \)
\( alp_{21_x} \)
\( \alpha_{21} x \)
alp12.x \quad \alpha_{12} x \\
alp22.x \quad \alpha_{22} x \\
etta11.x \quad \eta_{11} k x \\
etta21.x \quad \eta_{21} k x \\
etta12.x \quad \eta_{12} k x \\
etta22.x \quad \eta_{22} k x \\
alp11.x \quad \alpha_{11} x^2 \\
alp21.x \quad \alpha_{21} x^2 \\
alp12.x \quad \alpha_{12} x^2 \\
alp22.x \quad \alpha_{22} x^2 \\
etta11.x \quad \eta_{11} k x^2 \\
etta21.x \quad \eta_{21} k x^2 \\
etta12.x \quad \eta_{12} k x^2 \\
etta22.x \quad \eta_{22} k x^2 \\
 b11rms \quad \langle b_{11}^2 \rangle^{1/2} \\
b21rms \quad \langle b_{21}^2 \rangle^{1/2} \\
b12rms \quad \langle b_{12}^2 \rangle^{1/2} \\
b22rms \quad \langle b_{22}^2 \rangle^{1/2} \\
b0rms \quad \langle b_0^2 \rangle^{1/2} \\
 E11rms \quad \langle \mathcal{E}_{11}^2 \rangle^{1/2} \\
E21rms \quad \langle \mathcal{E}_{21}^2 \rangle^{1/2} \\
E12rms \quad \langle \mathcal{E}_{12}^2 \rangle^{1/2} \\
E22rms \quad \langle \mathcal{E}_{22}^2 \rangle^{1/2} \\
E0rms \quad \langle \mathcal{E}_0^2 \rangle^{1/2} \\
E111z \quad \mathcal{E}_1^{11} \\
E211z \quad \mathcal{E}_2^{11} \\
E311z \quad \mathcal{E}_3^{11} \\
E121z \quad \mathcal{E}_1^{21} \\
E221z \quad \mathcal{E}_2^{21} \\
E321z \quad \mathcal{E}_3^{21} \\
E112z \quad \mathcal{E}_1^{12} \\
E212z \quad \mathcal{E}_2^{12} \\
E312z \quad \mathcal{E}_3^{12} \\
E122z \quad \mathcal{E}_1^{22} \\
E222z \quad \mathcal{E}_2^{22} \\
E322z \quad \mathcal{E}_3^{22} \\
E10z \quad \mathcal{E}_1^0 \\
E20z \quad \mathcal{E}_2^0 \\
E30z \quad \mathcal{E}_3^0 \\
EBpq \quad \mathcal{E} \cdot B^{pq} \\
 bx0mz \quad \langle b_x \rangle_{xy} \\
 by0mz \quad \langle b_y \rangle_{xy} \\
 bz0mz \quad \langle b_z \rangle_{xy} \\
alp11x \quad \alpha_{11}(x, t) \\
alp21x \quad \alpha_{21}(x, t) \\
alp12x \quad \alpha_{12}(x, t) \\
alp22x \quad \alpha_{22}(x, t) \\
etta11x \quad \eta_{11}(x, t)
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>eta21x</td>
<td>$\eta_{21}(x, t)$</td>
</tr>
<tr>
<td>eta12x</td>
<td>$\eta_{12}(x, t)$</td>
</tr>
<tr>
<td>eta22x</td>
<td>$\eta_{22}(x, t)$</td>
</tr>
</tbody>
</table>

**Module 'testfield_xz.f90'**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{111z}$</td>
<td>$\mathcal{E}_{11}^{11}$</td>
</tr>
<tr>
<td>$E_{211z}$</td>
<td>$\mathcal{E}_{21}^{11}$</td>
</tr>
<tr>
<td>$E_{311z}$</td>
<td>$\mathcal{E}_{31}^{11}$</td>
</tr>
<tr>
<td>$E_{121z}$</td>
<td>$\mathcal{E}_{12}^{21}$</td>
</tr>
<tr>
<td>$E_{221z}$</td>
<td>$\mathcal{E}_{22}^{21}$</td>
</tr>
<tr>
<td>$E_{321z}$</td>
<td>$\mathcal{E}_{32}^{21}$</td>
</tr>
<tr>
<td>alp11</td>
<td>$\alpha_{11}$</td>
</tr>
<tr>
<td>alp21</td>
<td>$\alpha_{21}$</td>
</tr>
<tr>
<td>alp12</td>
<td>$\alpha_{12}$</td>
</tr>
<tr>
<td>alp22</td>
<td>$\alpha_{22}$</td>
</tr>
<tr>
<td>alp31</td>
<td>$\alpha_{31}$</td>
</tr>
<tr>
<td>alp32</td>
<td>$\alpha_{32}$</td>
</tr>
<tr>
<td>alp13</td>
<td>$\alpha_{13}$</td>
</tr>
<tr>
<td>alp23</td>
<td>$\alpha_{23}$</td>
</tr>
<tr>
<td>eta11</td>
<td>$\eta_{113k}$ or $\eta_{11k}$ if leta_rank2=T</td>
</tr>
<tr>
<td>eta21</td>
<td>$\eta_{213k}$ or $\eta_{21k}$ if leta_rank2=T</td>
</tr>
<tr>
<td>eta31</td>
<td>$\eta_{313k}$</td>
</tr>
<tr>
<td>eta12</td>
<td>$\eta_{123k}$ or $\eta_{12k}$ if leta_rank2=T</td>
</tr>
<tr>
<td>eta22</td>
<td>$\eta_{223k}$ or $\eta_{22k}$ if leta_rank2=T</td>
</tr>
<tr>
<td>eta32</td>
<td>$\eta_{323k}$</td>
</tr>
<tr>
<td>alp11cc</td>
<td>$\alpha_{11} \cos^2 k z$</td>
</tr>
<tr>
<td>alp21sc</td>
<td>$\alpha_{21} \sin k z \cos k z$</td>
</tr>
<tr>
<td>alp12cs</td>
<td>$\alpha_{12} \cos k z \sin k z$</td>
</tr>
<tr>
<td>alp22ss</td>
<td>$\alpha_{22} \sin^2 k z$</td>
</tr>
<tr>
<td>eta11cc</td>
<td>$\eta_{11} \cos^2 k z$</td>
</tr>
<tr>
<td>eta21sc</td>
<td>$\eta_{21} \sin k z \cos k z$</td>
</tr>
<tr>
<td>eta12cs</td>
<td>$\eta_{12} \cos k z \sin k z$</td>
</tr>
<tr>
<td>eta22ss</td>
<td>$\eta_{22} \sin^2 k z$</td>
</tr>
<tr>
<td>s2kzDFm</td>
<td>$\langle \sin 2k z \nabla \cdot F \rangle$</td>
</tr>
<tr>
<td>M11</td>
<td>$\mathcal{M}_{11}$</td>
</tr>
<tr>
<td>M22</td>
<td>$\mathcal{M}_{22}$</td>
</tr>
<tr>
<td>M33</td>
<td>$\mathcal{M}_{33}$</td>
</tr>
<tr>
<td>M11cc</td>
<td>$\mathcal{M}_{11} \cos^2 k z$</td>
</tr>
<tr>
<td>M11ss</td>
<td>$\mathcal{M}_{11} \sin^2 k z$</td>
</tr>
<tr>
<td>M22cc</td>
<td>$\mathcal{M}_{22} \cos^2 k z$</td>
</tr>
<tr>
<td>M22ss</td>
<td>$\mathcal{M}_{22} \sin^2 k z$</td>
</tr>
<tr>
<td>M12cs</td>
<td>$\mathcal{M}_{12} \cos k z \sin k z$</td>
</tr>
</tbody>
</table>
bx11pt  \ b_{x}^{11}
bx21pt  \ b_{x}^{21}
bx12pt  \ b_{x}^{12}
bx22pt  \ b_{x}^{22}
by0pt  \ b_{y}^{0}
by11pt  \ b_{y}^{11}
by21pt  \ b_{y}^{21}
by12pt  \ b_{y}^{12}
by22pt  \ b_{y}^{22}
ybamp
b11rms  \ \langle b_{x}^{11} \rangle^{1/2}
b21rms  \ \langle b_{x}^{21} \rangle^{1/2}
b12rms  \ \langle b_{x}^{12} \rangle^{1/2}
b22rms  \ \langle b_{x}^{22} \rangle^{1/2}
b0rms  \ \langle b_{y}^{0} \rangle^{1/2}
jb0m  \ \langle jb_{0} \rangle
E11rms  \ \langle E_{x}^{11} \rangle^{1/2}
E21rms  \ \langle E_{x}^{21} \rangle^{1/2}
E12rms  \ \langle E_{x}^{12} \rangle^{1/2}
E22rms  \ \langle E_{x}^{22} \rangle^{1/2}
E0rms  \ \langle E_{0} \rangle^{1/2}
Ex11pt  \ \xi^{11}_{x}
Ex21pt  \ \xi^{21}_{x}
Ex12pt  \ \xi^{12}_{x}
Ex22pt  \ \xi^{22}_{x}
Ex0pt  \ \xi^{0}
Ey11pt  \ \xi^{11}_{y}
Ey21pt  \ \xi^{21}_{y}
Ey12pt  \ \xi^{12}_{y}
Ey22pt  \ \xi^{22}_{y}
Ey0pt  \ \xi^{0}_{y}
bamp  \ bamp
alp11z  \ \alpha_{11}(z,t)
alp21z  \ \alpha_{21}(z,t)
alp12z  \ \alpha_{12}(z,t)
alp22z  \ \alpha_{22}(z,t)
alp13z  \ \alpha_{13}(z,t)
alp23z  \ \alpha_{23}(z,t)
etal11z  \ \eta_{11}(z,t)
etal21z  \ \eta_{21}(z,t)
etal12z  \ \eta_{12}(z,t)
etal22z  \ \eta_{22}(z,t)
E111z  \ \xi^{11}_{1}
E211z  \ \xi^{11}_{2}
E311z  \ \xi^{11}_{3}
E121z  \ \xi^{21}_{1}
E221z  \ \xi^{21}_{2}
E321z  \ \xi^{21}_{3}
Module ‘testflow_z.f90’

gal \quad \text{GAL-coefficients, couple } \overline{F} \text{ and } \overline{U}

aklam \quad \text{AKA-} \lambda \text{-tensor, couples } \overline{F} \text{ and } \overline{W} = \nabla \times \overline{U}

gamma \quad \gamma \text{-vector, couples } \overline{F} \text{ and } \nabla \cdot \overline{U}

nu \quad \nu \text{-tensor, couples } \overline{F} \text{ and } \partial^2 \overline{U}/\partial z^2

zeta \quad \zeta \text{-vector, couples } \overline{F} \text{ and } \overline{G}_z = \nabla_z \overline{H}

xi \quad \xi \text{-vector, couples } \overline{F} \text{ and } \partial^2 \overline{H}/\partial z^2

aklamQ \quad \text{aklam}^Q \text{-vector, couples } \overline{Q} \text{ and } \overline{W}

gammaQ \quad \gamma^Q \text{-scalar, couples } \overline{Q} \text{ and } \nabla \cdot \overline{U} = dU_z/dz

nuQ \quad \nu^Q \text{-vector, couples } \overline{Q} \text{ and } \partial^2 \overline{U}/\partial z^2

zetaQ \quad \zeta^Q \text{-scalar, couples } \overline{Q} \text{ and } \overline{G}_z

xiQ \quad \xi^Q \text{-scalar, couples } \overline{Q} \text{ and } \partial^2 \overline{H}/\partial z^2

ux0mz \quad \langle u_x \rangle_{xy}

uy0mz \quad \langle u_y \rangle_{xy}

uz0mz \quad \langle u_z \rangle_{xy}

Module ‘testperturb.f90’

alp11 \quad \alpha_{11}

alp21 \quad \alpha_{21}

alp31 \quad \alpha_{31}

alp12 \quad \alpha_{12}

alp22 \quad \alpha_{22}

alp32 \quad \alpha_{32}

eta11 \quad \eta_{113} k

eta21 \quad \eta_{213} k

eta31 \quad \eta_{313} k

eta12 \quad \eta_{123} k

eta22 \quad \eta_{223} k

eta32 \quad \eta_{323} k
Module ‘testscalar.f90’

\texttt{gam11} \gamma_1^{(1)} \\
\texttt{gam12} \gamma_2^{(1)} \\
\texttt{gam13} \gamma_3^{(1)} \\
\texttt{gam21} \gamma_1^{(2)} \\
\texttt{gam22} \gamma_2^{(2)} \\
\texttt{gam23} \gamma_3^{(2)} \\
\texttt{gam31} \gamma_1^{(3)} \\
\texttt{gam32} \gamma_2^{(3)} \\
\texttt{gam33} \gamma_3^{(3)} \\
\texttt{kap11} \kappa_{11} \\
\texttt{kap21} \kappa_{21} \\
\texttt{kap31} \kappa_{31} \\
\texttt{kap12} \kappa_{12} \\
\texttt{kap22} \kappa_{22} \\
\texttt{kap32} \kappa_{32} \\
\texttt{kap13} \kappa_{13} \\
\texttt{kap23} \kappa_{23} \\
\texttt{kap33} \kappa_{33} \\
\texttt{gam11z} \gamma_1^{(1)}(z,t) \\
\texttt{gam12z} \gamma_2^{(1)}(z,t) \\
\texttt{gam13z} \gamma_3^{(1)}(z,t) \\
\texttt{gam21z} \gamma_1^{(2)}(z,t) \\
\texttt{gam22z} \gamma_2^{(2)}(z,t) \\
\texttt{gam23z} \gamma_3^{(2)}(z,t) \\
\texttt{gam31z} \gamma_1^{(3)}(z,t) \\
\texttt{gam32z} \gamma_2^{(3)}(z,t) \\
\texttt{gam33z} \gamma_3^{(3)}(z,t) \\
\texttt{kap11z} \kappa_{11}(z,t) \\
\texttt{kap21z} \kappa_{21}(z,t) \\
\texttt{kap31z} \kappa_{31}(z,t) \\
\texttt{kap12z} \kappa_{12}(z,t) \\
\texttt{kap22z} \kappa_{22}(z,t) \\
\texttt{kap32z} \kappa_{32}(z,t) \\
\texttt{kap13z} \kappa_{13}(z,t) \\
\texttt{kap23z} \kappa_{23}(z,t) \\
\texttt{kap33z} \kappa_{33}(z,t) \\
\texttt{mgam33} \tilde{\gamma}_{33} \\
\texttt{mkap33} \tilde{\kappa}_{33} \\
\texttt{ngam33} \hat{\gamma}_{33} \\
\texttt{nkap33} \hat{\kappa}_{33} \\
\texttt{c1rms} \langle c_1^2 \rangle^{1/2} \\
\texttt{c2rms} \langle c_2^2 \rangle^{1/2} \\
\texttt{c3rms} \langle c_3^2 \rangle^{1/2} \\
\texttt{c4rms} \langle c_4^2 \rangle^{1/2} \\
\texttt{c5rms} \langle c_5^2 \rangle^{1/2}
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>c6rms</td>
<td>$(c_6)^{1/2}$</td>
</tr>
<tr>
<td>c1pt, c2pt, c3pt, c4pt, c5pt, c6pt</td>
<td>$c_i$</td>
</tr>
<tr>
<td>F11z, F21z, F31z, F12z, F22z, F32z</td>
<td>$F_i$</td>
</tr>
</tbody>
</table>

Module ‘testscalar_axisym.f90’

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>muc1, muc2</td>
<td>$\mu^{(c)}$</td>
</tr>
<tr>
<td>gamec</td>
<td>$\gamma^{(c)}$</td>
</tr>
<tr>
<td>kapcPERP1, kapcPERP2</td>
<td>$\kappa^{(1)}<em>\perp, \kappa^{(2)}</em>\perp$</td>
</tr>
<tr>
<td>kapcPARA</td>
<td>$\kappa^{(1)}_\parallel$</td>
</tr>
<tr>
<td>mucz, gamecz</td>
<td>$\mu^{(c)}(z,t)$, $\gamma^{(c)}(z,t)$</td>
</tr>
<tr>
<td>kapcPERPz, kapcPARAz</td>
<td>$\kappa^{(1)}<em>\perp(z,t), \kappa^{(1)}</em>\parallel(z,t)$</td>
</tr>
<tr>
<td>gam11, gam12, gam13</td>
<td>$\gamma^{(1)}_1, \gamma^{(1)}_2, \gamma^{(1)}_3$</td>
</tr>
<tr>
<td>gam21, gam22, gam23</td>
<td>$\gamma^{(2)}_1, \gamma^{(2)}_2, \gamma^{(2)}_3$</td>
</tr>
<tr>
<td>gam31, gam32, gam33</td>
<td>$\gamma^{(3)}_1, \gamma^{(3)}_2, \gamma^{(3)}_3$</td>
</tr>
<tr>
<td>kap11, kap21, kap31</td>
<td>$\kappa^{(1)}<em>{11}, \kappa^{(1)}</em>{21}, \kappa^{(1)}_{31}$</td>
</tr>
<tr>
<td>kap12, kap22, kap32</td>
<td>$\kappa^{(1)}<em>{12}, \kappa^{(1)}</em>{22}, \kappa^{(1)}_{32}$</td>
</tr>
<tr>
<td>kap13, kap23, kap33</td>
<td>$\kappa^{(1)}<em>{13}, \kappa^{(1)}</em>{23}, \kappa^{(1)}_{33}$</td>
</tr>
<tr>
<td>gam11z, gam12z, gam13z</td>
<td>$\gamma^{(1)}_1(z,t), \gamma^{(1)}_2(z,t), \gamma^{(1)}_3(z,t)$</td>
</tr>
<tr>
<td>gam21z, gam22z</td>
<td>$\gamma^{(2)}_1(z,t), \gamma^{(2)}_2(z,t)$</td>
</tr>
</tbody>
</table>
gam23z $\gamma_3^{(2)}(z, t)$
gam31z $\gamma_1^{(3)}(z, t)$
gam32z $\gamma_2^{(3)}(z, t)$
gam33z $\gamma_3^{(3)}(z, t)$
gam3z $\gamma^{(c)}(z, t)$
kap11z $\kappa_{11}(z, t)$
kap21z $\kappa_{21}(z, t)$
kap31z $\kappa_{31}(z, t)$
kap12z $\kappa_{12}(z, t)$
kap22z $\kappa_{22}(z, t)$
kap32z $\kappa_{32}(z, t)$
kap13z $\kappa_{13}(z, t)$
kap23z $\kappa_{23}(z, t)$
kap33z $\kappa_{33}(z, t)$
mgam33 $\check{\gamma}_{33}$
mkap33 $\check{\kappa}_{33}$
ngam33 $\hat{\gamma}_{33}$
nkap33 $\hat{\kappa}_{33}$
c1rms $\langle c_1^2 \rangle^{1/2}$
c2rms $\langle c_2^2 \rangle^{1/2}$
c3rms $\langle c_3^2 \rangle^{1/2}$
c4rms $\langle c_4^2 \rangle^{1/2}$
c5rms $\langle c_5^2 \rangle^{1/2}$
c6rms $\langle c_6^2 \rangle^{1/2}$
c1pt $c_1$
c2pt $c_2$
c3pt $c_3$
c4pt $c_4$
c5pt $c_5$
c6pt $c_6$
F11z $F_1^1$
F21z $F_2^1$
F31z $F_3^1$
F12z $F_1^2$
F22z $F_2^2$
F32z $F_3^2$

---

Module 'testscalar_simple.f90'

| gam11 | $\gamma_1^{(1)}$ |
gam12 | $\gamma_2^{(1)}$ |
gam13 | $\gamma_3^{(1)}$ |
gam21 | $\gamma_1^{(2)}$ |
gam22 | $\gamma_2^{(2)}$ |
gam23 | $\gamma_3^{(2)}$ |
gam31 | $\gamma_1^{(3)}$ |
gam32 | $\gamma_2^{(3)}$ |
gam33 | $\gamma_3^{(3)}$ |
kap11 | $\kappa_{11}$ |
J.3 Parameters for 'print.in'

\[
\begin{align*}
\text{kap21} & \quad \kappa_{21} \\
\text{kap31} & \quad \kappa_{31} \\
\text{kap12} & \quad \kappa_{12} \\
\text{kap22} & \quad \kappa_{22} \\
\text{kap32} & \quad \kappa_{32} \\
\text{kap13} & \quad \kappa_{13} \\
\text{kap23} & \quad \kappa_{23} \\
\text{kap33} & \quad \kappa_{33} \\
\text{gam11z} & \quad \gamma_{11}^{(1)}(z,t) \\
\text{gam12z} & \quad \gamma_{12}^{(1)}(z,t) \\
\text{gam13z} & \quad \gamma_{13}^{(1)}(z,t) \\
\text{gam21z} & \quad \gamma_{21}^{(2)}(z,t) \\
\text{gam22z} & \quad \gamma_{22}^{(2)}(z,t) \\
\text{gam23z} & \quad \gamma_{23}^{(2)}(z,t) \\
\text{gam31z} & \quad \gamma_{31}^{(3)}(z,t) \\
\text{gam32z} & \quad \gamma_{32}^{(3)}(z,t) \\
\text{gam33z} & \quad \gamma_{33}^{(3)}(z,t) \\
\text{kap11z} & \quad \kappa_{11}(z,t) \\
\text{kap21z} & \quad \kappa_{21}(z,t) \\
\text{kap31z} & \quad \kappa_{31}(z,t) \\
\text{kap12z} & \quad \kappa_{12}(z,t) \\
\text{kap22z} & \quad \kappa_{22}(z,t) \\
\text{kap32z} & \quad \kappa_{32}(z,t) \\
\text{kap13z} & \quad \kappa_{13}(z,t) \\
\text{kap23z} & \quad \kappa_{23}(z,t) \\
\text{kap33z} & \quad \kappa_{33}(z,t) \\
\text{mgam33} & \quad \tilde{\gamma}_{33} \\
\text{mkap33} & \quad \tilde{\kappa}_{33} \\
\text{ngam33} & \quad \hat{\gamma}_{33} \\
\text{nkap33} & \quad \hat{\kappa}_{33} \\
\text{c1rms} & \quad \langle c_1^2 \rangle^{1/2} \\
\text{c2rms} & \quad \langle c_2^2 \rangle^{1/2} \\
\text{c3rms} & \quad \langle c_3^2 \rangle^{1/2} \\
\text{c4rms} & \quad \langle c_4^2 \rangle^{1/2} \\
\text{c5rms} & \quad \langle c_5^2 \rangle^{1/2} \\
\text{c6rms} & \quad \langle c_6^2 \rangle^{1/2} \\
\text{c1pt} & \quad c_1^1 \\
\text{c2pt} & \quad c_2^2 \\
\text{c3pt} & \quad c_3^3 \\
\text{c4pt} & \quad c_4^4 \\
\text{c5pt} & \quad c_5^5 \\
\text{c6pt} & \quad c_6^6 \\
\text{F11z} & \quad \mathcal{F}_1^1 \\
\text{F21z} & \quad \mathcal{F}_2^1 \\
\text{F31z} & \quad \mathcal{F}_3^1 \\
\text{F12z} & \quad \mathcal{F}_1^2 \\
\text{F22z} & \quad \mathcal{F}_2^2 \\
\text{F32z} & \quad \mathcal{F}_3^2
\end{align*}
\]
Module `thermal_energy.f90`

- $TT_{\text{max}}$: $\max(T)$
- $TT_{\text{min}}$: $\min(T)$
- $ppm$: $\langle p \rangle$
- $TT_{\text{m}}$: $\langle T \rangle$
- $eth_{\text{m}}$: $\langle e_{\text{th}} \rangle = \langle c_v \rho T \rangle$ (mean thermal energy)
- $eth_{\text{tot}}$: $\int_V e_{\text{th}} \, dV$ (total thermal energy)
- $eth_{\text{min}}$: $\min e_{\text{th}}$
- $eth_{\text{max}}$: $\max e_{\text{th}}$
- $e_{\text{em}}$: $\langle e \rangle = \langle c_v T \rangle$ (mean internal energy)
- $et_{\text{tot}}$: $\langle e_{\text{th}} + \rho u^2/2 \rangle$

Module `visc_smagorinsky.f90`

- $nu_{\text{LES}}$: Mean value of Smagorinsky viscosity

Module `viscosity.f90`

- $nu_{\text{tdep}}$: Time-dependent viscosity
- $fvisc_{\text{m}}$: Mean value of viscous acceleration
- $fvisc_{\text{min}}$: Min value of viscous acceleration
- $fvisc_{\text{max}}$: Max value of viscous acceleration
- $fvisc_{\text{rms}}$: Rms value of viscous acceleration for the vis_xaver_range
- $num$: Mean value of viscosity
- $nusmaggm$: Mean value of Smagorinsky viscosity
- $nusmag_{\text{min}}$: Min value of Smagorinsky viscosity
- $nusmag_{\text{max}}$: Max value of Smagorinsky viscosity
- $nu_{\text{LES}}$: Mean value of Smagorinsky viscosity
- $visc_{\text{heat}}$: Mean value of viscous heating
- $q_{\text{fvisc}}$: $\langle q \cdot f_{\text{visc}} \rangle$
- $uf_{\text{visc}}$: $\langle u \cdot f_{\text{visc}} \rangle$
- $S_{\text{j2m}}$: $\langle S^2 \rangle$
- $epsK$: $\langle 2\nu_0 S^2 \rangle$
- $dt_{nu}$: $\delta t/[c_{\text{St}}, \delta x^2/\nu_{\text{max}}]$ (time step relative to viscous time step; see §5.15)
- $mesh_{\text{Remax}}$: Max mesh Reynolds number
- $Reshock$: Mesh Reynolds number at shock

J.4 List of parameters for `video.in`

The following table lists all (at the time of writing, May 10, 2018) possible inputs to the file `video.in`.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$uu$</td>
<td>Velocity vector $u$; writes all three components separately to files <code>$u[xyz].\{xz, yz, xy, xy2\}$</code></td>
</tr>
<tr>
<td>$u2$</td>
<td>Kinetic energy density $u^2$; writes <code>$u2.\{xz, yz, xy, xy2\}$</code></td>
</tr>
</tbody>
</table>
vorticity vector $\omega = \nabla \times u$; writes all three components separately to files `oo[xyz].{xz,yz,xy,xy2}'

enstrophy $\omega^2 = |\nabla \times u|^2$; writes `o2. {xz,yz,xy,xy2}'

$\nabla \cdot u$; writes `divu. {xz,yz,xy,xy2}'

Mach number squared $Ma^2$; writes `mach. {xz,yz,xy,xy2}'

$\ln\rho$; writes `lnrho. {xz,yz,xy,xy2}'

density $\rho$; writes `rho. {xz,yz,xy,xy2}'

entropy $s$; writes `ss. {xz,yz,xy,xy2}'

pressure $p$; writes `pp. {xz,yz,xy,xy2}'

shock viscosity $\nu_{\text{shock}}$; writes `shock. {xz,yz,xy,xy2}'

ionization fraction $y_H$; writes `yH. {xz,yz,xy,xy2}'

radiative heating rate $Q_{\text{rad}}$; writes `Qrad. {xz,yz,xy,xy2}'

surface intensity $I_{\text{surf}}$ (?); writes `Isurf.xz'

magnetic vector potential $A$; writes `aa[xyz].{xz,yz,xy,xy2}'

magnetic flux density $B$; writes `bb[xyz].{xz,yz,xy,xy2}'

magnetic energy density $B^2$; writes `b2. {xz,yz,xy,xy2}'

current density $j$; writes `jj[xyz].{xz,yz,xy,xy2}'

current density squared $j^2$; writes `j2. {xz,yz,xy,xy2}'

$j B$; writes `jb. {xz,yz,xy,xy2}'

inverse plasma beta $B^2/(2\mu_0 p)$; writes `beta1. {xz,yz,xy,xy2}'

Poynting vector $\eta j \times B - (u \times B) \times B/\mu_0$; writes `Poynting[xyz].{xz,yz,xy,xy2}'

magnetic helicity density $A \cdot B$; writes `ab[xyz].{xz,yz,xy,xy2}'

logarithmic density of passive scalar $\ln c$; writes `lncc. {xz,yz,xy,xy2}'

energy $e_{\text{cr}}$ of cosmic rays (?); writes `ec. {xz,yz,xy,xy2}'
### J.5 List of parameters for ‘phiaver.in’

The following table lists all (at the time of writing, November 2003) possible inputs to the file ‘phiaver.in’.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module ‘cdata.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>rcylmphi</td>
<td>cylindrical radius $r = \sqrt{x^2 + y^2}$ (useful for debugging azimuthal averages)</td>
</tr>
<tr>
<td>phimphi</td>
<td>azimuthal angle $\varphi = \arctan \frac{y}{x}$ (useful for debugging)</td>
</tr>
<tr>
<td>zmphi</td>
<td>$z$-coordinate (useful for debugging)</td>
</tr>
<tr>
<td>rmphi</td>
<td>spherical radius $r = \sqrt{x^2 + z^2}$ (useful for debugging)</td>
</tr>
<tr>
<td><strong>Module ‘hydro.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>urmphi</td>
<td>shorthand for $u_{\varphi}$, $u_\varphi$ and $u_{z\varphi}$ together</td>
</tr>
<tr>
<td>upmphi</td>
<td>shorthand for $u_{\varphi}$, $u_\varphi$ and $u_{z\varphi}$ together</td>
</tr>
<tr>
<td>uzmphi</td>
<td>shorthand for $u_{\varphi}$, $u_\varphi$ and $u_{z\varphi}$ together</td>
</tr>
<tr>
<td>ursphmphi</td>
<td>shorthand for $u_{\varphi}$, $u_\varphi$ and $u_{z\varphi}$ together</td>
</tr>
<tr>
<td>uthmphi</td>
<td>shorthand for $u_{\varphi}$, $u_\varphi$ and $u_{z\varphi}$ together</td>
</tr>
<tr>
<td>uusphmphi</td>
<td>shorthand for $u_{\varphi}$, $u_\varphi$ and $u_{z\varphi}$ together</td>
</tr>
<tr>
<td>u2mphi</td>
<td>shorthand for $u_{\varphi}$, $u_\varphi$ and $u_{z\varphi}$ together</td>
</tr>
<tr>
<td><strong>Module ‘density.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>lnrhomphi</td>
<td>$\langle \ln \rho \rangle_\varphi$</td>
</tr>
<tr>
<td>rhomphi</td>
<td>$\langle \rho \rangle_\varphi$</td>
</tr>
<tr>
<td><strong>Module ‘entropy.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>ssmphi</td>
<td>$\langle s \rangle_\varphi$</td>
</tr>
<tr>
<td>cs2mphi</td>
<td>$\langle c^2 \rangle_\varphi$</td>
</tr>
<tr>
<td><strong>Module ‘magnetic.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>jbmphi</td>
<td>$\langle J \cdot B \rangle_\varphi$</td>
</tr>
<tr>
<td>brmphi</td>
<td>$\langle B_{\varphi} \rangle_\varphi$ [cyl. polar coords $(\varphi, z)$]</td>
</tr>
<tr>
<td>bpmphi</td>
<td>$\langle B_\varphi \rangle_\varphi$</td>
</tr>
<tr>
<td>bzmphi</td>
<td>$\langle B_z \rangle_\varphi$</td>
</tr>
<tr>
<td>bbsphmphi</td>
<td>shorthand for $brmphi$, $bpmphi$ and $bzmphi$ together</td>
</tr>
<tr>
<td>bssphmphi</td>
<td>shorthand for $brmphi$, $bpmphi$ and $bzmphi$ together</td>
</tr>
<tr>
<td>b2mphi</td>
<td>$\langle B^2 \rangle_\varphi$</td>
</tr>
<tr>
<td>brsphmphi</td>
<td>$\langle B_{\varphi} \rangle_\varphi$</td>
</tr>
<tr>
<td>bthmphi</td>
<td>$\langle B_\varphi \rangle_\varphi$</td>
</tr>
<tr>
<td><strong>Module ‘anelastic.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>lnrhomphi</td>
<td>$\langle \ln \rho \rangle_\varphi$</td>
</tr>
<tr>
<td>rhomphi</td>
<td>$\langle \rho \rangle_\varphi$</td>
</tr>
<tr>
<td><strong>Module ‘entropy_anelastic.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>ssmphi</td>
<td>$\langle s \rangle_\varphi$</td>
</tr>
</tbody>
</table>
J.6 List of parameters for ‘xyaver.in’

The following table lists possible inputs to the file ‘xyaver.in’. This list is not complete and maybe outdated.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>u2mz</td>
<td>$\langle u^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>o2mz</td>
<td>$\langle W^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>divu2mz</td>
<td>$\langle (\nabla \cdot u)^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>curlru2mz</td>
<td>$\langle (\nabla \times uU)^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>divru2mz</td>
<td>$\langle (\nabla \cdot uU)^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>fmasszmz</td>
<td>$\langle gu_z \rangle_{xy}$</td>
</tr>
<tr>
<td>fkinzmz</td>
<td>$\langle \frac{1}{2}gu^2u_z \rangle_{xy}$</td>
</tr>
<tr>
<td>uxmz</td>
<td>$\langle u_x \rangle_{xy}$ (horiz. averaged $x$ velocity)</td>
</tr>
<tr>
<td>uymz</td>
<td>$\langle u_y \rangle_{xy}$</td>
</tr>
<tr>
<td>uzmz</td>
<td>$\langle u_z \rangle_{xy}$</td>
</tr>
<tr>
<td>uzupmz</td>
<td>$\langle u_z^+ \rangle_{xy}$</td>
</tr>
<tr>
<td>uzdownmz</td>
<td>$\langle u_z^- \rangle_{xy}$</td>
</tr>
<tr>
<td>ruzupmz</td>
<td>$\langle gu_z^+ \rangle_{xy}$</td>
</tr>
<tr>
<td>ruzdownmz</td>
<td>$\langle gu_z^- \rangle_{xy}$</td>
</tr>
<tr>
<td>divumz</td>
<td>$\langle \text{div } u \text{ } \rangle_{xy}$</td>
</tr>
<tr>
<td>uzddivumz</td>
<td>$\langle u_z \text{div } u \text{ } \rangle_{xy}$</td>
</tr>
<tr>
<td>oxmz</td>
<td>$\langle \omega_x \rangle_{xy}$</td>
</tr>
<tr>
<td>oymz</td>
<td>$\langle \omega_y \rangle_{xy}$</td>
</tr>
<tr>
<td>ozmz</td>
<td>$\langle \omega_z \rangle_{xy}$</td>
</tr>
<tr>
<td>ux2mz</td>
<td>$\langle u_x^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>uy2mz</td>
<td>$\langle u_y^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>uz2mz</td>
<td>$\langle u_z^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>ox2mz</td>
<td>$\langle \omega_x^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>oy2mz</td>
<td>$\langle \omega_y^2 \rangle_{xy}$</td>
</tr>
<tr>
<td>oz2mz</td>
<td>$\langle \omega_z^2 \rangle_{xy}$</td>
</tr>
</tbody>
</table>
\[
\begin{align*}
\text{ruuxmz} & \quad \langle \rho u_x \rangle_{xy} \\
\text{ruymz} & \quad \langle \rho u_y \rangle_{xy} \\
\text{ruzmz} & \quad \langle \rho u_z \rangle_{xy} \\
\text{ruux2mz} & \quad \langle \rho u_x^2 \rangle_{xy} \\
\text{ruy2mz} & \quad \langle \rho u_y^2 \rangle_{xy} \\
\text{ruuz2mz} & \quad \langle \rho u_z^2 \rangle_{xy} \\
\text{uxuymz} & \quad \langle u_x u_y \rangle_{xy} \\
\text{uxuzmz} & \quad \langle u_x u_z \rangle_{xy} \\
\text{uyuzmz} & \quad \langle u_y u_z \rangle_{xy} \\
\text{ruyuymz} & \quad \langle \rho u_x u_y \rangle_{xy} \\
\text{ruuxzmz} & \quad \langle \rho u_x u_z \rangle_{xy} \\
\text{ruyuzzmz} & \quad \langle \rho u_y u_z \rangle_{xy} \\
\text{oxuxzmz} & \quad \langle \omega_x u_{x,x} \rangle_{xy} \\
\text{oyuxymz} & \quad \langle \omega_y u_{x,y} \rangle_{xy} \\
\text{oxuyzmz} & \quad \langle \omega_u u_{y,x} \rangle_{xy} \\
\text{oxyuzmz} & \quad \langle \omega_u u_{z,x} \rangle_{xy} \\
\text{uyuxzmz} & \quad \langle \omega_x u_{y,z} \rangle_{xy} \\
\text{uyuymz} & \quad \langle \omega_y u_{y,z} \rangle_{xy} \\
\text{uyuzmz} & \quad \langle \omega_z u_{z,y} \rangle_{xy} \\
\text{ox2mz} & \quad \langle \omega_x \nabla \cdot u \rangle_{xy} \\
\text{oy2mz} & \quad \langle \omega_y \nabla \cdot u \rangle_{xy} \\
\text{o2mz} & \quad \langle \omega_z \nabla \cdot u \rangle_{xy} \\
\text{accpowzmz} & \quad \langle (u_z D u_z / D t)^2 \rangle_{xy} \\
\text{accpowzupmz} & \quad \langle (u_z D u_z / D t)^2 \rangle_{xy} \\
\text{accpowzdownmz} & \quad \langle (u_z D u_z / D t)^2 \rangle_{xy} \\
\text{fkinmz} & \quad \left\{ \frac{1}{2} \rho u^2 \right\}_{xy} \\
\text{R} & \quad \left\{ \frac{\partial (\omega S_{ij})}{\partial x_j} \right\}_{xy} \\
\text{Rem} & \quad \left\{ \frac{\partial}{\partial x_j} (\omega u^2) \right\}_{xy} \\
\text{oguxzmz} & \quad \langle (\omega \cdot \nabla u)_{x} \rangle_{xy} \\
\text{oguymz} & \quad \langle (\omega \cdot \nabla u)_{y} \rangle_{xy} \\
\text{oguzmz} & \quad \langle (\omega \cdot \nabla u)_{z} \rangle_{xy} \\
\text{ogux2mz} & \quad \langle (\omega \cdot \nabla u^2)_{x} \rangle_{xy} \\
\text{oguy2mz} & \quad \langle (\omega \cdot \nabla u^2)_{y} \rangle_{xy} \\
\text{oguz2mz} & \quad \langle (\omega \cdot \nabla u^2)_{z} \rangle_{xy} \\
\text{oxdivu} & \quad \langle \omega_x \nabla \cdot u \rangle_{xy} \\
\text{oydivu} & \quad \langle \omega_y \nabla \cdot u \rangle_{xy} \\
\text{ozdivu} & \quad \langle \omega_z \nabla \cdot u \rangle_{xy} \\
\text{oxdivu2mz} & \quad \langle (\omega_x \nabla \cdot u^2) \rangle_{xy} \\
\text{oydivu2mz} & \quad \langle (\omega_y \nabla \cdot u^2) \rangle_{xy} \\
\text{ozdivu2mz} & \quad \langle (\omega_z \nabla \cdot u^2) \rangle_{xy} \\
\text{accpowzmz} & \quad \langle (u_z D u_z / D t)^2 \rangle_{xy} \\
\text{accpowzupmz} & \quad \langle (u_z D u_z / D t)^2 \rangle_{xy} \\
\text{accpowzdownmz} & \quad \langle (u_z D u_z / D t)^2 \rangle_{xy} \\
\text{fkinmz} & \quad \left\{ \frac{1}{2} \rho u^2 u_z \right\}_{yz} \\
\end{align*}
\]
## Parameters for `xyaver.in`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gzlnrhomp</code></td>
<td>$\langle \nabla_z \ln \rho \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uglnhomp</code></td>
<td>$\langle u \cdot \nabla \ln \rho \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>ugrhomp</code></td>
<td>$\langle u \cdot \nabla \rho \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uyglnhomp</code></td>
<td>$\langle u_y \nabla_z \ln \rho \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uzgylnhomp</code></td>
<td>$\langle u_z \nabla_y \ln \rho \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>rho2mx</code></td>
<td>$\langle \rho^2 \rangle_{yz}$</td>
<td></td>
</tr>
<tr>
<td><code>fradz</code></td>
<td>$\langle F_{\text{rad}} \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>fconvz</code></td>
<td>$\langle c_p u_z T \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>ssmz</code></td>
<td>$\langle s \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>ss2mz</code></td>
<td>$\langle s^2 \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>ppmz</code></td>
<td>$\langle p \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>TTmz</code></td>
<td>$\langle T \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>TT2mz</code></td>
<td>$\langle T^2 \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uxTTmz</code></td>
<td>$\langle u_x T \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uyTTmz</code></td>
<td>$\langle u_y T \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uzTTmz</code></td>
<td>$\langle u_z T \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>gTxgsxmq</code></td>
<td>$\langle (\nabla T \times \nabla s)<em>x \rangle</em>{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>gTxgsymq</code></td>
<td>$\langle (\nabla T \times \nabla s)<em>y \rangle</em>{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>gTxgsmq</code></td>
<td>$\langle (\nabla T \times \nabla s)<em>z \rangle</em>{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>gTxgsx2mq</code></td>
<td>$\langle (\nabla T \times \nabla s)<em>x^2 \rangle</em>{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>gTxgsym2mq</code></td>
<td>$\langle (\nabla T \times \nabla s)<em>y^2 \rangle</em>{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>gTxgsm2mq</code></td>
<td>$\langle (\nabla T \times \nabla s)<em>z^2 \rangle</em>{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>fradz_kramers</code></td>
<td>$F_{\text{rad}}$ (from Kramers' opacity)</td>
<td></td>
</tr>
<tr>
<td><code>fradz_Kprof</code></td>
<td>$F_{\text{rad}}$ (from Kprof)</td>
<td></td>
</tr>
<tr>
<td><code>fradz_constchi</code></td>
<td>$F_{\text{rad}}$ (from chi_const)</td>
<td></td>
</tr>
<tr>
<td><code>fturbz</code></td>
<td>$\langle \rho T(x \cdot \nabla s) \rangle_{xy}$ (turbulent heat flux)</td>
<td></td>
</tr>
<tr>
<td><code>fturbtxz</code></td>
<td>$\langle \rho T(x \cdot \nabla s)<em>x \rangle</em>{xy}$ (turbulent heat flux)</td>
<td></td>
</tr>
<tr>
<td><code>fturbmz</code></td>
<td>$\langle \rho T(x \cdot \nabla s)<em>z \rangle</em>{xy}$ (turbulent heat flux)</td>
<td></td>
</tr>
<tr>
<td><code>fturbfz</code></td>
<td>$\langle \rho T(x \cdot \nabla s')<em>x \rangle</em>{xy}$ (turbulent heat flux)</td>
<td></td>
</tr>
<tr>
<td><code>dcoolz</code></td>
<td>surface cooling flux</td>
<td></td>
</tr>
<tr>
<td><code>heatmz</code></td>
<td>heating</td>
<td></td>
</tr>
<tr>
<td><code>Kkramersmz</code></td>
<td>$\langle K_0 T^3 (3 - b) / \rho \delta (a + 1) \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>ethmz</code></td>
<td>$\langle \rho e \rangle_{xy}$</td>
<td></td>
</tr>
</tbody>
</table>

## Module `magnetic.f90`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expression</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>axmz</code></td>
<td>$\langle A_x \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>aymz</code></td>
<td>$\langle A_y \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>azmz</code></td>
<td>$\langle A_z \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>abuxmz</code></td>
<td>$\langle (A \cdot B) u_x \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>abuyzm</code></td>
<td>$\langle (A \cdot B) u_y \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>abuzmz</code></td>
<td>$\langle (A \cdot B) u_z \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uabxmx</code></td>
<td>$\langle (u \cdot A) B_x \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uabymz</code></td>
<td>$\langle (u \cdot A) B_y \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>uabzmz</code></td>
<td>$\langle (u \cdot A) B_z \rangle_{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>bbxmx</code></td>
<td>$\langle B'<em>x \rangle</em>{xy}$</td>
<td></td>
</tr>
<tr>
<td><code>bbyzmz</code></td>
<td>$\langle B'<em>y \rangle</em>{xy}$</td>
<td></td>
</tr>
</tbody>
</table>
bbzmz $\langle B_z \rangle_{xy}$
bxzmz $\langle B_x \rangle_{xy}$
bymz $\langle B_y \rangle_{xy}$
bzzmz $\langle B_z \rangle_{xy}$
jxmz $\langle J_x \rangle_{xy}$
jymz $\langle J_y \rangle_{xy}$
jzmz $\langle J_z \rangle_{xy}$
Exmz $\langle E_x \rangle_{xy}$
Eyzmz $\langle E_y \rangle_{xy}$
Ezmz $\langle E_z \rangle_{xy}$
bx2mz $\langle B^2_x \rangle_{xy}$
by2mz $\langle B^2_y \rangle_{xy}$
bzmz $\langle B^2_z \rangle_{xy}$
bx2rmz $\langle B^2_x/\rho \rangle_{xy}$
by2rmz $\langle B^2_y/\rho \rangle_{xy}$
Hz2mz $\langle B^2_z/\rho \rangle_{xy}$
beta1mz $\langle (B^2/2\mu_0)/\rho \rangle_{xy}$
betamz $\langle \beta \rangle_{xy}$
beta2mz $\langle \beta^2 \rangle_{xy}$
jbzmz $\langle J \cdot B \rangle_{xy}$
d6abmz $\langle \nabla^6 A \cdot B \rangle_{xy}$
d6amz1 $\langle \nabla^6 A \rangle_{xy} |_x$
d6amz2 $\langle \nabla^6 A \rangle_{xy} |_y$
d6amz3 $\langle \nabla^6 A \rangle_{xy} |_z$
abzmz $\langle A \cdot B \rangle_{xy}$
ubzmz $\langle u \cdot B \rangle_{xy}$
ubzmz $\langle u \cdot A \rangle_{xy}$
uxbxzmz $\langle u_x b_x \rangle_{xy}$
uybzmz $\langle u_y b_y \rangle_{xy}$
uzbzmz $\langle u_z b_z \rangle_{xy}$
uxbyzmz $\langle u_x b_y \rangle_{xy}$
uybyzmz $\langle u_y b_y \rangle_{xy}$
uzbyzmz $\langle u_z b_y \rangle_{xy}$
uxbzmz $\langle u_x b_z \rangle_{xy}$
uybzmz $\langle u_y b_z \rangle_{xy}$
uzbzmz $\langle u_z b_z \rangle_{xy}$
examz1 $\langle E \times A \rangle_{xy} |_x$
examz2 $\langle E \times A \rangle_{xy} |_y$
examz3 $\langle E \times A \rangle_{xy} |_z$
e3xamz1 $\langle E_{hyper3} \times A \rangle_{xy} |_x$
e3xamz2 $\langle E_{hyper3} \times A \rangle_{xy} |_y$
e3xamz3 $\langle E_{hyper3} \times A \rangle_{xy} |_z$
etatotalzm $\langle \eta \rangle_{xy}$
bbzmz $\langle B_x B_y \rangle_{xy}$
bbzmz $\langle B_x B_z \rangle_{xy}$
bbzmz $\langle B_y B_z \rangle_{xy}$
bbzmz $\langle B^2 \rangle_{xy}$
bbzmz $\langle B^2 \rangle_{xy}$
J.6 Parameters for 'xyaver.in'

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>(j2mz)</td>
<td>(\langle j^2 \rangle_{xy})</td>
</tr>
<tr>
<td>poynzmz</td>
<td>Averaged poynting flux in z direction</td>
</tr>
<tr>
<td>epsMmz</td>
<td>(\langle \eta\mu_0j^2 \rangle_{xy})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module 'bfield.f90'</th>
</tr>
</thead>
<tbody>
<tr>
<td>(bmz)</td>
</tr>
<tr>
<td>(b2mz)</td>
</tr>
<tr>
<td>(bxmz)</td>
</tr>
<tr>
<td>(bymz)</td>
</tr>
<tr>
<td>(bzmz)</td>
</tr>
<tr>
<td>(bx2mz)</td>
</tr>
<tr>
<td>(by2mz)</td>
</tr>
<tr>
<td>(bz2mz)</td>
</tr>
<tr>
<td>(bxbymz)</td>
</tr>
<tr>
<td>(bxbzmz)</td>
</tr>
<tr>
<td>(bybzmz)</td>
</tr>
<tr>
<td>(betamz)</td>
</tr>
<tr>
<td>(beta2mz)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module 'density_stratified.f90'</th>
</tr>
</thead>
<tbody>
<tr>
<td>(drhomz)</td>
</tr>
<tr>
<td>(drho2mz)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module 'gravity_simple.f90'</th>
</tr>
</thead>
<tbody>
<tr>
<td>(epotmz)</td>
</tr>
<tr>
<td>(epotuzmz)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module 'magnetic_shearboxJ.f90'</th>
</tr>
</thead>
<tbody>
<tr>
<td>(axmz)</td>
</tr>
<tr>
<td>(aymz)</td>
</tr>
<tr>
<td>(azmz)</td>
</tr>
<tr>
<td>(abuxmz)</td>
</tr>
<tr>
<td>(abuymz)</td>
</tr>
<tr>
<td>(abuzmz)</td>
</tr>
<tr>
<td>(uabxmz)</td>
</tr>
<tr>
<td>(uabymz)</td>
</tr>
<tr>
<td>(uabzmz)</td>
</tr>
<tr>
<td>(bbxmz)</td>
</tr>
<tr>
<td>(bbymz)</td>
</tr>
<tr>
<td>(bbzmz)</td>
</tr>
<tr>
<td>(bxmz)</td>
</tr>
<tr>
<td>(bymz)</td>
</tr>
<tr>
<td>(bzmz)</td>
</tr>
<tr>
<td>(jxmz)</td>
</tr>
<tr>
<td>(jymz)</td>
</tr>
<tr>
<td>(jzmxz)</td>
</tr>
<tr>
<td>(Exmz)</td>
</tr>
<tr>
<td>(Eymz)</td>
</tr>
<tr>
<td>(Ezmz)</td>
</tr>
</tbody>
</table>
\[ \langle B_z \rangle_{xy} \]
\[ \langle B_y \rangle_{xy} \]
\[ \langle B_z \rangle_{xy} \]
\[ \langle B_x \rangle_{xy} \]
\[ \langle B_y \rangle_{xy} \]
\[ \langle B_z \rangle_{xy} \]
\[ \langle B_x \rangle_{xy} \]
\[ \langle (B^2/2\mu_0)/p \rangle_{xy} \]
\[ \langle \beta \rangle_{xy} \]
\[ \langle \beta^2 \rangle_{xy} \]
\[ \langle J \cdot B \rangle_{xy} \]
\[ \langle \nabla^6 A \cdot B \rangle_{xy} \]
\[ \langle \nabla^6 A \rangle_{xy} \]
\[ \langle \nabla^6 A \rangle_{xy} \]
\[ \langle A \cdot B \rangle_{xy} \]
\[ \langle u \cdot B \rangle_{xy} \]
\[ \langle u \cdot A \rangle_{xy} \]
\[ \langle u_x b_x \rangle_{xy} \]
\[ \langle u_y b_x \rangle_{xy} \]
\[ \langle u_z b_x \rangle_{xy} \]
\[ \langle u_x b_y \rangle_{xy} \]
\[ \langle u_y b_y \rangle_{xy} \]
\[ \langle u_z b_y \rangle_{xy} \]
\[ \langle u_x b_z \rangle_{xy} \]
\[ \langle u_y b_z \rangle_{xy} \]
\[ \langle u_z b_z \rangle_{xy} \]
\[ \langle j \rangle_{xy} \]
\[ \langle E \times A \rangle_{xy} \]
\[ \langle E \times A \rangle_{xy} \]
\[ \langle E \times A \rangle_{xy} \]
\[ \langle E_{\text{hyper}3} \times A \rangle_{xy} \]
\[ \langle E_{\text{hyper}3} \times A \rangle_{xy} \]
\[ \langle E_{\text{hyper}3} \times A \rangle_{xy} \]
\[ \langle \eta \rangle_{xy} \]
\[ \langle B_x B_y \rangle_{xy} \]
\[ \langle B_x B_z \rangle_{xy} \]
\[ \langle B_y B_z \rangle_{xy} \]
\[ \langle B^2 \rangle_{xy} \]
\[ \langle B^2 \rangle_{xy} \]
\[ \langle j^2 \rangle_{xy} \]
\[ \langle \text{Averaged Poynting flux in z direction} \rangle_{xy} \]
\[ \langle \eta \mu_0 j^2 \rangle_{xy} \]

---

**Module 'meanfield.f90'**

\[ \langle q_p \rangle_{xy} \]

---

**Module 'shock_highorder.f90'**

---

**Module 'temperature_ideal_gas.f90'**

\[ \langle p \rangle_{xy} \]
### Parameters for 'xzaver.in'

The following table lists possible inputs to the file `xzaver.in`. This list is not complete and maybe outdated.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uxmy</code></td>
<td>( \langle u_x \rangle_{xz} )</td>
</tr>
<tr>
<td><code>uymy</code></td>
<td>( \langle u_y \rangle_{xz} )</td>
</tr>
<tr>
<td><code>uzmy</code></td>
<td>( \langle u_z \rangle_{xz} )</td>
</tr>
<tr>
<td><code>oumy</code></td>
<td>( \langle \omega \cdot u \rangle_{xz} )</td>
</tr>
<tr>
<td><code>rhomy</code></td>
<td>( \langle \rho \rangle_{xz} )</td>
</tr>
</tbody>
</table>

**Module 'hydro.f90'**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TTmz</code></td>
<td>( \langle T \rangle_{xy} )</td>
</tr>
<tr>
<td><code>etzmz</code></td>
<td>( \langle \epsilon \rangle_{xy} )</td>
</tr>
<tr>
<td><code>fresxmz</code></td>
<td>( \langle \nabla p \rangle_{xy} )</td>
</tr>
<tr>
<td><code>fresymz</code></td>
<td>( \langle \nabla p \rangle_{xy} )</td>
</tr>
<tr>
<td><code>freszmz</code></td>
<td>( \langle \nabla p \rangle_{xy} )</td>
</tr>
<tr>
<td><code>TT2mz</code></td>
<td>( \langle T^2 \rangle_{xy} )</td>
</tr>
<tr>
<td><code>uxTmz</code></td>
<td>( \langle u_x T \rangle_{xy} )</td>
</tr>
<tr>
<td><code>uyTmz</code></td>
<td>( \langle u_y T \rangle_{xy} )</td>
</tr>
<tr>
<td><code>uzTmz</code></td>
<td>( \langle u_z T \rangle_{xy} )</td>
</tr>
<tr>
<td><code>fradmz</code></td>
<td>( \langle F_{rad} \rangle_{xy} )</td>
</tr>
</tbody>
</table>

**Module 'temperature_ionization.f90'**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>puzmz</code></td>
<td>( \langle pu_z \rangle_{xy} )</td>
</tr>
<tr>
<td><code>pr1mz</code></td>
<td>( \langle p/\rho \rangle_{xy} )</td>
</tr>
<tr>
<td><code>eruzmz</code></td>
<td>( \langle c_p u_z \rangle_{xy} )</td>
</tr>
<tr>
<td><code>ffakez</code></td>
<td>( \langle c_p u_z T \rangle_{xy} )</td>
</tr>
<tr>
<td><code>mumz</code></td>
<td>( \langle \mu \rangle_{xy} )</td>
</tr>
<tr>
<td><code>TTmz</code></td>
<td>( \langle T \rangle_{xy} )</td>
</tr>
<tr>
<td><code>ssmz</code></td>
<td>( \langle s \rangle_{xy} )</td>
</tr>
<tr>
<td><code>eemz</code></td>
<td>( \langle e \rangle_{xy} )</td>
</tr>
<tr>
<td><code>ppmz</code></td>
<td>( \langle \rho \rangle_{xy} )</td>
</tr>
</tbody>
</table>

**Module 'thermal_energy.f90'**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ppmz</code></td>
<td>( \langle \rho \rangle_{xy} )</td>
</tr>
<tr>
<td><code>TTmz</code></td>
<td>( \langle T \rangle_{xy} )</td>
</tr>
</tbody>
</table>

**Module 'viscosity.f90'**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fviscmz</code></td>
<td>( 2\nu u_i S_{iz} ) ( (z)-component of viscous flux)</td>
</tr>
<tr>
<td><code>fviscsmmz</code></td>
<td>( 2\nu_{Smag} u_i S_{iz} ) ( (z)-component of viscous flux)</td>
</tr>
<tr>
<td><code>epsKmz</code></td>
<td>( 2\nu S^2 ) ( ) ( )</td>
</tr>
</tbody>
</table>

**Module 'density.f90'**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rhomy</code></td>
<td>( \langle \rho \rangle_{xz} )</td>
</tr>
</tbody>
</table>
### Module ‘entropy.f90’

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssmy</td>
<td>$\langle s \rangle_{xz}$</td>
</tr>
<tr>
<td>pppmy</td>
<td>$\langle p \rangle_{xz}$</td>
</tr>
<tr>
<td>TTmy</td>
<td>$\langle T \rangle_{xz}$</td>
</tr>
</tbody>
</table>

### Module ‘magnetic.f90’

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bxmy</td>
<td>$\langle B_x \rangle_{xz}$</td>
</tr>
<tr>
<td>bymy</td>
<td>$\langle B_y \rangle_{xz}$</td>
</tr>
<tr>
<td>bzymy</td>
<td>$\langle B_z \rangle_{xz}$</td>
</tr>
<tr>
<td>bx2my</td>
<td>$\langle B_x^2 \rangle_{xz}$</td>
</tr>
<tr>
<td>by2my</td>
<td>$\langle B_y^2 \rangle_{xz}$</td>
</tr>
<tr>
<td>b2my</td>
<td>$\langle B_z^2 \rangle_{xz}$</td>
</tr>
<tr>
<td>bxbymy</td>
<td>$\langle B_x B_y \rangle_{xz}$</td>
</tr>
<tr>
<td>bxbyzm</td>
<td>$\langle B_x B_z \rangle_{xz}$</td>
</tr>
<tr>
<td>bybzm</td>
<td>$\langle B_y B_z \rangle_{xz}$</td>
</tr>
</tbody>
</table>

### Module ‘density_stratified.f90’

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>drhomy</td>
<td>$\langle (\Delta \rho/\rho_0) \rangle_{xz}$</td>
</tr>
<tr>
<td>drho2my</td>
<td>$\langle (\Delta \rho/\rho_0)^2 \rangle_{xz}$</td>
</tr>
</tbody>
</table>

### Module ‘gravity_simple.f90’

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>epotmy</td>
<td>$\langle \phi \Phi_{grav} \rangle_{xz}$</td>
</tr>
</tbody>
</table>

### Module ‘magnetic_shearboxJ.f90’

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bxmy</td>
<td>$\langle B_x \rangle_{xz}$</td>
</tr>
<tr>
<td>bymy</td>
<td>$\langle B_y \rangle_{xz}$</td>
</tr>
<tr>
<td>bzymy</td>
<td>$\langle B_z \rangle_{xz}$</td>
</tr>
<tr>
<td>bx2my</td>
<td>$\langle B_x^2 \rangle_{xz}$</td>
</tr>
<tr>
<td>by2my</td>
<td>$\langle B_y^2 \rangle_{xz}$</td>
</tr>
<tr>
<td>b2my</td>
<td>$\langle B_z^2 \rangle_{xz}$</td>
</tr>
<tr>
<td>bxbymy</td>
<td>$\langle B_x B_y \rangle_{xz}$</td>
</tr>
<tr>
<td>bxbyzm</td>
<td>$\langle B_x B_z \rangle_{xz}$</td>
</tr>
<tr>
<td>bybzm</td>
<td>$\langle B_y B_z \rangle_{xz}$</td>
</tr>
</tbody>
</table>

### Module ‘shock_highorder.f90’

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pppmy</td>
<td>$\langle p \rangle_{xz}$</td>
</tr>
<tr>
<td>TTmy</td>
<td>$\langle T \rangle_{xz}$</td>
</tr>
</tbody>
</table>

### Module ‘temperature_idealgas.f90’

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pppmy</td>
<td>$\langle p \rangle_{xz}$</td>
</tr>
<tr>
<td>TTmy</td>
<td>$\langle T \rangle_{xz}$</td>
</tr>
</tbody>
</table>

### Module ‘thermal_energy.f90’

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pppmy</td>
<td>$\langle p \rangle_{xz}$</td>
</tr>
<tr>
<td>TTmy</td>
<td>$\langle T \rangle_{xz}$</td>
</tr>
</tbody>
</table>

### J.8 List of parameters for ‘yzaver.in’

The following table lists possible inputs to the file ‘yzaver.in’. This list is not complete and maybe outdated.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>uxmx</td>
<td>$\langle u_x \rangle_{yz}$</td>
</tr>
<tr>
<td>uymx</td>
<td>$\langle u_y \rangle_{yz}$</td>
</tr>
<tr>
<td>uzmx</td>
<td>$\langle u_z \rangle_{yz}$</td>
</tr>
<tr>
<td>ruxmx</td>
<td>$\langle \rho u_x \rangle_{yz}$</td>
</tr>
<tr>
<td>ruymx</td>
<td>$\langle \rho u_y \rangle_{yz}$</td>
</tr>
<tr>
<td>ruzmx</td>
<td>$\langle \rho u_z \rangle_{yz}$</td>
</tr>
<tr>
<td>rux2mx</td>
<td>$\langle \rho u_x^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>ruy2mx</td>
<td>$\langle \rho u_y^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>ruz2mx</td>
<td>$\langle \rho u_z^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>ruxuymx</td>
<td>$\langle \rho u_x u_y \rangle_{yz}$</td>
</tr>
<tr>
<td>ruxuzmx</td>
<td>$\langle \rho u_x u_z \rangle_{yz}$</td>
</tr>
<tr>
<td>ruyuzmx</td>
<td>$\langle \rho u_y u_z \rangle_{yz}$</td>
</tr>
<tr>
<td>ux2mx</td>
<td>$\langle u_x^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>uy2mx</td>
<td>$\langle u_y^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>uz2mx</td>
<td>$\langle u_z^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>ox2mx</td>
<td>$\langle \omega_x^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>oy2mx</td>
<td>$\langle \omega_y^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>oz2mx</td>
<td>$\langle \omega_z^2 \rangle_{yz}$</td>
</tr>
<tr>
<td>uxuymx</td>
<td>$\langle u_x u_y \rangle_{yz}$</td>
</tr>
<tr>
<td>uxuzmx</td>
<td>$\langle u_x u_z \rangle_{yz}$</td>
</tr>
<tr>
<td>uyuymx</td>
<td>$\langle u_y u_z \rangle_{yz}$</td>
</tr>
<tr>
<td>oumx</td>
<td>$\langle \omega \cdot u \rangle_{yz}$</td>
</tr>
<tr>
<td>ekinmx</td>
<td>$\langle 1/2 \rho u^2 \rangle_{xyz}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module ‘density.f90’</th>
</tr>
</thead>
<tbody>
<tr>
<td>rhomx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module ‘entropy.f90’</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssmx</td>
</tr>
<tr>
<td>ss2mx</td>
</tr>
<tr>
<td>ppmx</td>
</tr>
<tr>
<td>TTmx</td>
</tr>
<tr>
<td>TT2mx</td>
</tr>
<tr>
<td>uXTTmx</td>
</tr>
<tr>
<td>uyTTmx</td>
</tr>
<tr>
<td>uzTTmx</td>
</tr>
<tr>
<td>fconvxmx</td>
</tr>
<tr>
<td>fradmx</td>
</tr>
<tr>
<td>fturbmx</td>
</tr>
<tr>
<td>Kkramersmx</td>
</tr>
<tr>
<td>dcoolx</td>
</tr>
<tr>
<td>fradx_kramers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module ‘magnetic.f90’</th>
</tr>
</thead>
<tbody>
<tr>
<td>b2mx</td>
</tr>
</tbody>
</table>
### Module `bfield.f90`

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bxmx</code></td>
<td>$\langle B_x \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bymx</code></td>
<td>$\langle B_y \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bzmx</code></td>
<td>$\langle B_z \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bx2mx</code></td>
<td>$\langle B_x^2 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>by2mx</code></td>
<td>$\langle B_y^2 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bz2mx</code></td>
<td>$\langle B_z^2 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bxymx</code></td>
<td>$\langle B_x B_y \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bxzmx</code></td>
<td>$\langle B_x B_z \rangle_{yz}$</td>
</tr>
<tr>
<td><code>byzm</code></td>
<td>$\langle B_y B_z \rangle_{yz}$</td>
</tr>
<tr>
<td><code>betamx</code></td>
<td>$\langle \beta \rangle_{yz}$</td>
</tr>
<tr>
<td><code>beta2mx</code></td>
<td>$\langle \beta^2 \rangle_{yz}$</td>
</tr>
</tbody>
</table>

### Module `density_stratified.f90`

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>drhomx</code></td>
<td>$\langle \Delta \rho / \rho_0 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>drho2mx</code></td>
<td>$\langle (\Delta \rho / \rho_0)^2 \rangle_{yz}$</td>
</tr>
</tbody>
</table>

### Module `gravity_simple.f90`

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>epotmx</code></td>
<td>$\langle \phi_{\text{grav}} \rangle_{yz}$</td>
</tr>
<tr>
<td><code>epotuxmx</code></td>
<td>$\langle \phi_{\text{grav} u_x} \rangle_{yz}$ (potential energy flux)</td>
</tr>
</tbody>
</table>

### Module `magnetic_shearboxJ.f90`

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>b2mx</code></td>
<td>$\langle B_x^2 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bxmx</code></td>
<td>$\langle B_x \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bymx</code></td>
<td>$\langle B_y \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bzmx</code></td>
<td>$\langle B_z \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bx2mx</code></td>
<td>$\langle B_x^2 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>by2mx</code></td>
<td>$\langle B_y^2 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bz2mx</code></td>
<td>$\langle B_z^2 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bxymx</code></td>
<td>$\langle B_x B_y \rangle_{yz}$</td>
</tr>
<tr>
<td><code>bxzmx</code></td>
<td>$\langle B_x B_z \rangle_{yz}$</td>
</tr>
<tr>
<td><code>byzm</code></td>
<td>$\langle B_y B_z \rangle_{yz}$</td>
</tr>
<tr>
<td><code>betamx</code></td>
<td>$\langle \beta \rangle_{yz}$</td>
</tr>
<tr>
<td><code>beta2mx</code></td>
<td>$\langle \beta^2 \rangle_{yz}$</td>
</tr>
<tr>
<td><code>etatotalmx</code></td>
<td>$\langle \eta \rangle_{yz}$</td>
</tr>
</tbody>
</table>
### J.9 Parameters for ‘yaver.in’

The following table lists possible inputs to the file ‘yaver.in’. This list is not complete and maybe outdated.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module ‘hydro.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>$u_{xmxz}$</td>
<td>$\langle u_x \rangle_y$</td>
</tr>
<tr>
<td>$u_{ymxz}$</td>
<td>$\langle u_y \rangle_y$</td>
</tr>
<tr>
<td>$u_{zmxz}$</td>
<td>$\langle u_z \rangle_y$</td>
</tr>
<tr>
<td>$u_{2mxz}$</td>
<td>$\langle u_z^2 \rangle_y$</td>
</tr>
<tr>
<td>$u_{ymxz}$</td>
<td>$\langle u_y^2 \rangle_y$</td>
</tr>
<tr>
<td>$u_{z2mxz}$</td>
<td>$\langle u_z^2 \rangle_y$</td>
</tr>
<tr>
<td>$u_{uxymxz}$</td>
<td>$\langle u_x u_y \rangle_y$</td>
</tr>
<tr>
<td>$u_{uxzmxz}$</td>
<td>$\langle u_x u_z \rangle_y$</td>
</tr>
<tr>
<td>$u_{uyuzmxz}$</td>
<td>$\langle u_y u_z \rangle_y$</td>
</tr>
<tr>
<td>$oumxz$</td>
<td>$\langle \omega \cdot u \rangle_y$</td>
</tr>
<tr>
<td><strong>Module ‘density.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>$rhomxz$</td>
<td>$\langle \rho \rangle_y$</td>
</tr>
<tr>
<td><strong>Module ‘entropy.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>$TTmxz$</td>
<td>$\langle T \rangle_y$</td>
</tr>
<tr>
<td>$ssmxz$</td>
<td>$\langle s \rangle_y$</td>
</tr>
<tr>
<td><strong>Module ‘magnetic.f90’</strong></td>
<td></td>
</tr>
<tr>
<td>$b2mxz$</td>
<td>$\langle B^2 \rangle_y$</td>
</tr>
<tr>
<td>$axmxz$</td>
<td>$\langle A_x \rangle_y$</td>
</tr>
<tr>
<td>$aymxz$</td>
<td>$\langle A_y \rangle_y$</td>
</tr>
<tr>
<td>$azmxz$</td>
<td>$\langle A_z \rangle_y$</td>
</tr>
<tr>
<td>$bx1mxz$</td>
<td>$\langle</td>
</tr>
</tbody>
</table>
by1mxz $\langle |B_y| \rangle_y$
bz1mxz $\langle |B_z| \rangle_y$
bxmxz $\langle B_x \rangle_y$
bymxz $\langle B_y \rangle_y$
bzmxz $\langle B_z \rangle_y$
jxmxz $\langle J_x \rangle_y$
jymxz $\langle J_y \rangle_y$
jzmxz $\langle J_z \rangle_y$

bx2mxz $\langle B_z^2 \rangle_y$
by2mxz $\langle B_y^2 \rangle_y$

bz2mxz $\langle B_z^2 \rangle_y$

bxymxz $\langle B_x B_y \rangle_y$

bxbmxz $\langle B_x B_z \rangle_y$

bybmxz $\langle B_y B_z \rangle_y$

uybmxz $\langle U_y B_z \rangle_y$

uybmxz $\langle U_y B_z \rangle_y$

Exmxz $\langle \mathcal{E}_x \rangle_y$

Eymxz $\langle \mathcal{E}_y \rangle_y$

Ezmxz $\langle \mathcal{E}_z \rangle_y$

vAmxz $\langle v^2 \rangle_y$

---

Module 'density_stratified.f90'

drhomxz $\langle \Delta \rho/\rho_0 \rangle_y$
drho2mxz $\langle (\Delta \rho/\rho_0)^2 \rangle_y$

---

Module 'magnetic_shearboxJ.f90'

b2mxz $\langle B_z^2 \rangle_y$

axmxz $\langle A_x \rangle_y$

aymxz $\langle A_y \rangle_y$

azmxz $\langle A_z \rangle_y$

bx1mxz $\langle |B_x| \rangle_y$

by1mxz $\langle |B_y| \rangle_y$

bz1mxz $\langle |B_z| \rangle_y$

bxmxz $\langle B_x \rangle_y$

bymxz $\langle B_y \rangle_y$

bzmxz $\langle B_z \rangle_y$

jxmxz $\langle J_x \rangle_y$

jymxz $\langle J_y \rangle_y$

jzmxz $\langle J_z \rangle_y$

bx2mxz $\langle B_z^2 \rangle_y$

by2mxz $\langle B_y^2 \rangle_y$

bz2mxz $\langle B_z^2 \rangle_y$

bxbymxz $\langle B_x B_y \rangle_y$

bxbzmxz $\langle B_x B_z \rangle_y$

bybzmxz $\langle B_y B_z \rangle_y$

uybmxz $\langle U_y B_z \rangle_y$

uybzmxz $\langle U_y B_z \rangle_y$

Exmxz $\langle \mathcal{E}_x \rangle_y$
### J.10 Parameters for ‘zaver.in’

The following table lists possible inputs to the file ‘zaver.in’. This list is not complete and may be outdated.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Eymxz</strong></td>
<td>$\langle \mathcal{E}_y \rangle_y$</td>
</tr>
<tr>
<td><strong>Ezmxz</strong></td>
<td>$\langle \mathcal{E}_z \rangle_y$</td>
</tr>
<tr>
<td><strong>vAmxz</strong></td>
<td>$\langle v_A^2 \rangle_y$</td>
</tr>
<tr>
<td></td>
<td><strong>Module ‘meanfield.f90’</strong></td>
</tr>
<tr>
<td><strong>peffmxz</strong></td>
<td>$\langle P_{\text{eff}} \rangle_y$</td>
</tr>
<tr>
<td><strong>alpmxz</strong></td>
<td>$\langle \alpha \rangle_y$</td>
</tr>
<tr>
<td></td>
<td><strong>Module ‘temperature_idealgas.f90’</strong></td>
</tr>
<tr>
<td><strong>TTmxz</strong></td>
<td>$\langle T \rangle_y$</td>
</tr>
<tr>
<td><strong>Emymxz</strong></td>
<td>$\langle E_{\text{my}} \rangle_y$</td>
</tr>
</tbody>
</table>

**Emission in y-direction**

<table>
<thead>
<tr>
<th><strong>Module ‘thermal_energy.f90’</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TTmxz</strong></td>
</tr>
</tbody>
</table>

### J.10 List of parameters for ‘zaver.in’

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>uxmxy</strong></td>
<td>$\langle u_x \rangle_z$</td>
</tr>
<tr>
<td><strong>uymxy</strong></td>
<td>$\langle u_y \rangle_z$</td>
</tr>
<tr>
<td><strong>uzmxy</strong></td>
<td>$\langle u_z \rangle_z$</td>
</tr>
<tr>
<td><strong>uxuymxy</strong></td>
<td>$\langle u_x u_y \rangle_z$</td>
</tr>
<tr>
<td><strong>uxuzmxy</strong></td>
<td>$\langle u_x u_z \rangle_z$</td>
</tr>
<tr>
<td><strong>uyuzmxy</strong></td>
<td>$\langle u_y u_z \rangle_z$</td>
</tr>
<tr>
<td><strong>oxmxy</strong></td>
<td>$\langle \omega_x \rangle_z$</td>
</tr>
<tr>
<td><strong>oymxy</strong></td>
<td>$\langle \omega_y \rangle_z$</td>
</tr>
<tr>
<td><strong>ozmxy</strong></td>
<td>$\langle \omega_z \rangle_z$</td>
</tr>
<tr>
<td><strong>oumxy</strong></td>
<td>$\langle \omega \cdot u \rangle_z$</td>
</tr>
<tr>
<td><strong>pvzmx</strong></td>
<td>$(\omega_z + 2\Omega)/\rho$</td>
</tr>
<tr>
<td><strong>uguuxmxy</strong></td>
<td>$\langle (u \cdot \nabla u)_{x,z} \rangle_z$</td>
</tr>
<tr>
<td><strong>uguymxy</strong></td>
<td>$\langle (u \cdot \nabla u)_{y,z} \rangle_z$</td>
</tr>
<tr>
<td><strong>uguuzmxy</strong></td>
<td>$\langle (u \cdot \nabla u)_{z,z} \rangle_z$</td>
</tr>
<tr>
<td><strong>ruxmxy</strong></td>
<td>$\langle \rho u_x \rangle_z$</td>
</tr>
<tr>
<td><strong>ruymxy</strong></td>
<td>$\langle \rho u_y \rangle_z$</td>
</tr>
<tr>
<td><strong>ruzmxy</strong></td>
<td>$\langle \rho u_z \rangle_z$</td>
</tr>
<tr>
<td><strong>ux2mx</strong></td>
<td>$\langle u_x^2 \rangle_z$</td>
</tr>
<tr>
<td><strong>uy2mx</strong></td>
<td>$\langle u_y^2 \rangle_z$</td>
</tr>
<tr>
<td><strong>uz2mx</strong></td>
<td>$\langle u_z^2 \rangle_z$</td>
</tr>
<tr>
<td><strong>rux2mx</strong></td>
<td>$\langle \rho u_x^2 \rangle_z$</td>
</tr>
<tr>
<td><strong>ruy2mx</strong></td>
<td>$\langle \rho u_y^2 \rangle_z$</td>
</tr>
<tr>
<td><strong>ruz2mx</strong></td>
<td>$\langle \rho u_z^2 \rangle_z$</td>
</tr>
<tr>
<td><strong>ruxuymxy</strong></td>
<td>$\langle \rho u_x u_y \rangle_z$</td>
</tr>
<tr>
<td><strong>ruxuzmxy</strong></td>
<td>$\langle \rho u_x u_z \rangle_z$</td>
</tr>
</tbody>
</table>
ruyuzmxy $\langle \rho u_y u_z \rangle_z$
fkinxmxy $\langle \frac{1}{2} \rho u^2 u_x \rangle_z$
fkinymxy $\langle \frac{1}{2} \rho u^2 u_y \rangle_z$

Module ‘density.f90’

rhomxy $\langle \rho \rangle_z$

Module ‘entropy.f90’

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTmxy</td>
<td>$\langle T \rangle_z$</td>
</tr>
<tr>
<td>ssmxxy</td>
<td>$\langle s \rangle_z$</td>
</tr>
<tr>
<td>uxtTmxy</td>
<td>$\langle u_x T \rangle_z$</td>
</tr>
<tr>
<td>uytTmxy</td>
<td>$\langle u_y T \rangle_z$</td>
</tr>
<tr>
<td>uzTmxy</td>
<td>$\langle u_z T \rangle_z$</td>
</tr>
<tr>
<td>gTxmxy</td>
<td>$\langle \nabla_x T \rangle_z$</td>
</tr>
<tr>
<td>gTymxy</td>
<td>$\langle \nabla_y T \rangle_z$</td>
</tr>
<tr>
<td>gTzmxy</td>
<td>$\langle \nabla_z T \rangle_z$</td>
</tr>
<tr>
<td>gsxmxy</td>
<td>$\langle \nabla_x s \rangle_z$</td>
</tr>
<tr>
<td>gsymxy</td>
<td>$\langle \nabla_y s \rangle_z$</td>
</tr>
<tr>
<td>gszmxy</td>
<td>$\langle \nabla_z s \rangle_z$</td>
</tr>
<tr>
<td>gTxgsxmxy</td>
<td>$\langle \nabla T \times \nabla s \rangle_x \rangle_z$</td>
</tr>
<tr>
<td>gTxgsymxy</td>
<td>$\langle \nabla T \times \nabla s \rangle_y \rangle_z$</td>
</tr>
<tr>
<td>gTxgszmxy</td>
<td>$\langle \nabla T \times \nabla s \rangle_z \rangle_z$</td>
</tr>
<tr>
<td>gTxgsx2mxy</td>
<td>$\langle \nabla T \times \nabla s \rangle_x^2 \rangle_z$</td>
</tr>
<tr>
<td>gTxgsy2mxy</td>
<td>$\langle \nabla T \times \nabla s \rangle_y^2 \rangle_z$</td>
</tr>
<tr>
<td>gTxgsz2mxy</td>
<td>$\langle \nabla T \times \nabla s \rangle_z^2 \rangle_z$</td>
</tr>
<tr>
<td>fconvxy</td>
<td>$\langle c_p \rho u_x T \rangle_z$</td>
</tr>
<tr>
<td>fconvyxy</td>
<td>$\langle c_p \rho u_y T \rangle_z$</td>
</tr>
<tr>
<td>fconvzxy</td>
<td>$\langle c_p \rho u_z T \rangle_z$</td>
</tr>
<tr>
<td>fradxy_Kprof</td>
<td>$F^x_{\text{rad}}$ (x-component of radiative flux, z-averaged, from Kprof)</td>
</tr>
<tr>
<td>fradyx_Kprof</td>
<td>$F^y_{\text{rad}}$ (y-component of radiative flux, z-averaged, from Kprof)</td>
</tr>
<tr>
<td>fradxy_kramers</td>
<td>$F^z_{\text{rad}}$ (z-averaged, from Kramers’ opacity)</td>
</tr>
<tr>
<td>fturbxy</td>
<td>$\langle g T X_i \nabla_x s \rangle_z$</td>
</tr>
<tr>
<td>fturbymxy</td>
<td>$\langle g T X_i \nabla_y s \rangle_z$</td>
</tr>
<tr>
<td>fturbxy</td>
<td>$\langle g T X_i \nabla_x s \rangle_z$</td>
</tr>
<tr>
<td>fturbthxy</td>
<td>$\langle g T X_i \nabla_s \rangle_z$</td>
</tr>
<tr>
<td>dcoolxy</td>
<td>surface cooling flux</td>
</tr>
</tbody>
</table>

Module ‘magnetic.f90’

bxmxy $\langle B_x \rangle_z$
bymxy $\langle B_y \rangle_z$
bzmxy $\langle B_z \rangle_z$
jxmxxy $\langle J_x \rangle_z$
jymxy $\langle J_y \rangle_z$
jzmxxy $\langle J_z \rangle_z$
axmxy $\langle A_x \rangle_z$
asymxy $\langle A_y \rangle_z$
avmxy $\langle A_z \rangle_z$
azmxy $\langle A_z \rangle_z$

bx2mxy $\langle B_x^2 \rangle_z$
by2mxy $\langle B_y^2 \rangle_z$
bz2mxy $\langle B_z^2 \rangle_z$
bxbymxy $\langle B_x B_y \rangle_z$
bxbzmx $\langle B_x B_z \rangle_z$
bybzmxy $\langle B_y B_z \rangle_z$
poynxmx $\langle E \times B \rangle_x$
poynymx $\langle E \times B \rangle_y$
poynzmx $\langle E \times B \rangle_z$
jbxm $\langle J \cdot B \rangle_z$
abmx $\langle A \cdot B \rangle_z$
examxy1 $\langle E \times A \rangle_x$
examxy2 $\langle E \times A \rangle_y$
examxy3 $\langle E \times A \rangle_z$
StokesIm $\langle \epsilon_{BL} \rangle_z$
StokesQm $- \langle \epsilon_{BL} \cos 2\chi \rangle_z$
StokesU $- \langle \epsilon_{BL} \sin 2\chi \rangle_z$
StokesQ $+ \langle F \epsilon_{BL} \sin 2\chi \rangle_z$
StokesU1 $- \langle F \epsilon_{BL} \cos 2\chi \rangle_z$
beta1 $\langle B^2 / (2\mu_0 \rho) \rangle_z$

Module ‘density_stratified.f90’

drhom $\langle \Delta \rho / \rho_0 \rangle_z$
drho2m $\langle (\Delta \rho / \rho_0)^2 \rangle_z$

Module ‘gravity_simple.f90’

epotm $\langle \phi_{grav} \rangle_z$
epotuxm $\langle \phi_{grav} u_x \rangle_z$ (potential energy flux)

Module ‘magnetic_shearboxJ.f90’

bymx $\langle B_y \rangle_z$
bzmx $\langle B_z \rangle_z$
jymx $\langle J_y \rangle_z$
jzmx $\langle J_z \rangle_z$
axmx $\langle A_x \rangle_z$
aymx $\langle A_y \rangle_z$
azmx $\langle A_z \rangle_z$
bx2m $\langle B_x^2 \rangle_z$
by2m $\langle B_y^2 \rangle_z$
.bz2m $\langle B_z^2 \rangle_z$
bxbym $\langle B_x B_y \rangle_z$
bxbzm $\langle B_x B_z \rangle_z$
bybzm $\langle B_y B_z \rangle_z$
poynx $\langle E \times B \rangle_x$
poynym $\langle E \times B \rangle_y$
poynzmx $\langle E \times B \rangle_z$
jbmx $\langle J \cdot B \rangle_z$
abmx $\langle A \cdot B \rangle_z$
examxy1 $\langle E \times A \rangle_x$
\[ \mathbf{E} \times \mathbf{A} \mid_y \]
\[ \mathbf{E} \times \mathbf{A} \mid_z \]
\[ \langle \epsilon_B \perp \rangle \mid_z \]
\[ - \langle \epsilon_B \perp \cos 2\chi \rangle \mid_z \]
\[ - \langle \epsilon_B \perp \sin 2\chi \rangle \mid_z \]
\[ + \langle F \epsilon_B \perp \sin 2\chi \rangle \mid_z \]
\[ - \langle F \epsilon_B \perp \cos 2\chi \rangle \mid_z \]
\[ \langle B^2/(2\mu_0) \rangle \mid_z \]

\[ \langle T \rangle \]
\[ \langle E_{mz} \rangle \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>zero value in ghost zones, free value on boundary</td>
</tr>
<tr>
<td>p</td>
<td>periodic</td>
</tr>
<tr>
<td>s</td>
<td>symmetry, ( f_{N+i} = f_{N-i} ); implies ( f'(x_N) = f''(x_0) = 0 )</td>
</tr>
<tr>
<td>sf</td>
<td>symmetry with respect to interface</td>
</tr>
<tr>
<td>ss</td>
<td>symmetry, plus function value given</td>
</tr>
<tr>
<td>s0d</td>
<td>symmetry, function value such that ( df/dx=0 )</td>
</tr>
<tr>
<td>a</td>
<td>antisymmetry, ( f_{N+i} = -f_{N-i} ); implies ( f(x_N) = f''(x_0) = 0 )</td>
</tr>
<tr>
<td>af</td>
<td>antisymmetry with respect to interface</td>
</tr>
<tr>
<td>a2</td>
<td>antisymmetry relative to boundary value, ( f_{N+i} = 2f_N - f_{N-i} ); implies ( f''(x_0) = 0 )</td>
</tr>
<tr>
<td>a2v</td>
<td>set boundary value and antisymmetry relative to it ( f_{N+i} = 2f_N - f_{N-i} ); implies ( f''(x_0) = 0 )</td>
</tr>
<tr>
<td>a2r</td>
<td>sets ( d^2f/dr^2 + 2df/dr - 2f/r^2 = 0 ) This is the replacement of zero second derivative in spherical coordinates, in radial direction.</td>
</tr>
<tr>
<td>cpc</td>
<td>cylindrical perfect conductor implies ( f'' + f'/R = 0 )</td>
</tr>
<tr>
<td>cpp</td>
<td>cylindrical perfect conductor implies ( f'' + f'/R = 0 )</td>
</tr>
<tr>
<td>cpz</td>
<td>cylindrical perfect conductor implies ( f'' + f'/R = 0 )</td>
</tr>
<tr>
<td>spr</td>
<td>spherical perfect conductor implies ( f'' + 2f'/R = 0 ) and ( f(x_N) = 0 )</td>
</tr>
<tr>
<td>v</td>
<td>vanishing third derivative</td>
</tr>
</tbody>
</table>
**J.11 Boundary conditions**

- **cop**: copy value of last physical point to all ghost cells
- **1s**: onesided
- **dl1s**: onesided for 1st/2nd derivative in two first inner points, Dirichlet in boundary point
- **n1s**: onesided for 1st/2nd derivative in two first inner points, Neumann in boundary point
- **1so**: onesided
- **cT**: constant temperature (implemented as condition for entropy s or temperature T)
- **cl**: constant temperature (or maybe rather constant conductive flux??)
- **Fgs**: \( F_{\text{conv}} = -\chi_t \rho \cdot T \cdot \nabla(s) \)
- **Fct**: \( F_{\text{bot}} = -K\cdot \nabla(T) - \chi_t \rho \cdot T \cdot \nabla(s) \)
- **Fcm**: \( F_{\text{bot}} = -K\cdot \nabla(T) - \chi_t \rho \cdot T \cdot \nabla(s) \)
- **sT**: symmetric temperature, \( T_{N-i} = T_{N+i} \); implies \( T'(x_N) = T''(x_0) = 0 \)
- **asT**: select entropy for uniform ghost temperature matching fluctuating boundary value, \( T_{N-i} = T_N = 0 \); implies \( T'(x_N) = T''(x_0) = 0 \)
- **f**: “freeze” value, i.e. maintain initial
- **fg**: “freeze” value, i.e. maintain initial
- **1**: \( f = 1 \) (for debugging)
- **set**: set boundary value to \( f_{bcx} \)
- **der**: set derivative on boundary to \( f_{bcx} \)
- **slo**: set slope at the boundary = \( f_{bcx} \)
- **slp**: set slope at the boundary and in ghost cells = \( f_{bcx} \)
- **shx**: set shearing boundary proportional to x with slope=\( f_{bcx} \) and abscissa=\( f_{bcx2} \)
- **shy**: set shearing boundary proportional to y with slope=\( f_{bcx} \) and abscissa=\( f_{bcx2} \)
- **shz**: set shearing boundary proportional to z with slope=\( f_{bcx} \) and abscissa=\( f_{bcx2} \)
- **dr0**: set boundary value [really??]
- **ovr**: overshoot boundary condition for \( (d/dx - 1/dist) f = 0 \).
- **out**: allow outflow, but no inflow forces ghost cells and boundary to not point inwards
- **e1o**: allow outflow, but no inflow uses the e1 extrapolation scheme
- **ant**: stops and prompts for adding documentation
- **e1**: extrapolation [describe]
- **e2**: extrapolation [describe]
- **e3**: extrapolation in log [maintain a power law]
- **hat**: top hat jet profile in spherical coordinate.
- **jet**: top hat jet profile in Cartesian coordinate.
- **spd**: sets \( d(rA_\alpha)/dr = f_{bcx}(j) \)
- **sfr**: stress-free boundary condition for spherical coordinate system.
- **sr1**: Stress-free bc for spherical coordinate system. Implementation with one-sided derivative.
- **nfr**: Normal-field bc for spherical coordinate system. Some people call this the “(angry) hedgehog bc”.
- **nr1**: Normal-field bc for spherical coordinate system. Some people call this the “(angry) hedgehog bc”. Implementation with one-sided derivative.
- **sa2**: \( (d/dr)(rB_\phi) = 0 \) imposes boundary condition on 2nd derivative of \( rA_\phi \). Same applies to \( \theta \) component.
**pfc**  
perfect-conductor in spherical coordinate: \(d/dr(A_r) + 2/r = 0\).

**fix**  
set boundary value [really??]

**fil**  
set boundary value from a file

**cfr**  
radial centrifugal balance

**g**  
set to given value(s) or function

**nil**  
do nothing; assume that everything is set

**ioc**  
inlet/outlet on western/eastern hemisphere in cylindrical coordinates

do nothing; assume that everything is set

**s**  
implies \(f'(y_N) = f'''(y_0) = 0\)

---

**Module 'boundcond_alt.f90'**

<table>
<thead>
<tr>
<th>0</th>
<th>zero value in ghost zones, free value on boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>p</strong></td>
<td>periodic</td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>symmetry, (f_{N+i} = f_{N-i}); implies (f'(x_N) = f'''(x_0) = 0)</td>
</tr>
<tr>
<td><strong>ss</strong></td>
<td>symmetry, plus function value given</td>
</tr>
<tr>
<td><strong>s0d</strong></td>
<td>symmetry, function value such that (df/dx=0)</td>
</tr>
<tr>
<td><strong>a</strong></td>
<td>antisymmetry, (f_{N+i} = -f_{N-i}); implies (f'(x_N) = f'''(x_0) = 0)</td>
</tr>
<tr>
<td><strong>a2</strong></td>
<td>antisymmetry relative to boundary value, (f_{N+i} = 2f_N - f_{N-i}); implies (f'''(x_0) = 0)</td>
</tr>
<tr>
<td><strong>a2r</strong></td>
<td>sets (d^2f/dr^2 + 2df/dr - 2f/r^2 = 0) This is the replacement of zero second derivative in spherical coordinates, in radial direction.</td>
</tr>
<tr>
<td><strong>cpc</strong></td>
<td>cylindrical perfect conductor implies (f'' + f'/R = 0)</td>
</tr>
<tr>
<td><strong>cpp</strong></td>
<td>cylindrical perfect conductor implies (f'' + f'/R = 0)</td>
</tr>
<tr>
<td><strong>cpz</strong></td>
<td>cylindrical perfect conductor implies (f'' + f'/R = 0)</td>
</tr>
<tr>
<td><strong>spr</strong></td>
<td>spherical perfect conductor implies (f'' + 2f'/R = 0) and (f(x_N) = 0)</td>
</tr>
<tr>
<td><strong>v</strong></td>
<td>vanishing third derivative</td>
</tr>
<tr>
<td><strong>cop</strong></td>
<td>copy value of last physical point to all ghost cells</td>
</tr>
<tr>
<td><strong>1s</strong></td>
<td>onesided</td>
</tr>
<tr>
<td><strong>1so</strong></td>
<td>onesided</td>
</tr>
<tr>
<td><strong>c1</strong></td>
<td>constant temperature (or maybe rather constant conductive flux??)</td>
</tr>
<tr>
<td><strong>ct</strong></td>
<td>constant temperature (implemented as condition for entropy (s) or temperature (T))</td>
</tr>
<tr>
<td><strong>fct</strong></td>
<td>Fconv = - (\chi_t \cdot \rho \cdot T \cdot \text{grad}(s))</td>
</tr>
<tr>
<td><strong>Fct</strong></td>
<td>Fbot = - (K \cdot \text{grad}(T) - \chi_t \cdot \rho \cdot T \cdot \text{grad}(s))</td>
</tr>
<tr>
<td><strong>Fcm</strong></td>
<td>(Fbot = -K \cdot \text{grad}(T) - \chi_t \cdot \rho \cdot T \cdot \text{grad}(\tilde{s}))</td>
</tr>
<tr>
<td><strong>st</strong></td>
<td>symmetric temperature, (T_{N-i} = T_{N+i}); implies (T'(x_N) = T'''(x_0) = 0)</td>
</tr>
<tr>
<td><strong>asT</strong></td>
<td>select entropy for uniform ghost temperature matching fluctuating boundary value, (T_{N-i} = T_{N+i}); implies (T'(x_N) = T'(x_0) = 0)</td>
</tr>
<tr>
<td><strong>f</strong></td>
<td>&quot;freeze&quot; value, i.e. maintain initial</td>
</tr>
<tr>
<td><strong>fg</strong></td>
<td>&quot;freeze&quot; value, i.e. maintain initial</td>
</tr>
<tr>
<td><strong>1</strong></td>
<td>(f = 1) (for debugging)</td>
</tr>
<tr>
<td><strong>fbcx12</strong></td>
<td>set boundary value to (fbcx12)</td>
</tr>
<tr>
<td><strong>der</strong></td>
<td>set derivative on boundary to (fbcx12)</td>
</tr>
<tr>
<td><strong>slo</strong></td>
<td>set slope at the boundary = (fbcx12)</td>
</tr>
<tr>
<td><strong>dr0</strong></td>
<td>set boundary value [really??]</td>
</tr>
<tr>
<td><strong>ovr</strong></td>
<td>overshoot boundary condition ie ((d/dx - 1/dist)f = 0).</td>
</tr>
<tr>
<td><strong>out</strong></td>
<td>allow outflow, but no inflow forces ghost cells and boundary to not point inwards</td>
</tr>
<tr>
<td><strong>e1o</strong></td>
<td>allow outflow, but no inflow uses the e1 extrapolation scheme</td>
</tr>
<tr>
<td><strong>ant</strong></td>
<td>stops and prompts for adding documentation</td>
</tr>
</tbody>
</table>
J.11 Boundary conditions

e1 extrapolation [describe]
e2 extrapolation [describe]
e3 extrapolation in log [maintain a power law]
hat top hat jet profile in spherical coordinate.
jet top hat jet profile in cartesian coordinate.
spd sets \( \frac{d(rA_\phi)}{dr} = f_{bcx12}(j) \)
sfr stress-free boundary condition for spherical coordinate system.
nfr Normal-field bc for spherical coordinate system. Some people call this the “(angry) hedgehog bc”.
sa2 \( (d/dr)(rB_\phi) = 0 \) imposes boundary condition on 2nd derivative of \( rA_\phi \).
   Same applies to \( \theta \) component.
pfc perfect-conductor in spherical coordinate: \( \frac{d}{dr}(A_r) + \frac{2}{r} = 0 \).
fix set boundary value [really??]
fil set boundary value from a file
nil do nothing; assume that everything is set
ioc inlet/outlet on western/eastern hemisphere in cylindrical coordinates
do nothing; assume that everything is set
   \( s \) implies \( f'(y_N) = f'''(y_0) = 0 \)

J.11.2 Boundary condition bcy

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>sds</td>
<td>symmetric-derivative-set</td>
</tr>
<tr>
<td>0</td>
<td>zero value in ghost zones, free value on boundary</td>
</tr>
<tr>
<td>p</td>
<td>periodic</td>
</tr>
<tr>
<td>pp</td>
<td>periodic across the pole</td>
</tr>
<tr>
<td>yy</td>
<td>Yin-Yang grid</td>
</tr>
<tr>
<td>ap</td>
<td>anti-periodic across the pole</td>
</tr>
<tr>
<td>s</td>
<td>symmetry symmetry, ( f_{N+i} = f_{N-i} );</td>
</tr>
<tr>
<td>ss</td>
<td>symmetry, plus function value given</td>
</tr>
<tr>
<td>sds</td>
<td>symmetric-derivative-set</td>
</tr>
<tr>
<td>cds</td>
<td>complex symmetric-derivative-set</td>
</tr>
<tr>
<td>s0d</td>
<td>symmetry, function value such that ( df/dy=0 )</td>
</tr>
<tr>
<td>a</td>
<td>antisymmetry</td>
</tr>
<tr>
<td>a2</td>
<td>antisymmetry relative to boundary value</td>
</tr>
<tr>
<td>v</td>
<td>vanishing third derivative</td>
</tr>
<tr>
<td>v3</td>
<td>vanishing third derivative</td>
</tr>
<tr>
<td>out</td>
<td>allow outflow, but no inflow forces ghost cells and boundary to not point inwards</td>
</tr>
<tr>
<td>1s</td>
<td>onesided</td>
</tr>
<tr>
<td>d1s</td>
<td>onesided for 1st and 2nd derivative in two first inner points, Dirichlet in boundary point</td>
</tr>
<tr>
<td>n1s</td>
<td>onesided for 1st and 2nd derivative in two first inner points, Neumann in boundary point</td>
</tr>
<tr>
<td>cT</td>
<td>constant temp.</td>
</tr>
<tr>
<td>sT</td>
<td>symmetric temp.</td>
</tr>
</tbody>
</table>
asT  select entropy for uniform ghost temperature matching fluctuating boundary value, \( T_{N-i} = T_N = \); implies \( T'(x_N) = T'(x_0) = 0 \)
f  freeze value
s+f freeze value
fg “freeze” value, i.e. maintain initial
fBs frozen-in B-field (s)
fB frozen-in B-field (a2)
1 f=1 (for debugging)
set set boundary value
sse symmetry, set boundary value
sep set boundary value
e1 extrapolation
e2 extrapolation
e3 extrapolation in log [maintain a power law]
der set derivative on the boundary
cop outflow: copy value of last physical point to all ghost cells
c+k no-inflow: copy value of last physical point to all ghost cells, but suppressing any inflow
sfr stress-free boundary condition for spherical coordinate system.
nfr Normal-field bc for spherical coordinate system. Some people call this the “(angry) hedgehog bc”.
spt spherical perfect conducting boundary condition along \( \theta \) boundary \( f'' + \cot\theta f' = 0 \) and \( f(x_N) = 0 \)
pfc perfect conducting boundary condition along \( \theta \) boundary
nil', do nothing; assume that everything is set
sep set boundary value
crk no-inflow: copy value of last physical point to all ghost cells, but suppressing any inflow

Module ‘boundcond_alt.f90’

sds symmetric-derivative-set
0 zero value in ghost zones, free value on boundary
p periodic
pp periodic across the pole
ap anti-periodic across the pole
s symmetry symmetry, \( f_{N+i} = f_{N-i} \);
ss symmetry, plus function value given
sds symmetric-derivative-set
cds complex symmetric-derivative-set
s0d symmetry, function value such that \( df/dy=0 \)
a antisymmetry
a2 antisymmetry relative to boundary value
v vanishing third derivative
v3 vanishing third derivative
out allow outflow, but no inflow forces ghost cells and boundary to not point inwards
1s onesided
cT constant temp.
sT symmetric temp.
asT  select entropy for uniform ghost temperature matching fluctuating
    boundary value, \( T_{N-i} = T_N = \); implies \( T'(x_N) = T'(x_0) = 0 \)

\( f \)  freeze value

\( s+f \)  freeze value

fg  “freeze” value, i.e. maintain initial

\( l \)  \( f=1 \) (for debugging)

set  set boundary value

sse  symmetry, set boundary value

sep  set boundary value

e1  extrapolation

e2  extrapolation

e3  extrapolation in log [maintain a power law]

der  set derivative on the boundary

cop  outflow: copy value of last physical point to all ghost cells

c+k  no-inflow: copy value of last physical point to all ghost cells, but sup-
    pressing any inflow

sfr  stress-free boundary condition for spherical coordinate system.

nfr  Normal-field bc for spherical coordinate system. Some people call this
    the “(angry) hedgehog bc”.

spt  spherical perfect conducting boundary condition along \( \theta \) boundary \( f'' + \cot \theta f' = 0 \) and \( f(x_N) = 0 \)

pfc  perfect conducting boundary condition along \( \theta \) boundary

nil’,  do nothing; assume that everything is set

sep  set boundary value

\[ \]

J.11.3 Boundary condition bcz

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>zero value in ghost zones, free value on boundary</td>
</tr>
<tr>
<td>p</td>
<td>periodic</td>
</tr>
<tr>
<td>yy</td>
<td>Yin-Yang grid</td>
</tr>
<tr>
<td>s</td>
<td>symmetry</td>
</tr>
<tr>
<td>sf</td>
<td>symmetry with respect to interface</td>
</tr>
<tr>
<td>s0d</td>
<td>symmetry, function value such that df/dz=0</td>
</tr>
<tr>
<td>0ds</td>
<td>symmetry, function value such that df/dz=0</td>
</tr>
<tr>
<td>a</td>
<td>antisymmetry</td>
</tr>
<tr>
<td>a2</td>
<td>antisymmetry relative to boundary value</td>
</tr>
<tr>
<td>a2v</td>
<td>set boundary value and antisymmetry relative to it</td>
</tr>
<tr>
<td>af</td>
<td>antisymmetry with respect to interface</td>
</tr>
<tr>
<td>a0d</td>
<td>antisymmetry with zero derivative</td>
</tr>
<tr>
<td>v</td>
<td>vanishing third derivative</td>
</tr>
<tr>
<td>v3</td>
<td>vanishing third derivative</td>
</tr>
<tr>
<td>1s</td>
<td>one-sided</td>
</tr>
<tr>
<td>d1s</td>
<td>onesided for 1st and 2nd derivative in two first inner points, Dirichlet</td>
</tr>
<tr>
<td>n1s</td>
<td>onesided for 1st and 2nd derivative in two first inner points, Neumann</td>
</tr>
</tbody>
</table>
a1s special for perfect conductor with const alpha and etaT when A considered as B; one-sided for 1st and 2nd derivative in two first inner points

fg “freeze” value, i.e. maintain initial

c1 complex

c1s complex

Fgs Fconv = - \chi_t \rho T \nabla \nabla(s)

Fct Fbot = - K \nabla \nabla(T) - \chi_t \rho T \nabla \nabla(s)

c3 constant flux at the bottom with a variable hcond

pfe potential field extrapolation

p1D potential field extrapolation in 1D

pot potential magnetic field

pwd a variant of ‘pot’ for nprocx=1

hds hydrostatic equilibrium with a high-frequency filter

cT constant temp.

cT1 constant temp.

cT2 constant temp. (keep lnrho)

cT3 constant temp. (keep lnrho)

hs hydrostatic equilibrium

hse hydrostatic extrapolation rho or lnrho is extrapolated linearly and the temperature is calculated in hydrostatic equilibrium.

cp constant pressure

sT symmetric temp.

cTz for interstellar runs copy T

cdz for interstellar runs limit rho

ism for interstellar runs limit rho

asT select entropy for uniform ghost temperature matching fluctuating boundary value, \( T_{N-1} = T_N = 0 \); implies \( T'(x_N) = T'(x_0) = 0 \)

c2 complex

db complex

ce complex

e1 extrapolation

e2 extrapolation

ex simple linear extrapolation in first order

exf simple linear extrapolation in first order

exd simple linear extrapolation in first order

exm simple linear extrapolation in first order

b1 extrapolation with zero value (improved ‘a’)

b2 extrapolation with zero value (improved ‘a’)

b3 extrapolation with zero value (improved ‘a’)

f’,fa freeze value + antisymmetry

fs freeze value + symmetry

fBs frozen-in B-field (s)

fB frozen-in B-field (a2)

g set to given value(s) or function

1 f=1 (for debugging)

StS solar surface boundary conditions

set set boundary value

der set derivative on the boundary

div set the divergence of \( u \) to a given value use bc = ‘div’ for iuz
| ovr  | set boundary value                  |
| inf  | allow inflow, but no outflow        |
| ouf  | allow outflow, but no inflow        |
| in   | allow inflow, but no outflow forces ghost cells and boundary to not point outwards |
| out  | allow outflow, but no inflow forces ghost cells and boundary to not point inwards |
| in0  | allow inflow, but no outflow forces ghost cells and boundary to not point outwards relaxes to vanishing 1st derivative at boundary |
| ou0  | allow outflow, but no inflow forces ghost cells and boundary to not point inwards relaxes to vanishing 1st derivative at boundary |
| ind  | allow inflow, but no outflow forces ghost cells and boundary to not point inwards creates inwards pointing or zero 1st derivative at boundary |
| oud  | allow outflow, but no inflow forces ghost cells and boundary to not point inwards creates outwards pointing or zero 1st derivative at boundary |
| ubs  | copy boundary outflow              |
| win  | forces mass flux given as $\sum \rho_i (u_i + u_0) = fbcz1/2(\rho)$ |
| cop  | copy value of last physical point to all ghost cells |
| nil  | do nothing; assume that everything is set |

Module 'boundcond_alt.f90'

| cfb  | radial centrifugal balance         |
| fBs  | frozen-in B-field (s)              |
| fB   | frozen-in B-field (a2)             |
| 0    | zero value in ghost zones, free value on boundary |
| p    | periodic                            |
| s    | symmetry                            |
| sf   | symmetry with respect to interface  |
| s0d  | symmetry, function value such that df/dz=0 |
| 0ds  | symmetry, function value such that df/dz=0 |
| a    | antisymmetry                        |
| a2   | antisymmetry relative to boundary value |
| af   | antisymmetry with respect to interface |
| a0d  | antisymmetry with zero derivative   |
| v    | vanishing third derivative          |
| v3   | vanishing third derivative          |
| 1s   | one-sided                           |
| fg   | “freeze” value, i.e. maintain initial |
| c1   | complex                             |
| Fgs  | $F_{\text{conv}} = - \chi_t\rho^T\text{grad}(s)$ |
| Fct  | $F_{\text{bot}} = - K\text{grad}(T) - \chi_t\rho^T\text{grad}(s)$ |
| c3   | constant flux at the bottom with a variable hcond |
| pfe  | potential field extrapolation       |
| p1D  | potential field extrapolation in 1D  |
| pot  | potential magnetic field            |
| pwd  | a variant of 'pot' for nprocx=1     |
| hds  | hydrostatic equilibrium with a high-frequency filter |
| cT   | constant temp.                      |
| cT2  | constant temp. (keep lnrho)         |
| cT3  | constant temp. (keep lnrho)         |
hs  hydrostatic equilibrium
hse hydrostatic extrapolation rho or ln rho is extrapolated linearly and the
temperature is calculated in hydrostatic equilibrium.
cp constant pressure
sT symmetric temp.
cTz for interstellar runs copy T
cDz for interstellar runs limit rho
asT select entropy for uniform ghost temperature matching fluctuating
boundary value, \( T_{N-1} = T_N =; \) implies \( T'(x_N) = T'(x_0) = 0 \)
c2 complex
db complex
ce complex
e1 extrapolation
e2 extrapolation
ex simple linear extrapolation in first order
exf simple linear extrapolation in first order
exd simple linear extrapolation in first order
exm simple linear extrapolation in first order
b1 extrapolation with zero value (improved 'a')
b2 extrapolation with zero value (improved 'a')
b3 extrapolation with zero value (improved 'a')
f',fa freeze value + antisymmetry
fs freeze value + symmetry
fBs frozen-in B-field (s)
fB frozen-in B-field (a2)
g set to given value(s) or function
1 f=1 (for debugging)
SsS solar surface boundary conditions
set set boundary value
der set derivative on the boundary
div set the divergence of \( u \) to a given value use bc = 'div' for iuz
ovr set boundary value
inf allow inflow, but no outflow
ouf allow outflow, but no inflow
in allow inflow, but no outflow forces ghost cells and boundary to not point
    outwards
out allow outflow, but no inflow forces ghost cells and boundary to not point
    inwards
in0 allow inflow, but no outflow forces ghost cells and boundary to not point
    outwards relaxes to vanishing 1st derivative at boundary
ou0 allow outflow, but no inflow forces ghost cells and boundary to not point
    inwards relaxes to vanishing 1st derivative at boundary
ind allow inflow, but no outflow forces ghost cells and boundary to not point
    outwards creates inwards pointing or zero 1st derivative at boundary
oud allow outflow, but no inflow forces ghost cells and boundary to not point
    inwards creates outwards pointing or zero 1st derivative at boundary
ubs copy boundary outflow,
win forces massflux given as \( \Sigma \rho_i (u_i + u_0) = fbcz1/2(\rho) \)
cop copy value of last physical point to all ghost cells
nil do nothing; assume that everything is set
The following tables list which parameters from each Namelist are required (●), optional (○) or irrelevant (blank). The distinction is made between required and optional where by a parameter requires a setting if the default value would give an invalid or degenerate case for the initial condition.

<table>
<thead>
<tr>
<th>inituu</th>
<th>ampluu</th>
<th>widthuu</th>
<th>urand</th>
<th>uu.left</th>
<th>uu.right</th>
<th>uu.upper</th>
<th>uu.lower</th>
<th>uy.left</th>
<th>uy.right</th>
<th>kx</th>
<th>ky</th>
<th>kz</th>
<th>uu</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gaussian-noise</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gaussian-noise-x</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xjump</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beltrami-x</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beltrami-y</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beltrami-z</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinear-x</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinear-y</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinear-z</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cos-cos-sin-uz</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tor_pert</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinear-x</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sound-wave</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shock-tube</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bullets</td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alfven-circ-x</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>const-ux</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>const-uy</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tang-discont-z</td>
<td>○</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fourier-trunc</td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>up-down</td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References


Part IV

Indexes
File Index

*.c ........................ 12
*.f90 ........................ 12
*.in ................... 32 171
*.local ................. 32
../32c ..................... 145
/host-ID ................. 18
.adapt-mkfile.inc ...... 77
.bashrc .................... 5
.conf ........................ 19
.cshrc ........................ 5
.emacs ........................ 125
.svn/ ........................ 12
/scratch/ .................... 7
<ID> ......................... 17
$HOME/.pencil/config-local 18
$HOME/.pencil/host-ID 18
$PENCIL_HOME ...... 87
$PENCIL_HOME/config 18
$PENCIL_HOME/config-local 18
$PENCIL_HOME/config/locals 18
$PENCIL_HOME/config/compilers/*.conf 17
$PENCIL_HOME/config/hosts 17
$PENCIL_HOME/host-ID 18
$PENCIL_HOME/python 46
${PENCIL_HOME}/config-local/bottig .................. 18
~/idl_history ............ 44
~/pencil-auto-test 106
1D_loop.f90 ............. 186
1d-test/ambipolar-diffusion conv-slab/ ....... 11
2d-tests/baroclinic 105
2d-tests/battery_term 85
cparam.inc .. 14 32 78
cparam.local ... 6 12
14 22 41 49 54
100 145
cparam_pencils.inc 102
timeavg.local ... 12 30
31
data/7 14 27 33 50 86
data/ ....................... 15
data/dim.dat ....... 165
165
data/procN/ ...... 27 28
data/proc/*/slice* 27 28
data/proc*/alive.info 26
data/proc*/slice* .. 27
data/proc*/var.dat .40
datadir.in ... 7 14 15
deebug_c.c .......... 109
density.f90 .... 31 66
data/dim.dat .... 15 16 32
div {xz,yz,xy,xy2} 215
dustdensity.f90 .... 189
dustdensity.f90 ... 189
dx/basic .................. 41
dx/macros/ ............. 32
dx/macros/ ............. 32
detontenate.f90 ..... 189
dim.dat ............ 15 16 32
div {xz,yz,xy,xy2} 215
doc/ ....................... 12
dustdensity.f90 ... 189
dustdensity.f90 ... 189
dx/macros/ ............. 32
dx/basic .................. 41
dx/macros/ ............. 32
detontenate.f90 ..... 189
entropy.f90 109 181
215 216 219
224 225 227 230
entropy_anelastic.f90
189 216
eos_ionization.f90 215
Ent ....................... 95
equ.f90 ................. 98
examples/pro/ ........... 42
examples/pro/ ........... 42
<table>
<thead>
<tr>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>experimental</td>
<td>126</td>
</tr>
<tr>
<td>forced/</td>
<td>12</td>
</tr>
<tr>
<td>forced/idl/</td>
<td>55</td>
</tr>
<tr>
<td>fourier_fftpack.f90</td>
<td>53</td>
</tr>
<tr>
<td>fourier_transform_y</td>
<td>53</td>
</tr>
<tr>
<td>generate_kvectors.pro</td>
<td>148</td>
</tr>
<tr>
<td>getconf.csh</td>
<td>9, 12, 14, 33</td>
</tr>
<tr>
<td>gravitational_waves.f90</td>
<td>190</td>
</tr>
<tr>
<td>gravitational_waves_hij6.f90</td>
<td>190</td>
</tr>
<tr>
<td>gravity_simple.f90</td>
<td>190, 221, 224, 226, 231</td>
</tr>
<tr>
<td>gravz/</td>
<td>12</td>
</tr>
<tr>
<td>grid.dat</td>
<td>16, 23</td>
</tr>
<tr>
<td>heatflux.f90</td>
<td>191</td>
</tr>
<tr>
<td>helical-MHDturb</td>
<td>35, 148</td>
</tr>
<tr>
<td>host_ID.conf</td>
<td>17</td>
</tr>
<tr>
<td>hosts/</td>
<td>18</td>
</tr>
<tr>
<td>hsections.pro</td>
<td>149</td>
</tr>
<tr>
<td>hydro.f90</td>
<td>31, 66, 76, 95, 100, 178, 214, 215, 217, 223, 225, 227, 229</td>
</tr>
<tr>
<td>hyperresi_strict_2nd</td>
<td>137</td>
</tr>
<tr>
<td>hypervisc_strict_2nd</td>
<td>137</td>
</tr>
<tr>
<td>idl/</td>
<td>12, 43</td>
</tr>
<tr>
<td>index.pro</td>
<td>15, 32</td>
</tr>
<tr>
<td>Intel.conf</td>
<td>17</td>
</tr>
<tr>
<td>Intel_MPI.conf</td>
<td>17</td>
</tr>
<tr>
<td>interlocked-fluxrings</td>
<td>105</td>
</tr>
<tr>
<td>interstellar.dat</td>
<td>15</td>
</tr>
<tr>
<td>Isurf.xz</td>
<td>215</td>
</tr>
<tr>
<td>j2.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>jb.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>jj[xyz].{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>k.dat</td>
<td>32, 148</td>
</tr>
<tr>
<td>K_VECTORS/</td>
<td>148</td>
</tr>
<tr>
<td>kinematic/</td>
<td>12</td>
</tr>
<tr>
<td>legend.dat</td>
<td>15</td>
</tr>
<tr>
<td>lncc.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>lnrho.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>lnrho.net</td>
<td>42</td>
</tr>
<tr>
<td>lnTT.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>lorenz_gauge.f90</td>
<td>191</td>
</tr>
<tr>
<td>mach.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>magnetic_shearboxJ.f90</td>
<td>191, 217, 221, 224, 226, 228, 231</td>
</tr>
<tr>
<td>Makefile</td>
<td>13, 21, 22, 31, 32, 78, 79, 122</td>
</tr>
<tr>
<td>Makefile.local</td>
<td>viii, 6, 12–14, 32, 49, 52, 54, 62, 64, 77, 78, 88</td>
</tr>
<tr>
<td>Makefile.src</td>
<td>6, 12, 14, 21, 78</td>
</tr>
<tr>
<td>meanfield.f90</td>
<td>195, 222, 229</td>
</tr>
<tr>
<td>meanfield_demfdt.f90</td>
<td>195</td>
</tr>
<tr>
<td>mkcpparam</td>
<td>14, 78, 102</td>
</tr>
<tr>
<td>mpicomm.f90</td>
<td>79</td>
</tr>
<tr>
<td>neutraldensity.f90</td>
<td>66</td>
</tr>
<tr>
<td>neutralvelocity.f90</td>
<td>66</td>
</tr>
<tr>
<td>NEWDIR</td>
<td>34, 83</td>
</tr>
<tr>
<td>nochiral</td>
<td>78</td>
</tr>
<tr>
<td>noentropy.f90</td>
<td>196</td>
</tr>
<tr>
<td>NOERASE</td>
<td>86</td>
</tr>
<tr>
<td>noinitial_condition.f90</td>
<td>104</td>
</tr>
<tr>
<td>noionization.f90</td>
<td>174</td>
</tr>
<tr>
<td>nomagnetic.f90</td>
<td>viii, 99, 98</td>
</tr>
<tr>
<td>nompicomm.f90</td>
<td>79, 111</td>
</tr>
<tr>
<td>nospecial.f90</td>
<td>103</td>
</tr>
<tr>
<td>o2.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>oo[xyz].{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>os/Unix.conf</td>
<td>19</td>
</tr>
<tr>
<td>param.nml</td>
<td>15, 32, 145</td>
</tr>
<tr>
<td>param2.nml</td>
<td>15, 32</td>
</tr>
<tr>
<td>params.log</td>
<td>15, 33, 34</td>
</tr>
<tr>
<td>particles_chemistry.f90</td>
<td>196</td>
</tr>
<tr>
<td>particles_dust.f90</td>
<td>196</td>
</tr>
<tr>
<td>particles_dust_shearboxJ.f90</td>
<td>196</td>
</tr>
<tr>
<td>particles_lagrangian_f90</td>
<td>196</td>
</tr>
<tr>
<td>particles_mass_shearboxJ.f90</td>
<td>196</td>
</tr>
<tr>
<td>particles_shearboxJ.f90</td>
<td>196</td>
</tr>
<tr>
<td>pc_read_phiavg.pro</td>
<td>30</td>
</tr>
<tr>
<td>pc_read_video</td>
<td>28</td>
</tr>
<tr>
<td>pc_read_xyaver</td>
<td>29</td>
</tr>
<tr>
<td>pc_setupsrcc</td>
<td>6</td>
</tr>
<tr>
<td>pencil-code/</td>
<td>11, 12, 87</td>
</tr>
<tr>
<td>pencil-code/doc/manual.tex</td>
<td>90, 104</td>
</tr>
<tr>
<td>pencil-code/idl/read</td>
<td>8</td>
</tr>
<tr>
<td>pencil-code/license/developers</td>
<td>3, 91</td>
</tr>
<tr>
<td>pencil-code/utils</td>
<td>33</td>
</tr>
<tr>
<td>pencil-runs/</td>
<td>12</td>
</tr>
<tr>
<td>Pencil::ConfigFinder</td>
<td>17</td>
</tr>
<tr>
<td>Pencil::ConfigParser</td>
<td>17</td>
</tr>
<tr>
<td>phiaver.in</td>
<td>29, 216</td>
</tr>
<tr>
<td>phiaverages.dat</td>
<td>29</td>
</tr>
<tr>
<td>PHIAVG.n</td>
<td>30</td>
</tr>
<tr>
<td>phiavg.pro</td>
<td>30</td>
</tr>
<tr>
<td>polymer.f90</td>
<td>197</td>
</tr>
<tr>
<td>power</td>
<td>53</td>
</tr>
<tr>
<td>power.pro</td>
<td>54</td>
</tr>
<tr>
<td>power_kin.dat</td>
<td>52</td>
</tr>
<tr>
<td>power_krms.dat</td>
<td>53</td>
</tr>
<tr>
<td>powerbx_x.dat</td>
<td>52</td>
</tr>
<tr>
<td>powerhel_magnet.dat</td>
<td>52</td>
</tr>
<tr>
<td>powerux_x.dat</td>
<td>52</td>
</tr>
<tr>
<td>Poynting[xyz].{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>pp.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>print.in</td>
<td>8, 14, 25, 98, 99, 146, 178</td>
</tr>
<tr>
<td>procN</td>
<td>15, 32, 33</td>
</tr>
<tr>
<td>proc0</td>
<td>15</td>
</tr>
<tr>
<td>proc1</td>
<td>15</td>
</tr>
<tr>
<td>procN/</td>
<td>108</td>
</tr>
<tr>
<td>pscalar.f90</td>
<td>186, 215</td>
</tr>
<tr>
<td>Qyav{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>File Name</td>
<td>Pages</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>solid_cells_CGEO.f90</td>
<td>197</td>
</tr>
<tr>
<td>solid_cells_reactive.f90</td>
<td>197</td>
</tr>
<tr>
<td>sound-spherical-noequi</td>
<td>24</td>
</tr>
<tr>
<td>sourceme</td>
<td>5, 75</td>
</tr>
<tr>
<td>sourceme.sh</td>
<td>5, 12, 75</td>
</tr>
<tr>
<td>SPEED</td>
<td>86</td>
</tr>
<tr>
<td>spher/</td>
<td>12</td>
</tr>
<tr>
<td>src/</td>
<td>14</td>
</tr>
<tr>
<td>src/* .local</td>
<td>32</td>
</tr>
<tr>
<td>src/cparam.local</td>
<td>145</td>
</tr>
<tr>
<td>src/Makefile.inc</td>
<td>52</td>
</tr>
<tr>
<td>src/Makefile.local</td>
<td>38</td>
</tr>
<tr>
<td>src/read_all_-videofiles.x</td>
<td>27</td>
</tr>
<tr>
<td>src/read_videofiles.x</td>
<td>27, 28</td>
</tr>
<tr>
<td>ss. {xz, yz, xy, xy2}</td>
<td>215</td>
</tr>
<tr>
<td>ss.net</td>
<td>42</td>
</tr>
<tr>
<td>start.csh</td>
<td>9, 12, 14, 31</td>
</tr>
<tr>
<td>start.csh</td>
<td>33, 39, 66, 82</td>
</tr>
<tr>
<td>start.in</td>
<td>8, 14</td>
</tr>
<tr>
<td>start.pro</td>
<td>32, 44, 86</td>
</tr>
<tr>
<td>start.x</td>
<td>8, 15, 39, 41, 77</td>
</tr>
<tr>
<td>start.x</td>
<td>82, 107, 175</td>
</tr>
<tr>
<td>STOP</td>
<td>33</td>
</tr>
<tr>
<td>structure.pro</td>
<td>55</td>
</tr>
<tr>
<td>sub.f90</td>
<td>96</td>
</tr>
<tr>
<td>TAVG</td>
<td>31</td>
</tr>
<tr>
<td>temperature_idealgas.f90</td>
<td>197, 215, 222</td>
</tr>
<tr>
<td>temperature_idealgas.f90</td>
<td>224, 227, 229, 232</td>
</tr>
<tr>
<td>temperature_ionization.f90</td>
<td>198, 223</td>
</tr>
<tr>
<td>testfield_axisym.f90</td>
<td>198</td>
</tr>
<tr>
<td>testfield_axisym2.f90</td>
<td>199</td>
</tr>
<tr>
<td>testfield_axisym3.f90</td>
<td></td>
</tr>
<tr>
<td>testfield_axisym4.f90</td>
<td>199</td>
</tr>
<tr>
<td>testfield_compress_-z.f90</td>
<td>199</td>
</tr>
<tr>
<td>testfield_meri.f90</td>
<td>202</td>
</tr>
<tr>
<td>testfield_nonlin_z.f90</td>
<td>203</td>
</tr>
<tr>
<td>testfield_x.f90</td>
<td>205</td>
</tr>
<tr>
<td>testfield_xz.f90</td>
<td>207</td>
</tr>
<tr>
<td>testfield_z.f90</td>
<td>207</td>
</tr>
<tr>
<td>testflow_z.f90</td>
<td>209</td>
</tr>
<tr>
<td>testperturb.f90</td>
<td>209</td>
</tr>
<tr>
<td>testscalar.f90</td>
<td>210</td>
</tr>
<tr>
<td>testscalar_axisym.f90</td>
<td>211</td>
</tr>
<tr>
<td>testscalar_simple.f90</td>
<td>212</td>
</tr>
<tr>
<td>thermal_energy.f90</td>
<td>214</td>
</tr>
<tr>
<td>time.dat</td>
<td>16, 40</td>
</tr>
<tr>
<td>time_series.dat</td>
<td>8, 14</td>
</tr>
<tr>
<td>ts.pro</td>
<td>42, 44, 149</td>
</tr>
<tr>
<td>tsnap.dat</td>
<td>15, 172</td>
</tr>
<tr>
<td>TT. {xz, yz, xy, xy2}</td>
<td>215</td>
</tr>
<tr>
<td>tvtid.dat</td>
<td>15</td>
</tr>
<tr>
<td>u2. {xz, yz, xy, xy2}</td>
<td>214</td>
</tr>
<tr>
<td>u[xyz]. {xz, yz, xy, xy2}</td>
<td>214</td>
</tr>
<tr>
<td>uu.net</td>
<td>42</td>
</tr>
<tr>
<td>VAR</td>
<td>40, 86, 173</td>
</tr>
<tr>
<td>VARV</td>
<td>15, 26, 32, 33</td>
</tr>
<tr>
<td>VARV</td>
<td>36, 172</td>
</tr>
<tr>
<td>var.dat</td>
<td>8, 15, 16, 26, 31, 32</td>
</tr>
<tr>
<td>var.dat</td>
<td>34, 36, 40, 43, 45</td>
</tr>
<tr>
<td>var.dat</td>
<td>86, 144, 164, 172</td>
</tr>
<tr>
<td>var.general</td>
<td>173</td>
</tr>
<tr>
<td>VAR0</td>
<td>164</td>
</tr>
<tr>
<td>VAR1</td>
<td>144</td>
</tr>
<tr>
<td>VAR#</td>
<td>34</td>
</tr>
<tr>
<td>video.in</td>
<td>26, 28, 146, 214</td>
</tr>
<tr>
<td>visc_smagorinsky.f90</td>
<td>214</td>
</tr>
<tr>
<td>File</td>
<td>Lines</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------</td>
</tr>
<tr>
<td>viscosity.f90</td>
<td>214, 223</td>
</tr>
<tr>
<td></td>
<td>227, 232</td>
</tr>
<tr>
<td>vsections.pro</td>
<td>150</td>
</tr>
<tr>
<td>vsections2.pro</td>
<td>150</td>
</tr>
<tr>
<td>X.xy</td>
<td>16</td>
</tr>
<tr>
<td>X.xz</td>
<td>16</td>
</tr>
<tr>
<td>xyaver.in</td>
<td>29, 146, 217</td>
</tr>
<tr>
<td>xzaver.in</td>
<td>29, 223</td>
</tr>
<tr>
<td>yaver.in</td>
<td>29, 227</td>
</tr>
<tr>
<td>yaverages.dat</td>
<td>29</td>
</tr>
<tr>
<td>yH.{xz,yz,xy,xy2}</td>
<td>215</td>
</tr>
<tr>
<td>yzaver.in</td>
<td>29, 224</td>
</tr>
<tr>
<td>yzaverages.dat</td>
<td>29</td>
</tr>
<tr>
<td>zaver.in</td>
<td>29, 229</td>
</tr>
<tr>
<td>zaverages.dat</td>
<td>29</td>
</tr>
</tbody>
</table>
Variable Index

.true. 29  
0ds 237 239  
1s 233-237 239  
1so 233 234  
0 232 234 237 239  
1 233 234 236-238 240  
a 232 234-237 239  
a0d 237 239  
a11xy 202  
a12xy 202  
a13xy 202  
a1s 238  
a2 232 234-237 239  
a21xy 202  
a22xy 202  
a23xy 202  
a2m 184 193  
a2r 232 234  
a2v 232 237  
a31xy 202  
a32xy 202  
a33xy 202  
aa 215  
ab 215  
ab_int 182 191  
ABC.A 176  
ABC_B 176  
ABC_C 176  
abm 182 191  
abmh 182 191  
abmn 182 191  
abms 182 191  
abmxy 231  
abmx 220 222  
abrms 182 191  
abumx 182 191  
abumy 182 191  
abumz 182 191  
abuxmz 219 221  
abuymz 219 221  
abuzmz 219 221  
accpowzdownmz 218  
accpowzmz 218  
accpowzupmz 218  
af 232 237 239  
ajm 182 191  
aklam 209  
aklamQ 209  
alp11 200 203 205 207  
alp11x 205  
alp11x2 206  
alp11cc 200 203 205  
alp11ccs 207  
alp11z 208  
alp12 200 203 205 207  
alp12x 206  
alp122xy 206  
alp12cs 200 203 205 207  
alp13 208  
alp13z 208  
alp21 200 203 205 207  
alp21x 205  
alp212xy 206  
alp21sc 200 203 205  
alp21scs 207  
alp22 206  
alp22x 206  
alp222xy 206  
alp222sc 200 203 205  
alp222scs 207  
alp222z 208  
alp23 207  
alp233z 208  
alp31 200 203 205 207  
alp32 200 203 205 207  
alp322xy 206  
alp322sc 200 203 205  
alp322scs 207  
alpK 200 203  
alpM 200 203  
alpm 195  
alpMK 200 203  
alpmxz 229  
alpPARA 198 199  
alpPARAz 198 199  
alpPERP 198 199  
alpPERPz 198 199  
amax 184 195  
amb 185 194  
ambn 185 194  
ambmn 185 194  
ambms 185 194  
ampl_ff 177  
ampl_force 174  
ampl_ss 168  
amplaa 170  
amplaa2 170  
amplincc 170  
amplincc2 170  
amplinrhm 167  
ampluu 128 166  
ampluu=1e-1 128  
ant 233 234  
ap 235 236  
apbrms 187  
arms 184 193  
asT 233 234 236-238 240  
axmxy 230 231  
axmz 227 228  
axpx 219 221  
axpt 183 193  
aypt 183 192  
aypt 183 192  
aypx 227 228  
aypx 227 228  
aymz 219 221  
aypz 183 193  
aypt 183 192  
azm 219 221  
azmz 219 221  
azpz 183 193  
azpt 183 192  
b0rms 201 204 206 208  
b1 238 240  
b111xy 202  
b112xy 203
<table>
<thead>
<tr>
<th>VARIABLE INDEX</th>
<th>251</th>
</tr>
</thead>
<tbody>
<tr>
<td>b11rms ..........</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>206</td>
</tr>
<tr>
<td>b121xy ..........</td>
<td>202</td>
</tr>
<tr>
<td>b122xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b12rms ..........</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>208</td>
</tr>
<tr>
<td>b131xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b132xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b1b23m ..........</td>
<td>185</td>
</tr>
<tr>
<td>b1b32m ..........</td>
<td>185</td>
</tr>
<tr>
<td>b1m ............</td>
<td>182</td>
</tr>
<tr>
<td>b1rms ...........</td>
<td>198</td>
</tr>
<tr>
<td>b2 ............</td>
<td>215</td>
</tr>
<tr>
<td>b21lxy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b212xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b21rms ..........</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>206</td>
</tr>
<tr>
<td>b221xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b222xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b22rms ..........</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>208</td>
</tr>
<tr>
<td>b231xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b232xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b2b13m ..........</td>
<td>185</td>
</tr>
<tr>
<td>b2b31m ..........</td>
<td>185</td>
</tr>
<tr>
<td>b2divum ..........</td>
<td>186</td>
</tr>
<tr>
<td>b2m ............</td>
<td>182</td>
</tr>
<tr>
<td>b2mphi ..........</td>
<td>216</td>
</tr>
<tr>
<td>b2mx ..........</td>
<td>225</td>
</tr>
<tr>
<td>b2mzx ..........</td>
<td>227</td>
</tr>
<tr>
<td>b2mz ..........</td>
<td>220</td>
</tr>
<tr>
<td>b2rms ..........</td>
<td>198</td>
</tr>
<tr>
<td>b2ruzm ..........</td>
<td>182</td>
</tr>
<tr>
<td>b2tm ..........</td>
<td>182</td>
</tr>
<tr>
<td>b2uzm ..........</td>
<td>182</td>
</tr>
<tr>
<td>b3 ............</td>
<td>238</td>
</tr>
<tr>
<td>b311xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b312xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b312xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b313xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b332xy ..........</td>
<td>203</td>
</tr>
<tr>
<td>b3b12m ..........</td>
<td>185</td>
</tr>
<tr>
<td>b3b21m ..........</td>
<td>185</td>
</tr>
<tr>
<td>b3rms ..........</td>
<td>198</td>
</tr>
<tr>
<td>b4m ............</td>
<td>182</td>
</tr>
<tr>
<td>B_ext ..........</td>
<td>176</td>
</tr>
<tr>
<td>bamp ..........</td>
<td>201</td>
</tr>
<tr>
<td>bb .............</td>
<td>109</td>
</tr>
<tr>
<td>bnmphi ..........</td>
<td>216</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VARIABLE INDEX</th>
<th>251</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbmsphmphi ......</td>
<td>216</td>
</tr>
<tr>
<td>bbxmx ...........</td>
<td>184</td>
</tr>
<tr>
<td>bbxmxz ..........</td>
<td>219</td>
</tr>
<tr>
<td>bbymax ..........</td>
<td>184</td>
</tr>
<tr>
<td>bbymz ...........</td>
<td>219</td>
</tr>
<tr>
<td>bbzmax ..........</td>
<td>184</td>
</tr>
<tr>
<td>bbzmz ..........</td>
<td>220</td>
</tr>
<tr>
<td>bc(x,y,z) ........</td>
<td>39</td>
</tr>
<tr>
<td>bcosphz ..........</td>
<td>185</td>
</tr>
<tr>
<td>bcx ..........</td>
<td>39</td>
</tr>
<tr>
<td>bcy ..........</td>
<td>39</td>
</tr>
<tr>
<td>bcz ..........</td>
<td>39</td>
</tr>
<tr>
<td>bet1 ..........</td>
<td>215</td>
</tr>
<tr>
<td>beta1 ..........</td>
<td>184</td>
</tr>
<tr>
<td>beta1max ..........</td>
<td>184</td>
</tr>
<tr>
<td>beta1mxy ..........</td>
<td>231</td>
</tr>
<tr>
<td>beta1mx ..........</td>
<td>220</td>
</tr>
<tr>
<td>beta2m ...........</td>
<td>220</td>
</tr>
<tr>
<td>beta2mz ..........</td>
<td>220</td>
</tr>
<tr>
<td>betam ..........</td>
<td>184</td>
</tr>
<tr>
<td>betamax ..........</td>
<td>184</td>
</tr>
<tr>
<td>betamin ..........</td>
<td>184</td>
</tr>
<tr>
<td>betamx ..........</td>
<td>220</td>
</tr>
<tr>
<td>betamz ..........</td>
<td>198</td>
</tr>
<tr>
<td>betPARA ..........</td>
<td>198</td>
</tr>
<tr>
<td>betPARAz ........</td>
<td>198</td>
</tr>
<tr>
<td>betPERP ..........</td>
<td>198</td>
</tr>
<tr>
<td>betPERP2 ..........</td>
<td>199</td>
</tr>
<tr>
<td>betPERPz ..........</td>
<td>198</td>
</tr>
<tr>
<td>bfrms ..........</td>
<td>184</td>
</tr>
<tr>
<td>bgmu5rms ..........</td>
<td>188</td>
</tr>
<tr>
<td>bgtheta5rms .......</td>
<td>188</td>
</tr>
<tr>
<td>bjtm ..........</td>
<td>182</td>
</tr>
<tr>
<td>bm .............</td>
<td>182</td>
</tr>
<tr>
<td>bm2 ..........</td>
<td>182</td>
</tr>
<tr>
<td>bmax ..........</td>
<td>184</td>
</tr>
<tr>
<td>bmin ..........</td>
<td>184</td>
</tr>
<tr>
<td>bmx ...........</td>
<td>184</td>
</tr>
<tr>
<td>bmxz ..........</td>
<td>184</td>
</tr>
<tr>
<td>bmys ...........</td>
<td>184</td>
</tr>
<tr>
<td>bmz ...........</td>
<td>184</td>
</tr>
<tr>
<td>bmz2 ..........</td>
<td>184</td>
</tr>
<tr>
<td>bmzphi ..........</td>
<td>185</td>
</tr>
<tr>
<td>bmzphe ..........</td>
<td>185</td>
</tr>
<tr>
<td>bmzs2 ..........</td>
<td>184</td>
</tr>
<tr>
<td>by0mz ..........</td>
<td>202</td>
</tr>
<tr>
<td>by0pt ..........</td>
<td>201</td>
</tr>
</tbody>
</table>
The Pencil Code

by11pt ... 200 204 208
by12pt ... 200 204 208
by1mxz ... 228
by21pt ... 200 204 208
by22pt ... 201 204 208
by2m ... 185 187 194
by2mx ... 226
by2mxy ... 231
by2mxz ... 228
by2my ... 224
by2mx ... 220–222
by2rmz ... 220–222
by2mz ... 187
by2mzx ... 226
by2mzy ... 231
by2mzx ... 228
by2my ... 224
bybzmz ... 220–222
bym ... 184 187 193
bymax ... 184 187 193
bym ... 184 187 193
bymx ... 226
bymxy ... 230–231
bymxz ... 228
bmy ... 224
bymz ... 220–221
byp2 ... 183–192
bypt ... 183 192
bz0mz ... 202 205 206 209
bz1mxz ... 228
bz2m ... 185 187 194
bz2mx ... 226
bz2mxy ... 231
bz2mxz ... 228
bz2my ... 224
bz2mz ... 220–222
bz2rmz ... 220–222
bz ... 184 187 193
bzmax ... 184 187 193
bzmin ... 184 187 193
bzmphi ... 216 217
bzm ... 226
bzmxy ... 230 231
bzmxz ... 228
bzmy ... 224
bzmz ... 220 221
bzp2 ... 183 192
bzpt ... 183 192
c+k ... 236 237
cl ... 233 234 238 239
c1pt ... 211–213
clrms ... 210 212–213
c1s ... 238 239
c2 ... 238 240
c2pt ... 211–213
c2rms ... 210 212–213
c3 ... 238 239
cl ... 238 239
c3pt ... 211–213
c3rms ... 210 212–213
c4pt ... 211–213
c4rms ... 210 212–213
c5pt ... 211–213
c5rms ... 210 212–213
c6pt ... 211–213
c6rms ... 210 211–213
cglnrm ... 186
cmax ... 186
ediffrho ... 174
cds ... 235 236
ckt ... 38 41 172
cdts ... 38
ct ... 38
ctlv ... 38 172
cdz ... 238 240
ce ... 238 240
cfb ... 234 239
chi ... 175
chit ... 175
coeff_grid ... 23 25
cool ... 175
cooltype ... 175
cop ... 233 234 236–237
239 240
cosjbm ... 186 195
cosubm ... 183 192
cp ... 238 240
cpc ... 232 234
cpp ... 232 234
crz ... 232 234
crk ... 236
cs0 ... 167 174
cs ... 167 174
cs2 ... 109
clsbot ... 167 174
cs2cool ... 175
cs2mphi ... 216 217
cs2top ... 167 174
csm ... 181 189 198
cT ... 233 236 238 239
ct1 ... 236 238
ct2 ... 238 239
ct3 ... 238 239
c tz ... 238 240
curlr2mx ... 217
cutoff=0 ... 128
cvsid ... 164 171
d1s ... 233 235 237
d2aug ... 30 173
d2lambrms ... 187
d2lammr ... 187
d6am ... 220 222
d6amz1 ... 220 222
d6amz2 ... 220 222
d6amz3 ... 220 222
damp ... 173 174
dampu ... 173
dampuext ... 173
dampuext ... 173
datadir ... 173
db ... 238 240
dbxm ... 187
dbxm ... 187
dby2m ... 187
dby2m ... 187
dbym ... 187
dbym ... 187
dby ... 187
dby ... 187
d ... 233 234 236 238 240
div ... 173 179
div ... 173
div ... 173
dcoolk ... 225
dcoolk ... 230
dcool ... 219
del ... 198 199
del2 ... 199
deltay ... 197
delz ... 198 199
der ... 233 234 236–238
240
detn ... 189
det ... 189
det ... 189
dexbmx ... 186 195
dexbmy ... 186 195
dexbmz ... 186 195
dheat_buffer1 ... 175
div ... 238 240
div ... 187
div ... 187
div ... 187
div ... 187
div ... 187
divh ... 187
divh ... 187

divh ... 187
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>epotuxmx</td>
<td>226</td>
</tr>
<tr>
<td>epotuxmxy</td>
<td>231</td>
</tr>
<tr>
<td>epotuzmz</td>
<td>221</td>
</tr>
<tr>
<td>epsAD</td>
<td>183, 192</td>
</tr>
<tr>
<td>epsilonaa</td>
<td>170</td>
</tr>
<tr>
<td>epsK</td>
<td>214</td>
</tr>
<tr>
<td>epsKmz</td>
<td>223</td>
</tr>
<tr>
<td>epsM</td>
<td>183, 192</td>
</tr>
<tr>
<td>epsMmz</td>
<td>221, 222</td>
</tr>
<tr>
<td>eruzmz</td>
<td>223</td>
</tr>
<tr>
<td>etajmax</td>
<td>186, 195</td>
</tr>
<tr>
<td>etajrhomax</td>
<td>186, 195</td>
</tr>
<tr>
<td>etasmagm</td>
<td>186, 195</td>
</tr>
<tr>
<td>etasmagmax</td>
<td>186, 195</td>
</tr>
<tr>
<td>etasmagmin</td>
<td>186, 195</td>
</tr>
<tr>
<td>etam</td>
<td>195</td>
</tr>
<tr>
<td>etatotalm</td>
<td>226</td>
</tr>
<tr>
<td>etatotalmz</td>
<td>220, 222</td>
</tr>
<tr>
<td>etavamax</td>
<td>186, 195</td>
</tr>
<tr>
<td>ethm</td>
<td>181, 189, 196, 198</td>
</tr>
<tr>
<td>ethmax</td>
<td>214</td>
</tr>
<tr>
<td>ethn</td>
<td>189</td>
</tr>
<tr>
<td>ethmcr</td>
<td>214</td>
</tr>
<tr>
<td>ethmz</td>
<td>219, 223</td>
</tr>
<tr>
<td>ethtot</td>
<td>181, 190, 198, 214</td>
</tr>
<tr>
<td>etot</td>
<td>214</td>
</tr>
<tr>
<td>ex</td>
<td>238, 240</td>
</tr>
<tr>
<td>Ex0pt</td>
<td>201, 204, 208</td>
</tr>
<tr>
<td>Ex11pt</td>
<td>201, 204, 208</td>
</tr>
<tr>
<td>Ex12pt</td>
<td>201, 204, 208</td>
</tr>
<tr>
<td>Ex21pt</td>
<td>201, 204, 208</td>
</tr>
<tr>
<td>Ex22pt</td>
<td>201, 204, 208</td>
</tr>
<tr>
<td>examx</td>
<td>185, 194</td>
</tr>
<tr>
<td>examxy1</td>
<td>231</td>
</tr>
<tr>
<td>examxy2</td>
<td>231, 232</td>
</tr>
<tr>
<td>examxy3</td>
<td>231, 232</td>
</tr>
<tr>
<td>examy</td>
<td>185, 194</td>
</tr>
<tr>
<td>examz</td>
<td>185, 194</td>
</tr>
<tr>
<td>examz1</td>
<td>220, 222</td>
</tr>
<tr>
<td>examz2</td>
<td>220, 222</td>
</tr>
<tr>
<td>examz3</td>
<td>220, 222</td>
</tr>
<tr>
<td>exatop</td>
<td>183, 193</td>
</tr>
<tr>
<td>exd</td>
<td>238, 240</td>
</tr>
<tr>
<td>exf</td>
<td>238, 240</td>
</tr>
<tr>
<td>exjm</td>
<td>185, 195</td>
</tr>
<tr>
<td>exjmy</td>
<td>185, 195</td>
</tr>
<tr>
<td>exjmz</td>
<td>185, 195</td>
</tr>
<tr>
<td>exem</td>
<td>238, 240</td>
</tr>
<tr>
<td>Exmxy</td>
<td>186, 195</td>
</tr>
<tr>
<td>Exmxz</td>
<td>186, 195</td>
</tr>
<tr>
<td>Exmxz</td>
<td>228</td>
</tr>
<tr>
<td>Exmz</td>
<td>220, 222</td>
</tr>
<tr>
<td>Exp2</td>
<td>183, 192</td>
</tr>
<tr>
<td>Expt</td>
<td>183, 192</td>
</tr>
<tr>
<td>Ey0pt</td>
<td>201, 205, 208</td>
</tr>
<tr>
<td>Ey11pt</td>
<td>201, 204, 208</td>
</tr>
<tr>
<td>Ey12pt</td>
<td>201, 205, 208</td>
</tr>
<tr>
<td>Ey21pt</td>
<td>201, 204, 208</td>
</tr>
<tr>
<td>Ey22pt</td>
<td>201, 205, 208</td>
</tr>
<tr>
<td>Variable</td>
<td>Page(s)</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>fpsrsymz</td>
<td>223</td>
</tr>
<tr>
<td>fpsrszmz</td>
<td>223</td>
</tr>
<tr>
<td>fradbot</td>
<td>181, 190, 198</td>
</tr>
<tr>
<td>fradmx</td>
<td>225</td>
</tr>
<tr>
<td>fradmz</td>
<td>223</td>
</tr>
<tr>
<td>fradtop</td>
<td>181, 190, 197</td>
</tr>
<tr>
<td>fradx_kramers</td>
<td>225</td>
</tr>
<tr>
<td>fradxy_kprof</td>
<td>230</td>
</tr>
<tr>
<td>fradykramers</td>
<td>230</td>
</tr>
<tr>
<td>fradymxy_kprof</td>
<td>230</td>
</tr>
<tr>
<td>fradz</td>
<td>219</td>
</tr>
<tr>
<td>fradz_constchi</td>
<td>219</td>
</tr>
<tr>
<td>fradz_kprof</td>
<td>219</td>
</tr>
<tr>
<td>fradz_kramers</td>
<td>219</td>
</tr>
<tr>
<td>fring1, fring2</td>
<td>170</td>
</tr>
<tr>
<td>fmax</td>
<td>197</td>
</tr>
<tr>
<td>fs</td>
<td>238, 240</td>
</tr>
<tr>
<td>fsum</td>
<td>38</td>
</tr>
<tr>
<td>fturbfz</td>
<td>219</td>
</tr>
<tr>
<td>fturbmx</td>
<td>225</td>
</tr>
<tr>
<td>fturbmz</td>
<td>219</td>
</tr>
<tr>
<td>fturbxy</td>
<td>230</td>
</tr>
<tr>
<td>fturbtxy</td>
<td>230</td>
</tr>
<tr>
<td>fturbtz</td>
<td>219</td>
</tr>
<tr>
<td>fturbz</td>
<td>219</td>
</tr>
<tr>
<td>fum</td>
<td>180</td>
</tr>
<tr>
<td>fviscm</td>
<td>214</td>
</tr>
<tr>
<td>fviscmax</td>
<td>214</td>
</tr>
<tr>
<td>fviscmin</td>
<td>214</td>
</tr>
<tr>
<td>fviscmx</td>
<td>227</td>
</tr>
<tr>
<td>fviscmxy</td>
<td>232</td>
</tr>
<tr>
<td>fviscmyz</td>
<td>223</td>
</tr>
<tr>
<td>fviscmyz</td>
<td>214</td>
</tr>
<tr>
<td>fviscmmxy</td>
<td>232</td>
</tr>
<tr>
<td>fviscmmz</td>
<td>232</td>
</tr>
<tr>
<td>fviscmyz</td>
<td>214</td>
</tr>
<tr>
<td>fviscmxy</td>
<td>232</td>
</tr>
<tr>
<td>fxbxm</td>
<td>183, 192</td>
</tr>
<tr>
<td>g</td>
<td>234, 235, 238, 240</td>
</tr>
<tr>
<td>g11pt</td>
<td>190</td>
</tr>
<tr>
<td>g12pt</td>
<td>190</td>
</tr>
<tr>
<td>g22pt</td>
<td>190</td>
</tr>
<tr>
<td>g32pt</td>
<td>190</td>
</tr>
<tr>
<td>g31pt</td>
<td>190</td>
</tr>
<tr>
<td>g33pt</td>
<td>190</td>
</tr>
<tr>
<td>gal</td>
<td>209</td>
</tr>
<tr>
<td>gam</td>
<td>198, 199</td>
</tr>
<tr>
<td>gam11</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam11z</td>
<td>210, 211, 212</td>
</tr>
<tr>
<td>gam12</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam12z</td>
<td>210, 211, 212</td>
</tr>
<tr>
<td>gam13</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam13z</td>
<td>210, 211, 212</td>
</tr>
<tr>
<td>gam21</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam21z</td>
<td>210, 211, 212</td>
</tr>
<tr>
<td>gam22</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam22z</td>
<td>210, 211, 212</td>
</tr>
<tr>
<td>gam23</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam23z</td>
<td>210, 212, 213</td>
</tr>
<tr>
<td>gam31</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam31z</td>
<td>210, 212, 213</td>
</tr>
<tr>
<td>gam32</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam32z</td>
<td>210, 212, 213</td>
</tr>
<tr>
<td>gam33</td>
<td>210, 212</td>
</tr>
<tr>
<td>gam33z</td>
<td>210, 212, 213</td>
</tr>
<tr>
<td>gam34</td>
<td>212</td>
</tr>
<tr>
<td>gamc</td>
<td>211</td>
</tr>
<tr>
<td>game</td>
<td>211</td>
</tr>
<tr>
<td>gamma</td>
<td>167, 174, 209</td>
</tr>
<tr>
<td>gammaQ</td>
<td>209</td>
</tr>
<tr>
<td>gamz</td>
<td>198, 199</td>
</tr>
<tr>
<td>gdivu2m</td>
<td>179</td>
</tr>
<tr>
<td>gg2m</td>
<td>190</td>
</tr>
<tr>
<td>ggT2m</td>
<td>190</td>
</tr>
<tr>
<td>ggTm</td>
<td>190</td>
</tr>
<tr>
<td>ggTpt2</td>
<td>190</td>
</tr>
<tr>
<td>ggTpt</td>
<td>190</td>
</tr>
<tr>
<td>ggTXm</td>
<td>190</td>
</tr>
<tr>
<td>ggX2m</td>
<td>190</td>
</tr>
<tr>
<td>ggXm</td>
<td>190</td>
</tr>
<tr>
<td>ggXp2</td>
<td>190</td>
</tr>
<tr>
<td>ggXpt</td>
<td>190</td>
</tr>
<tr>
<td>gLambm</td>
<td>186</td>
</tr>
<tr>
<td>gLamrms</td>
<td>187</td>
</tr>
<tr>
<td>gmu5mx</td>
<td>188</td>
</tr>
<tr>
<td>gmu5my</td>
<td>189</td>
</tr>
<tr>
<td>gmu5mz</td>
<td>189</td>
</tr>
<tr>
<td>gmu5rms</td>
<td>188</td>
</tr>
<tr>
<td>gpu_asteroth, f90</td>
<td>163</td>
</tr>
<tr>
<td>grads0</td>
<td>168</td>
</tr>
<tr>
<td>grav_ambient</td>
<td>168</td>
</tr>
<tr>
<td>grav_profile</td>
<td>129, 130</td>
</tr>
<tr>
<td>grav_tilt</td>
<td>168</td>
</tr>
<tr>
<td>gravz</td>
<td>129, 167, 177</td>
</tr>
<tr>
<td>ggeomax</td>
<td>181</td>
</tr>
<tr>
<td>grid_func</td>
<td>23</td>
</tr>
<tr>
<td>ghydrogen</td>
<td>190</td>
</tr>
<tr>
<td>gshockmax</td>
<td>197</td>
</tr>
<tr>
<td>gsrms</td>
<td>182</td>
</tr>
<tr>
<td>gtheta5alpha</td>
<td>188</td>
</tr>
<tr>
<td>gtheta5beta</td>
<td>188</td>
</tr>
<tr>
<td>gtheta5gamma</td>
<td>188</td>
</tr>
<tr>
<td>gtheta5mz</td>
<td>188</td>
</tr>
<tr>
<td>gtheta5rms</td>
<td>188</td>
</tr>
<tr>
<td>gT0m</td>
<td>197</td>
</tr>
<tr>
<td>gTmax</td>
<td>182, 197</td>
</tr>
<tr>
<td>gTmaxm</td>
<td>182</td>
</tr>
<tr>
<td>gTmaxzy</td>
<td>182</td>
</tr>
<tr>
<td>gTxgs2mx</td>
<td>230</td>
</tr>
<tr>
<td>gTxgs2mz</td>
<td>219</td>
</tr>
<tr>
<td>gTxgsymx</td>
<td>230</td>
</tr>
<tr>
<td>gTxgzy2mz</td>
<td>230</td>
</tr>
<tr>
<td>gTmaxz</td>
<td>230</td>
</tr>
<tr>
<td>gTyx</td>
<td>230</td>
</tr>
<tr>
<td>gymyx</td>
<td>230</td>
</tr>
<tr>
<td>gTm</td>
<td>230</td>
</tr>
<tr>
<td>guyTm</td>
<td>197</td>
</tr>
<tr>
<td>guygTm</td>
<td>197</td>
</tr>
<tr>
<td>gTgTm</td>
<td>230</td>
</tr>
<tr>
<td>gTgTm</td>
<td>230</td>
</tr>
<tr>
<td>gzirrhomz</td>
<td>219</td>
</tr>
<tr>
<td>h22rms</td>
<td>190</td>
</tr>
<tr>
<td>h23rms</td>
<td>190</td>
</tr>
<tr>
<td>h33rms</td>
<td>190</td>
</tr>
<tr>
<td>hat</td>
<td>233, 235</td>
</tr>
<tr>
<td>hcond0</td>
<td>174, 175</td>
</tr>
<tr>
<td>hcond1</td>
<td>174</td>
</tr>
<tr>
<td>hcond2</td>
<td>174</td>
</tr>
<tr>
<td>hds</td>
<td>238, 239</td>
</tr>
<tr>
<td>headt</td>
<td>108</td>
</tr>
<tr>
<td>height</td>
<td>108</td>
</tr>
<tr>
<td>heatmz</td>
<td>219</td>
</tr>
<tr>
<td>height_12</td>
<td>176</td>
</tr>
<tr>
<td>height_21</td>
<td>177</td>
</tr>
<tr>
<td>hhT2m</td>
<td>190</td>
</tr>
<tr>
<td>hhTthXm</td>
<td>190</td>
</tr>
<tr>
<td>hhTpt2</td>
<td>190</td>
</tr>
<tr>
<td>hhTpt</td>
<td>190</td>
</tr>
<tr>
<td>hhTXm</td>
<td>190</td>
</tr>
<tr>
<td>hhX2m</td>
<td>190</td>
</tr>
<tr>
<td>hhXpt2</td>
<td>190</td>
</tr>
<tr>
<td>Symbol</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>hhXpt</td>
<td>190</td>
</tr>
<tr>
<td>hiparallelm</td>
<td>186, 195</td>
</tr>
<tr>
<td>hiperpm</td>
<td>186, 195</td>
</tr>
<tr>
<td>hjrns</td>
<td>184, 193</td>
</tr>
<tr>
<td>Hmax</td>
<td>181</td>
</tr>
<tr>
<td>hrms</td>
<td>190</td>
</tr>
<tr>
<td>hs</td>
<td>238 240</td>
</tr>
<tr>
<td>hse</td>
<td>238 240</td>
</tr>
<tr>
<td>hydro.f90</td>
<td></td>
</tr>
<tr>
<td>ialive</td>
<td>25 173</td>
</tr>
<tr>
<td>ialive=0</td>
<td>26</td>
</tr>
<tr>
<td>idiajbm</td>
<td>98</td>
</tr>
<tr>
<td>IDL_PATH</td>
<td>5 55</td>
</tr>
<tr>
<td>idx_tavg</td>
<td>30 31 173</td>
</tr>
<tr>
<td>iforce</td>
<td>176</td>
</tr>
<tr>
<td>iforce2</td>
<td>176</td>
</tr>
<tr>
<td>iheatcond</td>
<td>174 176</td>
</tr>
<tr>
<td>imax</td>
<td>54</td>
</tr>
<tr>
<td>in</td>
<td>239 240</td>
</tr>
<tr>
<td>in0</td>
<td>239 240</td>
</tr>
<tr>
<td>ind</td>
<td>239 240</td>
</tr>
<tr>
<td>inf</td>
<td>239 240</td>
</tr>
<tr>
<td>init_ads_mol_frac</td>
<td>171</td>
</tr>
<tr>
<td>init_surf_mol_frac</td>
<td>171</td>
</tr>
<tr>
<td>initaa</td>
<td>169</td>
</tr>
<tr>
<td>initaa2</td>
<td>170</td>
</tr>
<tr>
<td>initlnnc</td>
<td>170</td>
</tr>
<tr>
<td>initlnnc2</td>
<td>170</td>
</tr>
<tr>
<td>initlnrho</td>
<td>167</td>
</tr>
<tr>
<td>initpower</td>
<td>128</td>
</tr>
<tr>
<td>initpower2</td>
<td>128</td>
</tr>
<tr>
<td>initpower2=5/3</td>
<td>128</td>
</tr>
<tr>
<td>initpower=2</td>
<td>128</td>
</tr>
<tr>
<td>initpower4</td>
<td>128</td>
</tr>
<tr>
<td>initpower_aa=2</td>
<td>128</td>
</tr>
<tr>
<td>initss</td>
<td>168</td>
</tr>
<tr>
<td>inituu</td>
<td>166</td>
</tr>
<tr>
<td>ioc</td>
<td>234 235</td>
</tr>
<tr>
<td>ip</td>
<td>35 164 171</td>
</tr>
<tr>
<td>Iring1,Iring2</td>
<td>170</td>
</tr>
<tr>
<td>isav</td>
<td>25</td>
</tr>
<tr>
<td>isave</td>
<td>172</td>
</tr>
<tr>
<td>ism</td>
<td>238</td>
</tr>
<tr>
<td>isothtop</td>
<td>169 175</td>
</tr>
<tr>
<td>Isurf</td>
<td>215</td>
</tr>
<tr>
<td>it</td>
<td>8 33 34 178</td>
</tr>
<tr>
<td>it1</td>
<td>25 29 172</td>
</tr>
<tr>
<td>it1d</td>
<td>29 172</td>
</tr>
<tr>
<td>itorder</td>
<td>36 172</td>
</tr>
<tr>
<td>iuut</td>
<td>100</td>
</tr>
<tr>
<td>ivar</td>
<td>108</td>
</tr>
<tr>
<td>ivisc</td>
<td>177</td>
</tr>
<tr>
<td>iwig</td>
<td>172</td>
</tr>
<tr>
<td>ix</td>
<td>261 172</td>
</tr>
<tr>
<td>iy</td>
<td>261 172</td>
</tr>
<tr>
<td>iz</td>
<td>261 172</td>
</tr>
<tr>
<td>iz2</td>
<td>261 172</td>
</tr>
<tr>
<td>j11rms</td>
<td>201 204</td>
</tr>
<tr>
<td>j2</td>
<td>215</td>
</tr>
<tr>
<td>j2m</td>
<td>182 183 191</td>
</tr>
<tr>
<td>j2mz</td>
<td>221 222</td>
</tr>
<tr>
<td>jb</td>
<td>98 215</td>
</tr>
<tr>
<td>jb0m</td>
<td>201 204 208</td>
</tr>
<tr>
<td>jb_int</td>
<td>182 191</td>
</tr>
<tr>
<td>jbm</td>
<td>182 191</td>
</tr>
<tr>
<td>jbmh</td>
<td>182 191</td>
</tr>
<tr>
<td>jbn</td>
<td>182 192</td>
</tr>
<tr>
<td>jbmphi</td>
<td>216 217</td>
</tr>
<tr>
<td>jbm</td>
<td>182 192</td>
</tr>
<tr>
<td>jbxm</td>
<td>231</td>
</tr>
<tr>
<td>jbmz</td>
<td>220 222</td>
</tr>
<tr>
<td>jbrms</td>
<td>182 191</td>
</tr>
<tr>
<td>jbtm</td>
<td>182 191</td>
</tr>
<tr>
<td>jdel2am</td>
<td>233 235</td>
</tr>
<tr>
<td>jet</td>
<td>186</td>
</tr>
<tr>
<td>jj</td>
<td>215</td>
</tr>
<tr>
<td>jm</td>
<td>187</td>
</tr>
<tr>
<td>jm2</td>
<td>182 191</td>
</tr>
<tr>
<td>jmax</td>
<td>184 187 193</td>
</tr>
<tr>
<td>jmbmz</td>
<td>185 194</td>
</tr>
<tr>
<td>jmin</td>
<td>187</td>
</tr>
<tr>
<td>jmx</td>
<td>184 194</td>
</tr>
<tr>
<td>jmy</td>
<td>185 194</td>
</tr>
<tr>
<td>jmz</td>
<td>185 194</td>
</tr>
<tr>
<td>jparalellm</td>
<td>186 195</td>
</tr>
<tr>
<td>jperpm</td>
<td>186 195</td>
</tr>
<tr>
<td>jrms</td>
<td>184 187 193</td>
</tr>
<tr>
<td>jx2m</td>
<td>188</td>
</tr>
<tr>
<td>jxaprms</td>
<td>187</td>
</tr>
<tr>
<td>jxrms</td>
<td>187</td>
</tr>
<tr>
<td>jxbm</td>
<td>185 194</td>
</tr>
<tr>
<td>jxbmx</td>
<td>185 194</td>
</tr>
<tr>
<td>jxbmz</td>
<td>185 194</td>
</tr>
<tr>
<td>jxb2m</td>
<td>186 195</td>
</tr>
<tr>
<td>jxbmax</td>
<td>186 195</td>
</tr>
<tr>
<td>jxbzm</td>
<td>183 192</td>
</tr>
<tr>
<td>jxbym</td>
<td>183 192</td>
</tr>
<tr>
<td>jxb2m</td>
<td>183 192</td>
</tr>
<tr>
<td>jxgLamrms</td>
<td>187</td>
</tr>
<tr>
<td>jxmx</td>
<td>230 231</td>
</tr>
<tr>
<td>jxmxz</td>
<td>228</td>
</tr>
<tr>
<td>jxmx</td>
<td>220 221</td>
</tr>
<tr>
<td>jxmx</td>
<td>230 231</td>
</tr>
<tr>
<td>jy2m</td>
<td>188</td>
</tr>
<tr>
<td>jybxm</td>
<td>183 192</td>
</tr>
<tr>
<td>jybym</td>
<td>183 192</td>
</tr>
<tr>
<td>jybz2m</td>
<td>188</td>
</tr>
<tr>
<td>jym</td>
<td>220 221</td>
</tr>
<tr>
<td>jymax</td>
<td>184 188 193</td>
</tr>
<tr>
<td>jymyx</td>
<td>230 231</td>
</tr>
<tr>
<td>jymxz</td>
<td>228</td>
</tr>
<tr>
<td>jymz</td>
<td>220 221</td>
</tr>
<tr>
<td>jyp2</td>
<td>183 192</td>
</tr>
<tr>
<td>jypt</td>
<td>183 192</td>
</tr>
<tr>
<td>je2m</td>
<td>188</td>
</tr>
<tr>
<td>jebxm</td>
<td>183 192</td>
</tr>
<tr>
<td>jebym</td>
<td>183 192</td>
</tr>
<tr>
<td>jebzm</td>
<td>183 192</td>
</tr>
<tr>
<td>jxbr2m</td>
<td>228</td>
</tr>
<tr>
<td>jxbrmx</td>
<td>186 195</td>
</tr>
<tr>
<td>jxbrmz</td>
<td>185 194</td>
</tr>
<tr>
<td>jxbrmz</td>
<td>185 194</td>
</tr>
<tr>
<td>jypt2</td>
<td>183 192</td>
</tr>
<tr>
<td>jzpt</td>
<td>183 192</td>
</tr>
<tr>
<td>k_forc</td>
<td>174</td>
</tr>
<tr>
<td>kap11</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap11z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap12</td>
<td>210 211 213</td>
</tr>
<tr>
<td>kap12z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap13</td>
<td>210 211 213</td>
</tr>
<tr>
<td>kap13z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap21</td>
<td>210 211 213</td>
</tr>
<tr>
<td>kap21z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap22</td>
<td>210 211 213</td>
</tr>
<tr>
<td>kap22z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap23</td>
<td>210 211 213</td>
</tr>
<tr>
<td>kap23z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap31</td>
<td>210 211 213</td>
</tr>
<tr>
<td>kap31z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap32</td>
<td>210 211 213</td>
</tr>
<tr>
<td>kap32z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kap33</td>
<td>210 211 213</td>
</tr>
<tr>
<td>kap33z</td>
<td>210 212 213</td>
</tr>
<tr>
<td>kcapPARA</td>
<td>211</td>
</tr>
<tr>
<td>kcapPARAz</td>
<td>211</td>
</tr>
<tr>
<td>Variable</td>
<td>Line Numbers</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>spectrum.f90</td>
<td>163</td>
</tr>
<tr>
<td>noyinyang.f90</td>
<td>163</td>
</tr>
<tr>
<td>noyinyang_mpi.f90</td>
<td>163</td>
</tr>
<tr>
<td>npm</td>
<td>196</td>
</tr>
<tr>
<td>nprocy</td>
<td>50</td>
</tr>
<tr>
<td>nprocz</td>
<td>50</td>
</tr>
<tr>
<td>nr</td>
<td>233</td>
</tr>
<tr>
<td>nr_directions</td>
<td>54</td>
</tr>
<tr>
<td>nt</td>
<td>8, 36, 171</td>
</tr>
<tr>
<td>nu</td>
<td>17, 209</td>
</tr>
<tr>
<td>nu_epicycle</td>
<td>168, 177</td>
</tr>
<tr>
<td>nu_hyper2</td>
<td>177</td>
</tr>
<tr>
<td>nu_hyper3</td>
<td>177</td>
</tr>
<tr>
<td>nu_LES</td>
<td>214</td>
</tr>
<tr>
<td>nu_Idep</td>
<td>214</td>
</tr>
<tr>
<td>num</td>
<td>214</td>
</tr>
<tr>
<td>numx</td>
<td>227</td>
</tr>
<tr>
<td>nuQ</td>
<td>209</td>
</tr>
<tr>
<td>nusmags</td>
<td>214</td>
</tr>
<tr>
<td>nusmagsmax</td>
<td>214</td>
</tr>
<tr>
<td>nusmagsmin</td>
<td>214</td>
</tr>
<tr>
<td>nv</td>
<td>108</td>
</tr>
<tr>
<td>nvar</td>
<td>26, 83</td>
</tr>
<tr>
<td>nxgrid</td>
<td>53, 54, 108</td>
</tr>
<tr>
<td>ny</td>
<td>28, 29, 108, 144</td>
</tr>
<tr>
<td>nngird</td>
<td>41</td>
</tr>
<tr>
<td>nz</td>
<td>28, 29, 108, 144</td>
</tr>
<tr>
<td>nzgrid</td>
<td>41</td>
</tr>
<tr>
<td>o2</td>
<td>215</td>
</tr>
<tr>
<td>o2m</td>
<td>180</td>
</tr>
<tr>
<td>o2mx</td>
<td>217</td>
</tr>
<tr>
<td>odel2um</td>
<td>180</td>
</tr>
<tr>
<td>ogux2mz</td>
<td>218</td>
</tr>
<tr>
<td>oguxmx</td>
<td>218</td>
</tr>
<tr>
<td>oguy2mz</td>
<td>218</td>
</tr>
<tr>
<td>oguyzm</td>
<td>218</td>
</tr>
<tr>
<td>oguz2mz</td>
<td>218</td>
</tr>
<tr>
<td>oguzmz</td>
<td>218</td>
</tr>
<tr>
<td>omax</td>
<td>180</td>
</tr>
<tr>
<td>Omega</td>
<td>173</td>
</tr>
<tr>
<td>omega_ff</td>
<td>177</td>
</tr>
<tr>
<td>omumz</td>
<td>179</td>
</tr>
<tr>
<td>oo</td>
<td>215</td>
</tr>
<tr>
<td>orms</td>
<td>180</td>
</tr>
<tr>
<td>ou0</td>
<td>239, 240</td>
</tr>
<tr>
<td>ou_int</td>
<td>180</td>
</tr>
<tr>
<td>oud</td>
<td>239, 240</td>
</tr>
<tr>
<td>ouf</td>
<td>239, 240</td>
</tr>
<tr>
<td>oum</td>
<td>25, 180</td>
</tr>
<tr>
<td>oumphi</td>
<td>180</td>
</tr>
<tr>
<td>oumx</td>
<td>225</td>
</tr>
<tr>
<td>oumxy</td>
<td>229</td>
</tr>
<tr>
<td>oumzx</td>
<td>227</td>
</tr>
<tr>
<td>oumy</td>
<td>223</td>
</tr>
<tr>
<td>oumz</td>
<td>218</td>
</tr>
<tr>
<td>out</td>
<td>233, 236, 239, 240</td>
</tr>
<tr>
<td>out1</td>
<td>172</td>
</tr>
<tr>
<td>out2</td>
<td>172</td>
</tr>
<tr>
<td>outm</td>
<td>178</td>
</tr>
<tr>
<td>our</td>
<td>233, 234, 239, 240</td>
</tr>
<tr>
<td>ox2m</td>
<td>180</td>
</tr>
<tr>
<td>ox2mx</td>
<td>225</td>
</tr>
<tr>
<td>ox2mz</td>
<td>217</td>
</tr>
<tr>
<td>oxdivu2mz</td>
<td>218</td>
</tr>
<tr>
<td>oxdivumz</td>
<td>218</td>
</tr>
<tr>
<td>oxmxy</td>
<td>229</td>
</tr>
<tr>
<td>oxmx</td>
<td>217</td>
</tr>
<tr>
<td>oxmz</td>
<td>180</td>
</tr>
<tr>
<td>oxozm</td>
<td>180</td>
</tr>
<tr>
<td>oxuxxmxz</td>
<td>218</td>
</tr>
<tr>
<td>oxuxymz</td>
<td>218</td>
</tr>
<tr>
<td>oxuxzm</td>
<td>180</td>
</tr>
<tr>
<td>oxuxzmx</td>
<td>218</td>
</tr>
<tr>
<td>oy2m</td>
<td>180</td>
</tr>
<tr>
<td>oy2mx</td>
<td>225</td>
</tr>
<tr>
<td>oy2mz</td>
<td>217</td>
</tr>
<tr>
<td>oydivu2mz</td>
<td>218</td>
</tr>
<tr>
<td>oydivumz</td>
<td>218</td>
</tr>
<tr>
<td>oymxy</td>
<td>229</td>
</tr>
<tr>
<td>oymz</td>
<td>217</td>
</tr>
<tr>
<td>oyozm</td>
<td>180</td>
</tr>
<tr>
<td>oyuxymz</td>
<td>218</td>
</tr>
<tr>
<td>oyuyymz</td>
<td>218</td>
</tr>
<tr>
<td>oyuzym</td>
<td>180</td>
</tr>
<tr>
<td>oyuzymz</td>
<td>218</td>
</tr>
<tr>
<td>oz2m</td>
<td>180</td>
</tr>
<tr>
<td>oz2mx</td>
<td>225</td>
</tr>
<tr>
<td>oz2mz</td>
<td>217</td>
</tr>
<tr>
<td>ozdivu2mz</td>
<td>218</td>
</tr>
<tr>
<td>ozdivumz</td>
<td>218</td>
</tr>
<tr>
<td>ozmxy</td>
<td>229</td>
</tr>
<tr>
<td>ozmz</td>
<td>217</td>
</tr>
<tr>
<td>p</td>
<td>232, 234, 237, 239</td>
</tr>
<tr>
<td>p1D</td>
<td>238, 239</td>
</tr>
<tr>
<td>PATH</td>
<td>5, 6, 12</td>
</tr>
<tr>
<td>pdivum</td>
<td>181, 189, 196</td>
</tr>
<tr>
<td>peffmxz</td>
<td>229</td>
</tr>
<tr>
<td>PENCIL_HOME</td>
<td>5, 6, 75</td>
</tr>
<tr>
<td>PENCIL_HOST_ID</td>
<td>18</td>
</tr>
<tr>
<td>pertss</td>
<td>168</td>
</tr>
<tr>
<td>pfc</td>
<td>234, 237</td>
</tr>
<tr>
<td>pfe</td>
<td>238, 239</td>
</tr>
<tr>
<td>phi11</td>
<td>200, 203</td>
</tr>
<tr>
<td>phi12</td>
<td>200, 203</td>
</tr>
<tr>
<td>phi21</td>
<td>200, 203</td>
</tr>
<tr>
<td>phi22</td>
<td>200, 203</td>
</tr>
<tr>
<td>phi32</td>
<td>200, 203</td>
</tr>
<tr>
<td>phibmx</td>
<td>186, 195</td>
</tr>
<tr>
<td>phibmy</td>
<td>186, 195</td>
</tr>
<tr>
<td>phibmz</td>
<td>186, 195</td>
</tr>
<tr>
<td>phibzm</td>
<td>191</td>
</tr>
<tr>
<td>phiK</td>
<td>200, 203</td>
</tr>
<tr>
<td>phiM</td>
<td>200, 203</td>
</tr>
<tr>
<td>phim</td>
<td>191</td>
</tr>
<tr>
<td>phiMK</td>
<td>200, 203</td>
</tr>
<tr>
<td>phimphi</td>
<td>216</td>
</tr>
<tr>
<td>phi2</td>
<td>191</td>
</tr>
<tr>
<td>phipt</td>
<td>191</td>
</tr>
<tr>
<td>placeholder</td>
<td>178</td>
</tr>
<tr>
<td>polytrm</td>
<td>197</td>
</tr>
<tr>
<td>pot</td>
<td>238, 239</td>
</tr>
<tr>
<td>power_spectrum.f90</td>
<td>163</td>
</tr>
<tr>
<td>Poynting</td>
<td>215</td>
</tr>
<tr>
<td>poynxmxxy</td>
<td>231</td>
</tr>
<tr>
<td>poynymxy</td>
<td>231</td>
</tr>
<tr>
<td>poynzmxxy</td>
<td>231</td>
</tr>
<tr>
<td>poynznmz</td>
<td>221, 222</td>
</tr>
<tr>
<td>pp</td>
<td>215, 235, 236</td>
</tr>
<tr>
<td>ppm</td>
<td>181, 189, 198, 214</td>
</tr>
<tr>
<td>ppmx</td>
<td>225, 227</td>
</tr>
<tr>
<td>pmmx</td>
<td>219, 222, 223</td>
</tr>
<tr>
<td>ppmz</td>
<td>219, 222, 223</td>
</tr>
<tr>
<td>pr1mz</td>
<td>223</td>
</tr>
<tr>
<td>pretend_InTT</td>
<td>165</td>
</tr>
<tr>
<td>pscalar_diff</td>
<td>176</td>
</tr>
<tr>
<td>PSCALAR_INT_PARS</td>
<td>165</td>
</tr>
<tr>
<td>pscalar_sink_rate</td>
<td>177</td>
</tr>
<tr>
<td>psi11</td>
<td>200, 203</td>
</tr>
<tr>
<td>psi12</td>
<td>200, 203</td>
</tr>
<tr>
<td>psi21</td>
<td>200, 203</td>
</tr>
<tr>
<td>psi22</td>
<td>200, 203</td>
</tr>
<tr>
<td>puzm</td>
<td>223</td>
</tr>
<tr>
<td>puzmz</td>
<td>180</td>
</tr>
<tr>
<td>puzmxy</td>
<td>229</td>
</tr>
<tr>
<td>pwd</td>
<td>238, 239</td>
</tr>
<tr>
<td>q2m</td>
<td>180</td>
</tr>
</tbody>
</table>
VARIABLE INDEX

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>qam</td>
<td>195</td>
</tr>
<tr>
<td>qem</td>
<td>195</td>
</tr>
<tr>
<td>qfm</td>
<td>180</td>
</tr>
<tr>
<td>qfviscm</td>
<td>214</td>
</tr>
<tr>
<td>qmax</td>
<td>180, 186, 191</td>
</tr>
<tr>
<td>qom</td>
<td>180</td>
</tr>
<tr>
<td>qpm</td>
<td>195</td>
</tr>
<tr>
<td>qpmz</td>
<td>222</td>
</tr>
<tr>
<td>Qrad</td>
<td>215</td>
</tr>
<tr>
<td>qrms</td>
<td>180, 186, 191</td>
</tr>
<tr>
<td>qsatmin</td>
<td>191</td>
</tr>
<tr>
<td>qsatrms</td>
<td>191</td>
</tr>
<tr>
<td>qshear</td>
<td>171, 177</td>
</tr>
<tr>
<td>qsm</td>
<td>195</td>
</tr>
<tr>
<td>quxom</td>
<td>180</td>
</tr>
<tr>
<td>qxmax</td>
<td>191</td>
</tr>
<tr>
<td>qxmin</td>
<td>191</td>
</tr>
<tr>
<td>qymax</td>
<td>191</td>
</tr>
<tr>
<td>qymin</td>
<td>191</td>
</tr>
<tr>
<td>qzmax</td>
<td>191</td>
</tr>
<tr>
<td>qzmin</td>
<td>191</td>
</tr>
<tr>
<td>r_ext</td>
<td>176</td>
</tr>
<tr>
<td>r_ff</td>
<td>177</td>
</tr>
<tr>
<td>r_int</td>
<td>176</td>
</tr>
<tr>
<td>radius</td>
<td>170</td>
</tr>
<tr>
<td>radius_ss</td>
<td>168</td>
</tr>
<tr>
<td>random_gen</td>
<td>165, 173</td>
</tr>
<tr>
<td>rcool</td>
<td>175</td>
</tr>
<tr>
<td>rcylmphi</td>
<td>216</td>
</tr>
<tr>
<td>rdamp</td>
<td>173</td>
</tr>
<tr>
<td>rdampext</td>
<td>173</td>
</tr>
<tr>
<td>rdampint</td>
<td>173</td>
</tr>
<tr>
<td>rdiumum</td>
<td>179</td>
</tr>
<tr>
<td>REAL_PRECISION</td>
<td>52</td>
</tr>
<tr>
<td>reinitialize_Incc</td>
<td>176</td>
</tr>
<tr>
<td>relhel</td>
<td>148, 177</td>
</tr>
<tr>
<td>relhel_\text{uu}=0</td>
<td>128</td>
</tr>
<tr>
<td>relhel_\text{uu}=1</td>
<td>128</td>
</tr>
<tr>
<td>Remz</td>
<td>218</td>
</tr>
<tr>
<td>Reshock</td>
<td>214</td>
</tr>
<tr>
<td>rho</td>
<td>215</td>
</tr>
<tr>
<td>rho0</td>
<td>167, 174, 177</td>
</tr>
<tr>
<td>rho2mx</td>
<td>219</td>
</tr>
<tr>
<td>rho2mz</td>
<td>218</td>
</tr>
<tr>
<td>rho_left</td>
<td>167</td>
</tr>
<tr>
<td>rho_right</td>
<td>167</td>
</tr>
<tr>
<td>rhoccn</td>
<td>186</td>
</tr>
<tr>
<td>rhom</td>
<td>8, 25, 181, 187</td>
</tr>
<tr>
<td>rhomax</td>
<td>181, 189</td>
</tr>
<tr>
<td>rhomin</td>
<td>181, 189</td>
</tr>
<tr>
<td>rhomphi</td>
<td>216</td>
</tr>
<tr>
<td>rhomx</td>
<td>225</td>
</tr>
<tr>
<td>rhomxmask</td>
<td>181</td>
</tr>
<tr>
<td>rhomxy</td>
<td>230</td>
</tr>
<tr>
<td>rhomxz</td>
<td>227</td>
</tr>
<tr>
<td>rhomy</td>
<td>223</td>
</tr>
<tr>
<td>rhomz</td>
<td>218</td>
</tr>
<tr>
<td>rhomzmask</td>
<td>181</td>
</tr>
<tr>
<td>rl2x2m</td>
<td>180</td>
</tr>
<tr>
<td>rlz2m</td>
<td>180</td>
</tr>
<tr>
<td>rlxz2m</td>
<td>180</td>
</tr>
<tr>
<td>rlzm</td>
<td>180</td>
</tr>
<tr>
<td>rlz2m</td>
<td>180</td>
</tr>
<tr>
<td>Rmesh</td>
<td>178</td>
</tr>
<tr>
<td>Rmesh3</td>
<td>176</td>
</tr>
<tr>
<td>Rmmz</td>
<td>185</td>
</tr>
<tr>
<td>rmphi</td>
<td>216</td>
</tr>
<tr>
<td>Rring1,Rring2</td>
<td>170</td>
</tr>
<tr>
<td>rumax</td>
<td>179</td>
</tr>
<tr>
<td>rux2m</td>
<td>179</td>
</tr>
<tr>
<td>rux2m2x</td>
<td>225</td>
</tr>
<tr>
<td>rux2mxy</td>
<td>229</td>
</tr>
<tr>
<td>rux2m2z</td>
<td>218</td>
</tr>
<tr>
<td>ruxm</td>
<td>179</td>
</tr>
<tr>
<td>ruxmx</td>
<td>225</td>
</tr>
<tr>
<td>ruxmxy</td>
<td>229</td>
</tr>
<tr>
<td>ruxmxz</td>
<td>218</td>
</tr>
<tr>
<td>ruxtot</td>
<td>179</td>
</tr>
<tr>
<td>ruxy2mz</td>
<td>218</td>
</tr>
<tr>
<td>ruxy2mz</td>
<td>229</td>
</tr>
<tr>
<td>ruxym</td>
<td>225</td>
</tr>
<tr>
<td>ruxymx</td>
<td>180</td>
</tr>
<tr>
<td>ruxymxy</td>
<td>225</td>
</tr>
<tr>
<td>ruxzym</td>
<td>180</td>
</tr>
<tr>
<td>rux2mz</td>
<td>225</td>
</tr>
<tr>
<td>ruxzm</td>
<td>179</td>
</tr>
<tr>
<td>rux2mx</td>
<td>225</td>
</tr>
<tr>
<td>rux2mx2z</td>
<td>218</td>
</tr>
<tr>
<td>ruxzn</td>
<td>180</td>
</tr>
<tr>
<td>ruxz2mz</td>
<td>225</td>
</tr>
<tr>
<td>ruxz2mxy</td>
<td>229</td>
</tr>
<tr>
<td>ruxz2mx</td>
<td>225</td>
</tr>
<tr>
<td>ruxz2mxy2</td>
<td>218</td>
</tr>
<tr>
<td>ruym</td>
<td>179</td>
</tr>
<tr>
<td>ruymx</td>
<td>225</td>
</tr>
<tr>
<td>ruymxy</td>
<td>229</td>
</tr>
<tr>
<td>ruymz</td>
<td>218</td>
</tr>
<tr>
<td>ruyuz2mz</td>
<td>218</td>
</tr>
<tr>
<td>ruyuzmx</td>
<td>180</td>
</tr>
<tr>
<td>ruyuz2mx</td>
<td>225</td>
</tr>
<tr>
<td>ruysmz</td>
<td>225</td>
</tr>
<tr>
<td>ruz2m</td>
<td>179</td>
</tr>
<tr>
<td>ruz2mx</td>
<td>225</td>
</tr>
<tr>
<td>ruz2mxy</td>
<td>229</td>
</tr>
<tr>
<td>ruz2mxz</td>
<td>218</td>
</tr>
<tr>
<td>ruzdownmz</td>
<td>217</td>
</tr>
<tr>
<td>ruzm</td>
<td>179</td>
</tr>
<tr>
<td>ruzmx</td>
<td>225</td>
</tr>
<tr>
<td>ruzmxy</td>
<td>229</td>
</tr>
<tr>
<td>ruzmz</td>
<td>218</td>
</tr>
<tr>
<td>ruzupmz</td>
<td>217</td>
</tr>
<tr>
<td>s</td>
<td>232, 234, 237, 239</td>
</tr>
<tr>
<td>s+f</td>
<td>236, 237</td>
</tr>
<tr>
<td>s0d</td>
<td>232, 234, 237, 239</td>
</tr>
<tr>
<td>s2khDFm</td>
<td>200, 204, 207</td>
</tr>
<tr>
<td>sa2</td>
<td>233, 235</td>
</tr>
<tr>
<td>sds</td>
<td>235, 236</td>
</tr>
<tr>
<td>sep</td>
<td>236, 237</td>
</tr>
<tr>
<td>set</td>
<td>233, 234, 236, 238, 240</td>
</tr>
<tr>
<td>sf</td>
<td>232, 237, 239</td>
</tr>
<tr>
<td>sfr</td>
<td>233, 235, 237</td>
</tr>
<tr>
<td>Shchm</td>
<td>196</td>
</tr>
<tr>
<td>shock</td>
<td>215</td>
</tr>
<tr>
<td>shockmax</td>
<td>197</td>
</tr>
<tr>
<td>shx</td>
<td>233</td>
</tr>
<tr>
<td>shy</td>
<td>233</td>
</tr>
<tr>
<td>shz</td>
<td>233</td>
</tr>
<tr>
<td>Sij2m</td>
<td>214</td>
</tr>
<tr>
<td>slice_postion</td>
<td>26, 27, 172</td>
</tr>
<tr>
<td>slo</td>
<td>233, 234</td>
</tr>
<tr>
<td>slp</td>
<td>233</td>
</tr>
<tr>
<td>srd</td>
<td>233, 235</td>
</tr>
<tr>
<td>spr</td>
<td>232, 234</td>
</tr>
<tr>
<td>spt</td>
<td>236, 237</td>
</tr>
<tr>
<td>sr1</td>
<td>233</td>
</tr>
<tr>
<td>ss</td>
<td>215, 232, 234, 236</td>
</tr>
<tr>
<td>ss2m</td>
<td>181, 189</td>
</tr>
<tr>
<td>ss2mx</td>
<td>225</td>
</tr>
<tr>
<td>ss2mz</td>
<td>219</td>
</tr>
<tr>
<td>sse</td>
<td>236, 237</td>
</tr>
<tr>
<td>ssm</td>
<td>8, 181, 189, 198</td>
</tr>
<tr>
<td>ssmx</td>
<td>182</td>
</tr>
<tr>
<td>ssmphi</td>
<td>182</td>
</tr>
<tr>
<td>ssmx</td>
<td>225</td>
</tr>
<tr>
<td>ssmy</td>
<td>190, 230</td>
</tr>
<tr>
<td>ssxz</td>
<td>190, 227</td>
</tr>
</tbody>
</table>
ssmy ......... 224
ssmz .......... 219 223
ssruzm ....... 181
ssuuzm ........ 181
sT ........... 233 236 238 240
StokesImxy .... 231 232
StokesQ1mxy ... 231 232
StokesQmxy .... 231 232
StokesU1mxy ... 231 232
StokesUmxy ..... 231 232
strTpt .......... 190
strXpt .......... 190
StS ........... 238 240
t ........... 8 25 108 178
tauheat_buffer ... 175
tauhtnmin ...... 181
tavg .......... 30 31 178
tdamp .......... 173
tDxpm .......... 197
tDydm .......... 197
tDzpm .......... 197
tensor_pscaler_diff ... 176
thcool ....... 198
theta .......... 178
theta5m .......... 188
theta5rms ....... 188
timestep.f90 ........ 163
timestep_strang.f90 ... 163
timestep_subcycle,f90 ... 163
tmax .......... 172
tot_ang_mom ....... 180
ttotal_carbon_sites ... 171
totmass ........ 181
Trms .......... 197
ts ........... 44
TT ........... 215
TT2m .......... 197
TT2mx ........... 225
TT2mz .......... 219 223
TTheat_buffer ... 175
TTm ........... 181 197 198 214
TTmax ....... 182 197 198 214
TTmin ........... 182 197 198 214
TTmx ........... 225 227
TTmxy .......... 230 232
TTmz ........... 227 229
TTmy ........... 224
TTmz ........... 219 223
ttransient ....... 173
TTtop .......... 181 190
TT2mask ....... 190
TugTm .......... 197
Tugux_uugTm ... 197
Tuguy_uugTm ... 197
Tuguz_uugTm ... 197
u0rms .......... 201 204
u11rms .......... 201 204
u12rms .......... 201 204
u1u23m .......... 179
u1u32m .......... 179
u2 .......... 214
u21rms .......... 201 204
u22rms .......... 201 204
u2m .......... 178
u2mphi .......... 216
u2mz .......... 217
u2tm .......... 178
u2u13m .......... 179
u2u31m .......... 179
u3u12m .......... 179
u3u21m .......... 179
uabzmz ....... 219 221
uabymz ....... 219 221
uabzmz ....... 219 221
uam .......... 183 192
uamz .......... 220 222
ubbzm .......... 182 191
ubm .......... 182 192
ubmz .......... 220 222
ubs .......... 239 240
udpxxm ........ 181
ufpresm ....... 182
ufviscm ....... 214
uqlrhom .......... 181
uqlrhomz ........ 219
ugm .......... 190
ugrhom .......... 181 187
ugrhomz ........ 219
ugu2m .......... 180
ugurmsx .......... 180
uguxmxy .......... 229
uguymxy .......... 229
uguzmxy .......... 229
ujm .......... 183 192
ujxbm .......... 186 195
umamz .......... 179
umax .......... 8 178
umbmz .......... 179
uambmz .......... 179
umin .......... 178
umx .......... 29 179
umxbmz .......... 179
umy .......... 29 179
umz .......... 29 179
unit_density ....... 37 165
unit_length ....... 37 165
unit_system ....... 37 165
unit_temperature ... 36 37
unit_velocity ....... 37 165
uotm .......... 178
umphi .......... 216
urand .......... 166
urmphi .......... 216
urms ........ 8 25 128 178
urms=3.981E-01 .......... 128
urms=5.560E-01 .......... 128
urmsx .......... 178
urmsz .......... 178
ursphmphi .......... 216
uthmphi .......... 216
uu .......... 214
uu_left .......... 166
uu_right .......... 166
uumphi .......... 216
uusphmphi .......... 216
uut .......... 100
ux0m .......... 201 204
ux0mx .......... 209
ux11m .......... 201 204
ux2cm .......... 179
ux2m .......... 179
ux2mx .......... 229
ux2mxy .......... 229
ux2mz .......... 227
ux2mxz .......... 227
ux2mx .......... 217
ux2ssm .......... 179
uxbm ........ 185 194
uxbmx ........ 185 194
uxbmy ........ 185 194
uxbmz ........ 185 194
uxbzm .......... 182 192
uxbzmz .......... 220 222
uxbxmz .......... 220 222
uxbxm .......... 183 192
uxbxmz .......... 220 222
uxbxm .......... 183 192
uxbxmz .......... 220 222
uxglnrym .......... 180
uxm .......... 179
uxmax .......... 179
<table>
<thead>
<tr>
<th>Variable</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uxmin</td>
<td>179</td>
</tr>
<tr>
<td>uxmx</td>
<td>225</td>
</tr>
<tr>
<td>uxmxy</td>
<td>229</td>
</tr>
<tr>
<td>uxmxz</td>
<td>227</td>
</tr>
<tr>
<td>uxmz</td>
<td>217</td>
</tr>
<tr>
<td>uxp2</td>
<td>178</td>
</tr>
<tr>
<td>uxt</td>
<td>178</td>
</tr>
<tr>
<td>uxrms</td>
<td>178</td>
</tr>
<tr>
<td>uXTm</td>
<td>197</td>
</tr>
<tr>
<td>uXTmz</td>
<td>223</td>
</tr>
<tr>
<td>uTTmx</td>
<td>225</td>
</tr>
<tr>
<td>uTTmxxy</td>
<td>230</td>
</tr>
<tr>
<td>uuxyzm</td>
<td>219</td>
</tr>
<tr>
<td>uuywdivum</td>
<td>180</td>
</tr>
<tr>
<td>uxuym</td>
<td>179</td>
</tr>
<tr>
<td>uuxymx</td>
<td>225</td>
</tr>
<tr>
<td>uuxymxy</td>
<td>229</td>
</tr>
<tr>
<td>uuxymxz</td>
<td>227</td>
</tr>
<tr>
<td>uuxymz</td>
<td>218</td>
</tr>
<tr>
<td>uuxzm</td>
<td>179</td>
</tr>
<tr>
<td>uuxzmx</td>
<td>225</td>
</tr>
<tr>
<td>uuxzmxxy</td>
<td>229</td>
</tr>
<tr>
<td>uuxzmxz</td>
<td>227</td>
</tr>
<tr>
<td>uuxzms</td>
<td>181</td>
</tr>
<tr>
<td>uuxrns</td>
<td>181</td>
</tr>
<tr>
<td>uuxrms</td>
<td>181</td>
</tr>
<tr>
<td>uy0m</td>
<td>201, 204</td>
</tr>
<tr>
<td>uy0mz</td>
<td>209</td>
</tr>
<tr>
<td>uy11m</td>
<td>201, 204</td>
</tr>
<tr>
<td>uy2ccm</td>
<td>179</td>
</tr>
<tr>
<td>uy2m</td>
<td>179</td>
</tr>
<tr>
<td>uy2mx</td>
<td>225</td>
</tr>
<tr>
<td>uy2mxy</td>
<td>229</td>
</tr>
<tr>
<td>uy2mxz</td>
<td>227</td>
</tr>
<tr>
<td>uy2mz</td>
<td>217</td>
</tr>
<tr>
<td>uy2ssm</td>
<td>179</td>
</tr>
<tr>
<td>uybmx</td>
<td>182, 192</td>
</tr>
<tr>
<td>uybmxz</td>
<td>228</td>
</tr>
<tr>
<td>uybym</td>
<td>220, 222</td>
</tr>
<tr>
<td>uybymz</td>
<td>183, 192</td>
</tr>
<tr>
<td>uybzm</td>
<td>220, 222</td>
</tr>
<tr>
<td>uybzmz</td>
<td>183, 192</td>
</tr>
<tr>
<td>uybzmz</td>
<td>228</td>
</tr>
<tr>
<td>uyglnrzm</td>
<td>180</td>
</tr>
<tr>
<td>uyglnrhomz</td>
<td>219</td>
</tr>
<tr>
<td>uym</td>
<td>179</td>
</tr>
<tr>
<td>uymax</td>
<td>179</td>
</tr>
<tr>
<td>uzpt</td>
<td>178</td>
</tr>
<tr>
<td>uzrms</td>
<td>179</td>
</tr>
<tr>
<td>uzrms</td>
<td>179</td>
</tr>
<tr>
<td>uzTm</td>
<td>197</td>
</tr>
<tr>
<td>uzTmz</td>
<td>223</td>
</tr>
<tr>
<td>uzTTmx</td>
<td>225</td>
</tr>
<tr>
<td>uzTTmxy</td>
<td>230</td>
</tr>
<tr>
<td>uzTTmz</td>
<td>219</td>
</tr>
<tr>
<td>uzupmz</td>
<td>217</td>
</tr>
<tr>
<td>uzys</td>
<td>181</td>
</tr>
<tr>
<td>v</td>
<td>232, 234, 237, 239</td>
</tr>
<tr>
<td>v3</td>
<td>235, 237, 239</td>
</tr>
<tr>
<td>vAmax</td>
<td>184, 188, 193</td>
</tr>
<tr>
<td>vAmin</td>
<td>188</td>
</tr>
<tr>
<td>vArmz</td>
<td>228, 229</td>
</tr>
<tr>
<td>visc_heatm</td>
<td>214</td>
</tr>
<tr>
<td>vol</td>
<td>181</td>
</tr>
<tr>
<td>upx2m</td>
<td>196</td>
</tr>
<tr>
<td>vpxm</td>
<td>196</td>
</tr>
<tr>
<td>vpxmax</td>
<td>196</td>
</tr>
<tr>
<td>upxmin</td>
<td>196</td>
</tr>
<tr>
<td>vrepsm</td>
<td>196</td>
</tr>
<tr>
<td>w_forc</td>
<td>174</td>
</tr>
<tr>
<td>walltime</td>
<td>178</td>
</tr>
<tr>
<td>wool</td>
<td>175</td>
</tr>
<tr>
<td>wdam</td>
<td>173, 174</td>
</tr>
<tr>
<td>wheat</td>
<td>175</td>
</tr>
<tr>
<td>width_ff</td>
<td>177</td>
</tr>
<tr>
<td>widthhaa</td>
<td>170</td>
</tr>
<tr>
<td>widthlnrho</td>
<td>167</td>
</tr>
<tr>
<td>widthss</td>
<td>168, 174, 175</td>
</tr>
<tr>
<td>widththu</td>
<td>166</td>
</tr>
<tr>
<td>win</td>
<td>239, 240</td>
</tr>
<tr>
<td>wr1,wr2</td>
<td>170</td>
</tr>
<tr>
<td>write_slices</td>
<td>27</td>
</tr>
<tr>
<td>wsnapsf90</td>
<td>172</td>
</tr>
<tr>
<td>xi</td>
<td>209</td>
</tr>
<tr>
<td>xiQ</td>
<td>209</td>
</tr>
<tr>
<td>xp2m</td>
<td>196</td>
</tr>
<tr>
<td>xpm</td>
<td>196</td>
</tr>
<tr>
<td>xpmx</td>
<td>196</td>
</tr>
<tr>
<td>xpmn</td>
<td>196</td>
</tr>
<tr>
<td>xy</td>
<td>172</td>
</tr>
<tr>
<td>xy2</td>
<td>172</td>
</tr>
<tr>
<td>xyz0</td>
<td>23, 25, 164</td>
</tr>
<tr>
<td>xyz_star</td>
<td>25</td>
</tr>
<tr>
<td>yH</td>
<td>215</td>
</tr>
<tr>
<td>Variable</td>
<td>Page Numbers</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>yHm</td>
<td>181, 198</td>
</tr>
<tr>
<td>yHmax</td>
<td>181, 198</td>
</tr>
<tr>
<td>yHmin</td>
<td>198</td>
</tr>
<tr>
<td>yin_yang.f90</td>
<td>163</td>
</tr>
<tr>
<td>yin_yang_mpi.f90</td>
<td>163</td>
</tr>
<tr>
<td>yy</td>
<td>235, 237</td>
</tr>
<tr>
<td>z0aa</td>
<td>170</td>
</tr>
<tr>
<td>z1</td>
<td>167</td>
</tr>
<tr>
<td>z2</td>
<td>168</td>
</tr>
<tr>
<td>yin_yang_mpi.f90</td>
<td>163</td>
</tr>
<tr>
<td>zbot_slice</td>
<td>26, 28, 172</td>
</tr>
<tr>
<td>zeta</td>
<td>177, 209</td>
</tr>
<tr>
<td>zetaQ</td>
<td>209</td>
</tr>
<tr>
<td>z0</td>
<td>170</td>
</tr>
<tr>
<td>zmphi</td>
<td>216</td>
</tr>
<tr>
<td>z0</td>
<td>170</td>
</tr>
<tr>
<td>z0</td>
<td>170</td>
</tr>
<tr>
<td>z0</td>
<td>170</td>
</tr>
<tr>
<td>z0</td>
<td>170</td>
</tr>
</tbody>
</table>
Index

This index contains options, names, definitions and commands. Files and variables have their own indexes.

'VAR10' 45
.r 45
.run 45
.svn 12
/trimall 45
$PENCIL_HOME/idl/files 46
2N-scheme 156
6th-order derivatives 153

cronab -e 106
CSH viii
Csh 115275
csh 75
CVS 2
CVS vii
Cvs 11


cvs-add-rundir 32
daainit 146
Data directory 14
datafiles 25
Density_init_pars 166
Density_run_pars 174
diff rho_hyper3_mesh=2


double precision 5144
Download 24475
Download forbidden 75
DX viii151232141


Emacs settings 125
Entropy 61
Entropy 154061
Entropy.f90 27
Entropy_init_pars 168
Entropy_run_pars 174
Equation of state 63
Error, diffusive 157
Etatest 146


f-array 100
f90 2122
F95 vii1
f95 21
FAQ 75
FBCX1 146
FBCX2 146
FC=mpiif90 81
ff 45
ff:varname 46
FFT 13
Fftp 52


Filters 137
Flag 163171
Flux rings 128
Forcing_run_pars 36
Fortran record 2630
Fortran record 26
fp-array 101
Frequently Asked Questions 75
ftp 44
Fully qualified hostname 18


G77 77
G95 5276
GDL 42
Gfortran viii52
Ghost points viii52
Ghost zones 3950
Git 2
Git 23


Glibc 76
Gnu Data Language 42
GNU gcc viii
Gnuplot 1541
gnuplot 44
Grav_init_pars 167
Grav_r 13
Grav_run_pars 146177
Gravitational Waves 71
Gravity 13
Gravity_simple 13
grep vii9596
grid, nonuniform 23


h-index 91
Hydro.f90 27
Hydro_init_pars 165
Hydro_run_pars 173
hyperdiffusivity 137
Hyperviscosity 137139


Icc 76


263
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>power</td>
<td>53, 54</td>
</tr>
<tr>
<td>pretend_InTT</td>
<td>62, 165</td>
</tr>
<tr>
<td>Programming style</td>
<td>97</td>
</tr>
<tr>
<td>Pscalar</td>
<td>104</td>
</tr>
<tr>
<td>Pscalar_init_pars</td>
<td>170</td>
</tr>
<tr>
<td>Pscalar_run_pars</td>
<td>176</td>
</tr>
<tr>
<td>Python</td>
<td>46</td>
</tr>
<tr>
<td>Radiative transfer</td>
<td>66</td>
</tr>
<tr>
<td>Readline</td>
<td>44</td>
</tr>
<tr>
<td>Regridding</td>
<td>144</td>
</tr>
<tr>
<td>Remeshing</td>
<td>44</td>
</tr>
<tr>
<td>RERUN file</td>
<td>34</td>
</tr>
<tr>
<td>restart-new-dir</td>
<td>145</td>
</tr>
<tr>
<td>Repeating</td>
<td>40, 44</td>
</tr>
<tr>
<td>rluwrap</td>
<td>44</td>
</tr>
<tr>
<td>Run directory</td>
<td>6</td>
</tr>
<tr>
<td>run.x</td>
<td>30</td>
</tr>
<tr>
<td>Run_pars</td>
<td>26-28, 34, 145</td>
</tr>
<tr>
<td>Runge-Kutta</td>
<td>156</td>
</tr>
<tr>
<td>Runge-Kutta time step</td>
<td>38</td>
</tr>
<tr>
<td>Runge-Kutta-Fehlberg time step</td>
<td>38</td>
</tr>
<tr>
<td>Scripts</td>
<td>31</td>
</tr>
<tr>
<td>Setup</td>
<td>3, 42, 75</td>
</tr>
<tr>
<td>Shear</td>
<td>40</td>
</tr>
<tr>
<td>Shear_init_pars</td>
<td>170</td>
</tr>
<tr>
<td>Shear_run_pars</td>
<td>146, 177</td>
</tr>
<tr>
<td>Shock viscosity</td>
<td>37, 63</td>
</tr>
<tr>
<td>SI units</td>
<td>36, 165</td>
</tr>
<tr>
<td>Sixth-order derivatives</td>
<td>153</td>
</tr>
<tr>
<td>slice files</td>
<td>28</td>
</tr>
<tr>
<td>Slice files</td>
<td>26, 27</td>
</tr>
<tr>
<td>Special module</td>
<td>103</td>
</tr>
<tr>
<td>start.csh</td>
<td>40</td>
</tr>
<tr>
<td>Stdout</td>
<td>25</td>
</tr>
<tr>
<td>Stratification</td>
<td>129</td>
</tr>
<tr>
<td>structure</td>
<td>55</td>
</tr>
<tr>
<td>Style</td>
<td>3, 97, 122</td>
</tr>
<tr>
<td>Sub</td>
<td>109</td>
</tr>
<tr>
<td>summarize-history</td>
<td>34</td>
</tr>
<tr>
<td>svn</td>
<td>2-4, 14, 11, 31, 32, 79, 117, 164, 171</td>
</tr>
<tr>
<td>svn annotate src/*.f90</td>
<td>3</td>
</tr>
<tr>
<td>svn mv file.f90 experimental</td>
<td>126</td>
</tr>
<tr>
<td>svn up sourceme.csh</td>
<td>87</td>
</tr>
<tr>
<td>svn up sourceme.sh</td>
<td>87</td>
</tr>
<tr>
<td>svn update</td>
<td>107</td>
</tr>
<tr>
<td>svn update -r ######</td>
<td>4</td>
</tr>
<tr>
<td>Svn/git</td>
<td>2, 3</td>
</tr>
<tr>
<td>Syscalls</td>
<td>76</td>
</tr>
<tr>
<td>tab</td>
<td>97</td>
</tr>
<tr>
<td>Tab characters</td>
<td>122</td>
</tr>
<tr>
<td>tag_names</td>
<td>45</td>
</tr>
<tr>
<td>Tcsh</td>
<td>5</td>
</tr>
<tr>
<td>teach/PencilCode/material/Burger Testfield method</td>
<td>71, 145</td>
</tr>
<tr>
<td>Time averages</td>
<td>30</td>
</tr>
<tr>
<td>Time step</td>
<td>38, 156</td>
</tr>
<tr>
<td>Toroidal averages</td>
<td>29</td>
</tr>
<tr>
<td>touch NEWDIR</td>
<td>34</td>
</tr>
<tr>
<td>touch NOERASE</td>
<td>164</td>
</tr>
<tr>
<td>touch RELOAD</td>
<td>171</td>
</tr>
<tr>
<td>uname</td>
<td>22</td>
</tr>
<tr>
<td>Underscore problem</td>
<td>77</td>
</tr>
<tr>
<td>Units</td>
<td>36, 165</td>
</tr>
<tr>
<td>Unix</td>
<td>1</td>
</tr>
<tr>
<td>Upwinding</td>
<td>37, 95, 154</td>
</tr>
<tr>
<td>use</td>
<td>108</td>
</tr>
<tr>
<td>Vector potential</td>
<td>61</td>
</tr>
<tr>
<td>Video files</td>
<td>26</td>
</tr>
<tr>
<td>Viscosity</td>
<td>37, 63</td>
</tr>
<tr>
<td>Viscosity_run_pars</td>
<td>146, 177</td>
</tr>
<tr>
<td>Weyl gauge</td>
<td></td>
</tr>
<tr>
<td>Whitespace</td>
<td>61</td>
</tr>
<tr>
<td>Xmgrace</td>
<td>76</td>
</tr>
<tr>
<td>Xmgrace</td>
<td>122</td>
</tr>
</tbody>
</table>
Id